UNIVERSITY OF SALERNO

Department of Computer Engineering, Electrical Engineering and Applied Mathematics

Master's Degree in Computer Engineering

Autonomous Vehicle Driving

MULTIMODAL END-TO-END DRIVING THROUGH CONDITIONAL IMITATION LEARNING

Supervisor:
Prof. Alessia Saggese

Author:
Francesco Sabbarese
ID 0622701190

Assistant Supervisor:
Prof. Antonio Greco
Prof. Mario Vento

Academic Year 2020-2021

Dedicated to my family.

CONTENTS

ABSTRACT

Autonomous Vehicles are an asset destinated to become central in the near future. As a result, all major car manufacturers are working to produce fully autonomous vehicles. Currently, the use of modular pipelines is predominant in car manufacturing. This strategy breaks down the complex problem of driving into a variety of tasks such as perception, localization, planning and control. Each of these modules requires a separate set of expert algorithms, often expensive to implement in terms of human labour and data labeling. The main alternative that has recently aroused interest is *End-To-End* driving. It is a different paradigm in which perception and control are learned simultaneously through a *Deep Neural Network*, capable of making driving decisions. These sensorimotor models are typically obtained by *Imitation Learning* from human demonstrations. The main advantage is the simplicity in the acquisition of labeled data through large fleets of human-driven vehicles. However, scaling End-To-End driving methods to behaviours more complex than simple lane keeping or lead vehicle following remains an open problem. To get to evaluate more complex behaviours there is a need for a safe environment for algorithm evaluation. For this purpose, a simulator may be used. *CARLA Simulator* is an open-source platform built from ground up for autonomous driving validation and prototyping. An AI driver is evaluated through a *Benchmark*, a standardized procedure that experimentally investigates the driving capabilities in reaching some target destination in an urban environment.

This thesis focuses on End-To-End autonomous driving. So far, most proposals relying on this paradigm assume RGB images as input sensor data. However, autonomous vehicles will not only be equipped with these cameras, but also with other sensors that provide accurate depth information (for example, depth cameras). Accordingly, this thesis analyses whether combining RGB and Depth modalities through *Early Fusion*, i.e. using RGBD data, produces better End-To-End AI drivers than relying on the three typical RGB channels. The data source is CARLA100, a dataset acquired on the homonymous simulator. With this we explore *Conditional Imitation Learning*, for which we provide a model not only with the perception of the environment and the control signals, but also a representation of the expert's intention corresponding to a high-level navigation instruction or command, in order to eliminate ambiguities at intersections.

Using the *CARLA Simulator* and *Conditional Imitation Learning*, we evaluate two architectures, characterized by feature extractors of different depths and propose an experimental procedure to understand some of the limitations of this method. This procedure requires that for a trained model, its driving capabilities are evaluated in a never observed scenario, but also through extensive benchmarking to compare this to well-known results. We show how the proposed architectures are not far from the *State of the Art* and are able to execute complex lateral and longitudinal manoeuvres, even in unseen environments, refining some crucial aspects such as those of *collisions*. However, we confirm some well-known limitations of the *Imitation Learning* approach such as dataset bias, a generalization issues, and training instabilities, all requiring further research before this methodology can graduate to real-world driving.

Chapter 1.

INTRODUCTION

1.1 AUTONOMOUS DRIVING

Autonomy in the context of robotics is the ability of a ***robot*** to operate without human intervention or control. A similar concept can be applied to ***Autonomous Vehicles (AVs)***. Robotic cars that can drive themselves are an asset imagined in the literature since cars have become a central asset in our society. From just imagination, this idea of having fully AVs is now considered as inevitable to happen at some point in the future. We can provide a definition for AVs [1]:

"An autonomous car is a vehicle capable of sensing its environment and operating without human involvement. A human passenger is not required to take control of the vehicle at any time, nor is a human passenger required to be present in the vehicle at all. An autonomous car can go anywhere a traditional car goes and do everything that an experienced human driver does."

Currently, this remains a tempting idea, not fully achievable.

The field of autonomous driving has long been investigated by researchers, but it has seen a surge of interest in the past decade, both from industry and academia. The accumulated knowledge in vehicle dynamics, breakthroughs in computer vision caused by the advent of deep learning and availability of new and more precise sensor have encouraged the research and industrial implementation.

Furthermore, an increase in public and market interest potential has precipitated the emergence of AVs with varying degrees of automation. However, robust automated driving has not yet been achieved in urban environments characterized by strong dynamism.

1.2 Motivation And Social Impact

While the widespread use of AV on large scales is not forthcoming, its potential impact and benefits can be predicted with some degree:

- ***Problems that can be solved:*** one of the most important is safety. According to ***World Health Organization (WHO)*** approximately 1.3 million people die each year because of road traffic crashes and most of these are caused by human errors, such as distraction or inaccurate decision making. As their purpose, autonomous vehicles are being developed with the promise of preventing accidents. No less important would be the benefits from the point of view of traffic mitigation and the reduction of CO_2 emissions.
- ***Opportunities and benefits:*** the scenarios for convenience and quality-of-life improvements are limitless. AVs could reduce transportation costs, providing more affordable mobility, reallocation of driving time considering that all occupants could safely pursue more productive or entertaining activities, but also transporting the mobility impaired and enhance independence for seniors.
- ***New trends:*** consuming ***Mobility as a Service (MaaS)*** since through the removal of the drivers from the equation, allowing companies can further profit by providing mobility as a service, in a cheaper way, as Uber or Lift does today.

1.3 LEVELS OF AUTONOMY

There is no standard single definition of an AV since it depends on what the vehicle can perform and in which context. The context in which an AV can operate is called ***Operational Design Domain (ODD)***, that is operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or road characteristics.

As shown in Figure 1 ***Society of Automotive Engineers (SAE)*** currently defines 6 levels of driving automation ranging from Level 0 (fully manual) to Level 5 (fully autonomous):

- ***Level 0 (No Driving Automation)***: there is not automation involved, so the vehicle is totally controlled by a human driver. The human provides the *"dynamic driving task"* although there may be systems in place to help the driver. An example would be the emergency braking system—since it technically doesn't *"drive"* the vehicle, it does not qualify as automation.
- ***Level 1 (Driver Assistance)***: this is the lowest level of automation. The vehicle features a single automated system for driver assistance, such as steering or accelerating (*Cruise Control*). *Adaptive Cruise Control*, where the vehicle can be kept at a safe distance behind the next car, qualifies as Level 1 because the human driver monitors the other aspects of driving such as steering and braking.
- ***Level 2***: this means *Advanced Driver Assistance Systems* or *ADAS*. The vehicle can control both steering and accelerating/decelerating. Here the automation falls short of self-driving because a human sits in the driver's seat and can take control of the car at any time.

- ***Level 3 (Conditional Driving Automation)***: Vehicles have *"Environmental Detection"* capabilities and can make informed decisions for themselves, such as accelerating past a slow-moving vehicle. But they still require human override. The driver must remain alert and ready to take control if the system is unable to execute the task. In addition, the vehicle can operate only in a limited ODD.
- ***Level 4***: Under certain conditions specified in the ODD, the vehicle can take full control without need of the human driver. However, a human still has the option to manually override.
- ***Level 5***: vehicles do not require human attention—the *"dynamic driving task"* is eliminated. Level 5 cars won't even have steering wheels or accelerating/braking pedals. They will be free from geofencing, able to go anywhere and do anything that an experienced human driver can do. This level represents the full autonomy without any limitation on ODD.

Today, some autonomous vehicles are being marketed, which could be placed at levels 2 or 3 of autonomy. However, the levels of autonomy that could, in theory, fully provide both economic and safety benefits are levels 4 and 5. For this, several strategies are being developed, and these, including the one of interest of this thesis, they will be introduced in the following sections.

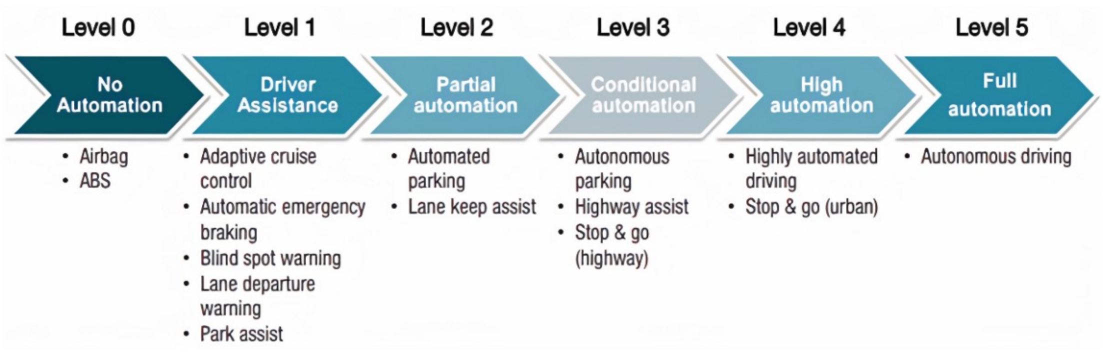

Figure 1: SAE Levels of Driving Automation.

1.4 ARCHITECTURAL APPROACHES

Building a Level 5 autonomous driving agent is a multi-disciplinary and complex task. It encapsulates many areas of knowledge such as sensors development, computer vision, machine learning, control theory, planning and even ethics. An autonomous vehicle must allow passengers to be transported to a desired destination, traveling in any city, and respecting the traffic rules. He must do it safely and therefore, he must also be able to react to any problematic situations that may possibly arise, such as inattentive drivers or sudden pedestrian crossings. In addition, everything must be done in any weather or road conditions.

To create a system capable of incorporating all these features, the two main approaches present in the scientific literature, as reported by [2], are the ***Modular*** and the ***End-to-End*** one, which are shown in Figure 2 and are covered in the following section.

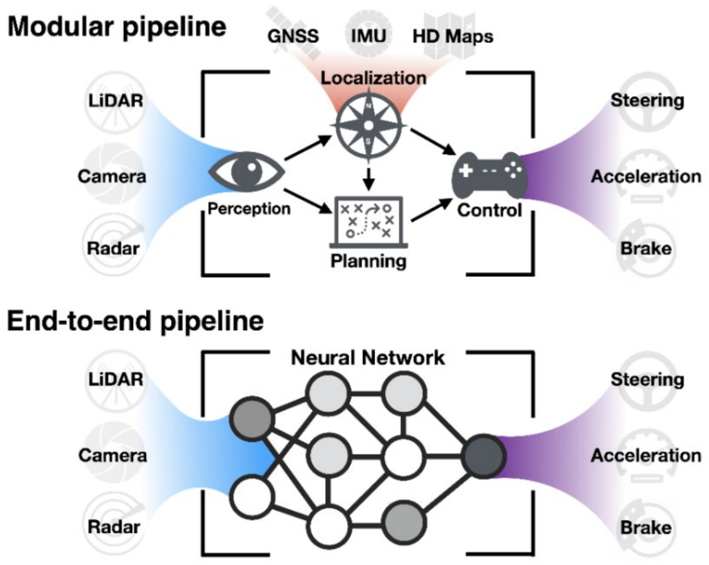

Figure 2: Modular pipeline vs End-To-End pipeline

1.4.1 MODULAR SYSTEMS

This represents the main paradigm currently used to produce self-driving vehicles in which several modules are interconnected to form a ***Modular Pipeline***.

Typical pipelines start with feeding raw data from sensors in input to system. From these the driving agent needs to be able to understand the scene as it drives through the environment. For that, it needs to detect all the moving objects and recognize them, as well as segment the road and other parts of the scene ***(Perception)***. It is also necessary to localize the AV on the road ***(Localization)*** and create a plan to reach the destination ***(Planning)***. Finally, it is necessary to generate a control command that determines its behaviour ***(Control)***.

Each module requires a separate set of expert algorithms which are costly in terms of development effort, integration, testing, and adaptability in the constantly changing open world.

The error propagation represents the main drawback, in addition to the need to predict a priori a set of rules to manage all the possible situations that may be encountered.

Developing individual modules separately divides the challenging task of automated driving into a series of easier-to-solve problems. These secondary tasks can benefit from knowledge transfer from other sectors such as robotics, computer vision and vehicle dynamics. As a major advantage, such pipelines are interpretable so that it is possible to reliably reason about how the system arrived at certain driving decisions and in case of malfunction or unexpected behaviour of the system it is possible to identify the module at fault [2].

Recently, there has been huge advances in each of these specific modules, due to the development of deep learning allied with huge amounts of available data and cheaper and more effective sensors, even if planning and control continue to be based on a classical engineering approach.

1.4.2 END-TO-END DRIVING

Instead of learning each single component, it is proposed to optimize the entire driving pipeline from the processing of sensory inputs to generating steering and acceleration commands as a single automatic learning activity carried out by an ***Artificial Neural Network (ANN)***, capable of making driving decisions [2]. With this the model should learn optimal intermediate representations for the target task in the sense that the model can learn by itself what visual patterns to attend to, extracting task-specific features.

Usually, the architecture is simpler than the modular stack with much fewer components. While conceptually appealing, this simplicity leads to problems typical of ANNs like that of interpretability since, with few or intermediate outputs, it is difficult or even impossible to figure out why the model misbehaves as well as to explain why the model arrived at a specific driving decision. Finally, it is well known that ANNs are susceptible to ***Adversarial Attacks***. By making small, but carefully chosen changes to the inputs, the models can be fooled and tricked into making errors leading to severe safety concerns when operating in a high-risk domain such as autonomous driving.

There are several ways to train an end-to-end driving agent: the models are learned either in a supervised way through ***Imitation Learning (IL)***, i.e. the acquisition of driving skills by observing an expert demonstrator carrying out a task, or through the online improvement of a driving policy via ***Reinforcement Learning (RL)*** following the paradigm of *trial-and-error* learning in the environment.

The focus of this thesis is the End-To-End driving based on Imitation Learning.

A general architecture

Based on the most promising approaches in the End-To-End models a general architecture can be drawn to visualize what a modern End-to-End model looks like [2]. The architecture in Figure 3 has not been implemented, or at least not completely. The main inputs used are those from *cameras*, *vehicle status measurements*, *lidar* acquisitions and *maps (HD maps).* While the first ones can be obtained in a simple way and at a relatively low cost, the use of advanced sensors is limited. Other inputs can be provided such as *high-level commands (e.g. turn left, turn right)* or *route plan*. Inputs are, typically, processed using *Convolutional Neural Network (CNN)* or *Multilayer Perceptron (MLP)* for the extraction of driving oriented features, which are then concatenated and further processed either through *Recurrent Neural Network (RNN)* and/or through MLP. The RNNs are used when it is necessary to extract temporal information from sequences. The system output can be represented by actuation commands typically steering, brake and accelerator, *Waypoints* or *Costmap*. In the case of branched architectures, the high-level commands can be used to select the branch that determines the output of the model. Further outputs called *Side-Tasks* are added both to improve the internal representation of the system but also to provide an explainability of errors. Examples are *semantic segmentation* and *speed prediction*.

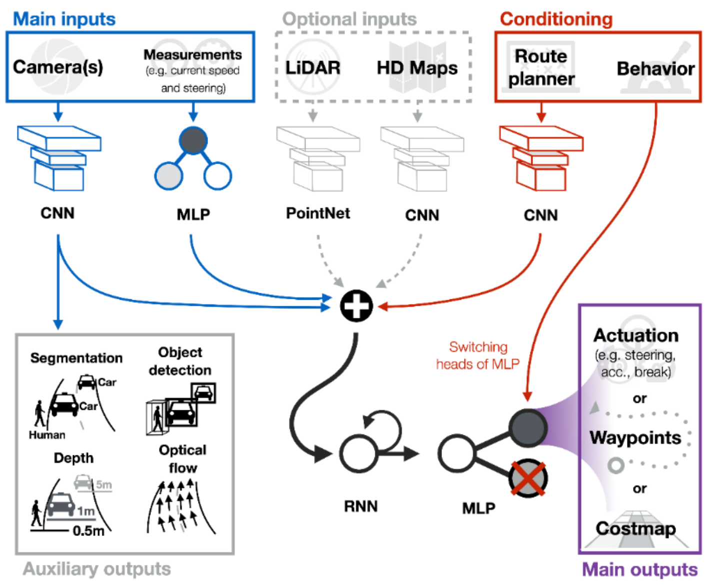

Figure 3: A general architecture for End-To-End driving.

1.5 Main Challenges

The following are the main challenges holding back the rise of fully AVs. These are not purely technological limitations, but also linked to ethical and bureaucratic issues.

Nondeterminism: AVs are complicated robotic systems that operate in indeterministic environment. The environmental variables, from weather conditions to surrounding human behaviour, are highly indeterministic and difficult to predict. Furthermore, system failures lead to accidents.

Acceptance: accidents caused by immature technology hinder public acceptance of AVs. Hence, the safety of these is questioned. On the other hand, even extremely cautious ADSs are making a negative impression.

Ethics: a set of challenges is posed by ethics. The question is that of the behaviour that an AV should perform in an unavoidable dangerous situation.

Certifications: certification of risk and reliability is yet another task to be solved. AVs must be designed with high redundancies that minimize the possibility of a failure.

Optimization: there are different optimization objectives such as time to reach the destination, fuel efficiency, comfort, making it more difficult a problem that already is. Therefore, carrying out all dynamic driving activities safely outside a well-defined area remains an open problem.

Chapter 2.

PROBLEM DEFINITION

In this chapter, ***Imitation Learning*** will be presented, providing a definition of the problem, contextualizing it in the autonomous driving sector, and analysing the main limitations with related possible solutions.

2.1 IMITATION LEARNING

While such tasks may be formulated as an optimization problem as in the Reinforcement Learning, it has become widely accepted that having prior knowledge provided by an expert is more effective and efficient then searching for a solution from scratch. A natural way to impart knowledge is to provide demonstrations for the desired behaviour that the learner is required to emulate.

Imitation Learning refers to an agent's ability to acquire skills or learn behaviours by observing an expert performing a task. The agent can learn from a set of collected instances. However, it often happens that direct imitation of the expert's behaviour is not sufficient due to task variations or inadequate demonstrations. Therefore, Imitation Learning techniques need to be able to learn a policy from the given demonstrations that can generalize to unseen scenarios. As such the agent learns to perform the task rather than deterministically copying the teacher.

Imitation Learning works by extracting information about the behaviour of the teacher and the surrounding environment, learning a ***mapping*** between the situations and demonstrated behaviours. Like traditional ***Supervised Learning*** where examples represent pairs of features and labels, in Imitation Learning the samples consist of ***state-action*** pairs.

Figure 4 show a typical learning flow. It begins with acquiring samples from an expert in terms of state-action pairs using different sensing methods. These data are processed to extract features capable to describe the state and surrounding of teacher. Features are then used to train a policy to mimic the demonstrated behaviours. Often learning the direct mapping between state and actions is not enough to achieve the desired behaviour. This can happen due to a number of issues such as error in acquiring the demonstrations, variance in the skeletons of the teacher and learner or insufficient demonstrations. Moreover, the task performed by the learner may slightly vary from the demonstrated task because of changes in the environment, obstacles, or targets. Therefore, Imitation Learning frequently involves another step to optimize the learned policy according to its performance about task. This can be thought as a post-learning step or can occurs in conjunction with learning from demonstration.

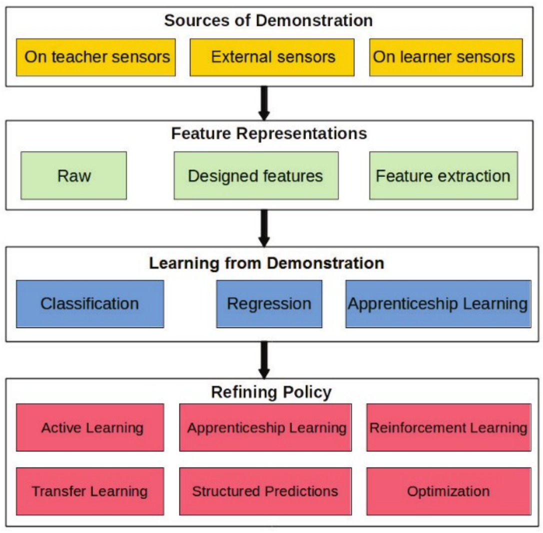

Figure 4: Learning flow for Imitation Learning.

2.1.1 FORMULATION

In this section we formalize the problem of Imitation Learning and introduce some preliminaries and definitions.

Definition 1. *The process of Imitation Learning is one by which an agent uses instances of performed actions to learn a policy that solves a given task.*

Definition 2. *An agent is defined as an entity that autonomously interacts within an environment towards achieving or optimizing a goal [3]. It receives information from the environment by sensing and acts upon the environment using a set of actuators.*

Definition 3. *A policy is a function that maps the state of the agent, such as positions and velocities to an action.*

Definition 4. *A demonstration is presented as a pair of input and output (s, a). s is a vector of features describing the state while a is the action performed by the demonstrator.*

A demonstration dataset $\mathcal{D} = \{\langle s_i^t,\ a_i^t \rangle\}_{i=0\ldots N}^{t=0\ldots T}$ contains a set of ***state-action*** pairs, where action $a_i^t \in \mathcal{A}$ is taken by expert at state $s_i^t \in \mathcal{S}$ under the guidance of an underlying policy π_E of the expert. Using the collected dataset $\mathcal{D}$, a common optimization-based strategy of Imitation Learning is to learn a policy $\pi^*: \mathcal{S} \to \mathcal{A}$ that mimic the expert' policy π_E by satisfying

$$\pi^* = argmin_\pi \, \mathbb{D}(\pi_E, \pi)$$

where $\mathbb{D}$ is a similarity measure between policies [4].

2.1.2 Feature Representations

Before learning a policy, it is important to represent the observable state in a form that is adequate and efficient for training. This representation is a ***feature vector*** that must contain information about the expert and his surroundings.

Recent feature extraction techniques based on ***Deep Leaning*** approaches automatically process raw data to extract features used for learning. These have the advantage of minimizing the task specific knowledge required for training agents.

2.2 IMITATION LEARNING FOR AUTONOMOUS VEHICLES

In the following section Imitation Learning will be contextualized for AVs to determine an End-To-End driving system, the purpose of this thesis.

In the case of autonomous driving, the expert is a human driver or a programmed software, and the actions a_i^t are the driving commands, e.g. steering, acceleration and braking. States s_i^t are observation about expert and environment coming from sensors. The main sensing modalities are shown in Figure 3. So, the system is optimized to produce the same driving actions, based on the sensory input recorded while the expert was driving.

Actually, ***Behavior Cloning (BC)*** is the dominant paradigm used in End-To-End driving. This is described in the following section.

2.2.1 BEHAVIOR CLONING

Behavior Cloning formulates the problem as a ***Supervised Learning*** process with the objective of matching the learned policy π_θ to the expert policy π_E:

$$min_\theta \mathbb{E} \| \pi_\theta - \pi_E \|$$

which is typically optimized by minimizing L1 or L2-loss.

2.2.2 LIMITATIONS AND SOLUTIONS

Imitation Learning suffers from some problems that are closely linked to the highly dynamic nature of the problem of autonomous driving [5]. The main ones will be described below, suggesting possible solutions.

Distribution shift problem: when a car is driven by the model, the model's output can influence the car's observation in next time step. If the driving decision led to unseen situation, the model may not know how to behave. When the expert collects reference data, rarely demonstrates ***recovery behaviour*** from potential mistakes or less common and frequent situations. The solution to these problems is the extension of the data that can be obtained through ***data augmentation***, ***data diversification*** and ***on-policy learning***.

- ***Data augmentation*** is essential when collecting large amounts of data is challenging. Therefore, new data can be generated artificially through transformations. Typical practices could be adding *blurring*, or *noise*, *cropping* or *changing in brightness*. Sometimes it may be necessary to associate the transformations of the input state with corresponding changes to the action, generating coherent state-action pairs. This is the case of adding lateral cameras, rotated by a certain angle, in order to simulate situations of recovery from errors.
- ***Data diversification*** is the practice of diversifying training data. An example is that of adding noise to the commands executed by the agent, to force it into situations that otherwise could not be observed, and then record the corresponding recovery actions.
- ***On-policy learning*** consists into alternate the trained model and expert during data collection. What happens is that the expert provides examples of how to recover from error situations that the trained model could lead to. These annotations can be obtained offline or online by ignoring the output provided by the model when it leads to errors. In this way, the dataset is enriched with data for recovery situations. At this point, the model needs to be retrained with the entire dataset.

Dataset Biased: biases in data can reduce the performance or generalization ability of the model at testing time. This is because driving data typically consists in either a few simple behaviours or a heavy tail of complex reactions to rare events and is rich in spurious correlations that can lead to ***Causal Confusion***. Optimize for average imitation accuracy, the errors on rare cases would tend to be smeared, preventing the model from improving. Therefore, a huge amount of data can lead to poor generalization performance. In some circumstances, such as lack of steering angle diversity or in case of architectures that incorporate high-level commands, a balancing or weighing of the data may be performed.

Causal confusion: given the lack of explainability of the observations, it leads to the inability to distinguish spurious correlations from the true causes of the actions associated with the observations in the training set. An example of this is represented by the ***Inertia problem*** in which if the ego vehicle is stationary, the probability that it remains in this state in the training data is high. This leads to a false correlation between low speed and no acceleration, causing the vehicle to stop excessively and difficult to restart.

Training instability: as for other problems, using ANNs it is not guaranteed to converge to same minima each time. Furthermore, Imitation Learning models can still be prone to overfitting. One way to make the convergence more stable is the initialization of the weights in a non-random way, as shown in [5], in which starting from a model pre-trained on a different task, a reduction of variance is achieved.

Chapter 3.

STATE OF THE ART

3.1 CARLA: AN OPEN DRIVING SIMULATOR

Creating an autonomous agent capable of navigating urban scenarios characterized by a high degree of dynamism, due to the presence of other agents and traffic rules, remains a challenge. But, in addition to the complexity of the driving task, research on autonomous driving is hampered by the high costs and challenges of real-world training and testing. Large amounts of data are required to train an agent and using a single vehicle would not be sufficient to collect the amount of data to cover the multitude of environmental conditions and situations that could occur. This would take an extremely long time. It would therefore be necessary to use several vehicles equipped with the same sensors, inevitably multiplying the costs. To enrich the data, unsafe situations or human errors must be simulated, providing the corresponding recovery commands. Finally, the trained agent must be validated in the physical environment.

Training and testing an agent in a real-world environment go beyond the limits for many research groups.

The alternative is to acquire data and validate in a simulated environment. This allows to easily collect labelled datasets and validate systems, contemplating those unsafe scenarios to stage in the physical world. Furthermore, a simulator can bring advantages in terms of execution times, repeatability, and parallelization of executions.

CARLA (Car Learning to Act) [6] is an open simulator for urban driving. It was developed to support the training and validation of autonomous driving models, including perception and control. It includes urban layouts, a multitude of models of vehicles, buildings, pedestrians, road signs, etc. The simulation platform supports the flexible configuration of sensors and provides different signals that can be used to train driving strategies, such as GPS coordinates, speed, acceleration and detailed data on collisions and other infringements. A wide range of environmental conditions can be specified, including weather conditions and time of day Figure 5. Two maps are provided in Figure 6, ***Town 1*** and ***Town 2***. The first, is a basic urban environment consisting of *T-intersections*, while the second is similar to the first but smaller.

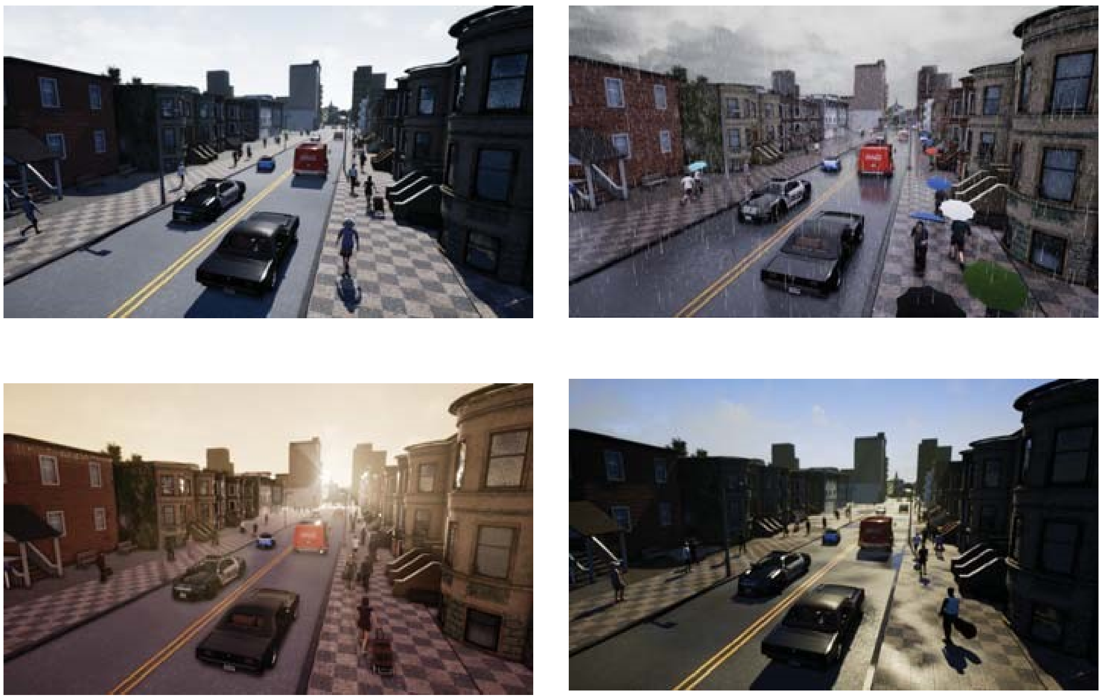

Figure 5: Some examples of weather conditions and time of day.

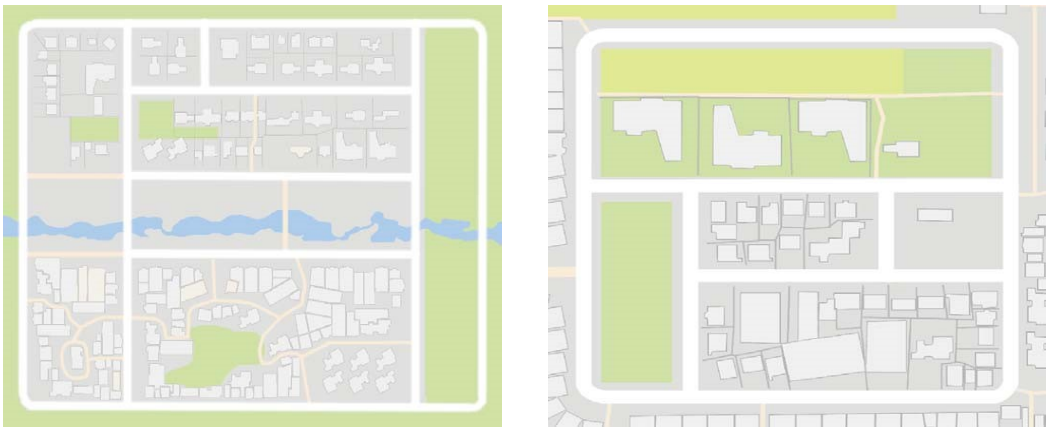

Figure 6: On the left the Town 1 map, while on the right the Town 2 map.

Today, CARLA is mainly used to build different ***benchmarks*** to evaluate driving performance, staging navigation scenarios with increasing difficulty, manipulating the complexity of the route, the presence of traffic and environmental conditions. This was used to evaluate the main solutions of the scientific literature, such as *CILRS* and *Learning by cheating*.

Obviously, migration policies from the simulated to the physical environment are required. The transfer can be carried out considering Source-Independent inputs, such as Semantic Segmentation, obtaining a common representation in terms of features both for the simulated and physical environment, or by realizing Transfer Learning, through fine-tuning on new datasets.

3.2 Dataset CARLA100

To train an Imitation Learning agent offline, it is necessary to have a dataset made up of ***state-action*** pairs. The states are the observation about expert and environment in terms of data read through the sensors, while the actions are the response of the expert in correspondence of these states.

3.2.1 EXPERT DEMONSTRATOR

CARLA100 [5] was collected by an expert in the CARLA simulator. The route followed by the expert, from the origin to the destination, are calculated using a planner based on the ***A* algorithm***, while the steering, throttle and brake commands are generated using a ***PID controller***. The expert drives at a speed of 35 km/h when go straight, reducing it to 15 km/h when turn. In addition, the expert is programmed to react to other agents, such as pedestrians and vehicles, reducing its speed proportionally to the collision distance, or in case of lead car.

To improve diversity, realism, and increase the possible observed situations, ***noise*** is added to the ego car controls according to the protocol described in [7]. The noise simulates a gradual drift away from the desired trajectory of the expert. During a training the drift is not used, but only the reaction performed by the expert.

3.2.2 CONTENT

The dataset collection is divided into goal directed episodes, where the expert goes from a start position to a goal position. In total 2373 episodes are collected, on Town 1. Each episode has the following features:

- ***Number of Pedestrians***: the total number of spawned pedestrians on the town. This number is randomly sampled from the interval [50,100].
- ***Number of Vehicles***: the total number of spawned vehicles on the town. This number is randomly sampled from the interval [30,70].

- ***Spawned seed for pedestrians and vehicles***: the random seed used for the CARLA object spawning process.
- ***Weather***: the used weather for the episode which is sampled from the set: ***Clear Noon, Clear Noon After Rain, Heavy Rain Noon, Clear Sunset***.

Each episode is partitioned in simulation steps of 100 ms. For each step, stored data are divided into two different categories, sensor data and measurement data.

Figure 7 shown sensors data acquisitions from different camera sensors.

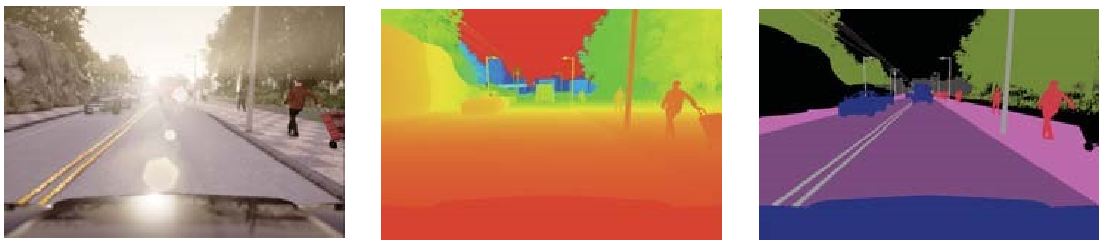

Figure 7: Some examples of data from sensors. On the left the acquisitions from the RGB camera, in the center those from the Depth camera, on the right the Semantic Segmentation.

For each sensor, data are acquired with three different orientations: aligned with the car centre, rotated 30 degrees to the left and rotated 30 degrees to the right, to simulate recovery situations during training.

Measurement's data are acquired about the state of the ego-vehicle, the world and other non-player agents. From the ego-vehicle and the world status the following data are collected:

- ***Step Number***: the step number of the current step, starts at zero and is incremented by one for every simulation step.
- ***Game Time-stamp***: the time that has passed since the simulation has started.
- ***Position***: the world position of the ego-vehicle. It is expressed as a three-dimensional vector $[x, y, z]$ in meters.

- ***Orientation***: the orientation of the vehicle with respect to the world. Expressed as Euler Angles (row, pitch and yaw).
- ***Acceleration***: the acceleration vector of the ego-vehicle with respect to the world.
- ***Forward Speed***: a scalar expressing the linear forward speed of the vehicle.
- ***Intentions***: a signal that is proportional to the effect that the dynamic objects on the scene are having on the ego car actions. Three different intention signals are used: stopping for pedestrians, stopping for cars and stopping for traffic lights.
- ***High Level Commands***: the high-level indication stating what the ego-vehicle would do on the next intersection: go straight, turn left, turn right, or do not care (the ego vehicle could pick any option). Each of these commands are encoded as an integer number. 2 is do not care, 3 for turn left, 4 for turn right, 5 for go straight.
- ***Waypoints***: a set containing the 10 future positions the vehicle would assume.
- ***Steering Angle***: the current steering angle of the vehicle's steering wheel.
- ***Throttle***: the current pressure on the throttle pedal.
- ***Brake***: the current pressure on the brake pedal.
- ***Hand Brake***: if the hand brake is activated.
- ***Steer Noise***: the current steering angle in the vehicle considering the noise function.
- ***Throttle Noise***: the current pressure on the throttle pedal considering the noise function.
- ***Brake Noise***: the current pressure on the brake pedal considering the noise function.

For each of the non-player agents (pedestrians, vehicles, traffic light), the following information are provided:

- ***Unique ID***: a unique identifier of this agent.
- ***Type***: if it is a pedestrian, a vehicle or a traffic light.
- ***Position***: the world position of the agent. It is expressed as a three-dimensional vector $[x, y, z]$ in meters.
- ***Orientation***: the orientation of the agent with respect to the world. Expressed as Euler angles (row, pitch and yaw).
- ***Forward Speed***: a scalar expressing the linear forward speed of the agent.
- ***State***: only for traffic lights, contains the state of the traffic light, it its either red, yellow or green.

3.3 EVALUATION: METHODS AND METRICS

Camera-based autonomous driving can be viewed as a ***Computer Vision Problem***. It requires analysing the input video stream and estimating certain high-level quantities, such as the control signal to be executed. A standard methodology in computer vision is to evaluate an algorithm by collecting a dataset with ground-truth annotation and evaluating the results produced by the algorithm against this ground truth by calculating a certain loss metric. End-to-end methods are commonly evaluated by collecting a large dataset of expert driving and measuring the average prediction error of the model on the dataset. However, driving, in contrast with most computer vision tasks, is inherently ***active***. That is, it involves interaction with the world and other agents. Thus, this O***ffline*** or O***pen-loop*** assessment may not be representative of the O***nline*** or C***losed-loop*** real driving performance [8]. Therefore, the two evaluation processes are not exclusive, but they are both necessary to build a model capable of interacting with the environment. Figure 8 shows the two evaluation processes described.

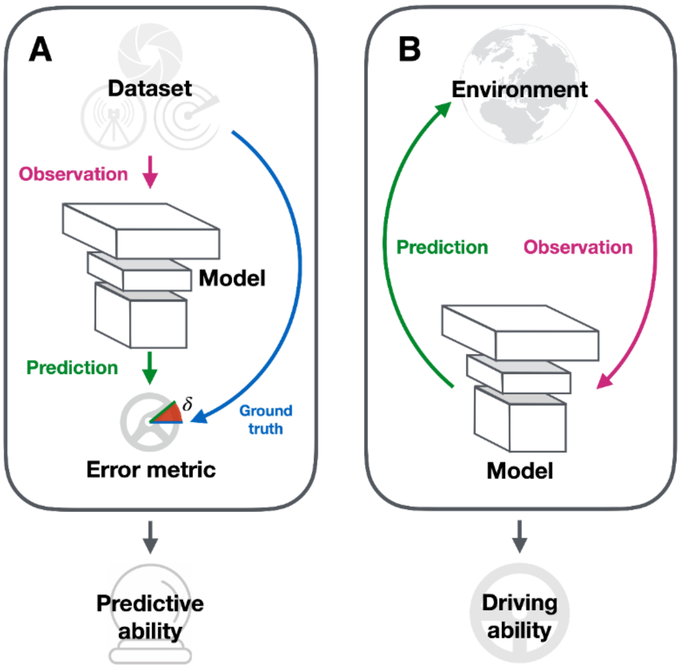

Figure 8: On the left the Open-loop evaluation (A), on the right the Closed-loop evaluation (B).

What is discovered in [8], is that offline performance can be poorly correlated with online driving: a network with a very low prediction error can be extremely negative during online driving, on the other hand, a model with high prediction errors, can achieve surprising driving performance.

There are offline metrics that correlate better with online performance and then should preferred during training. For example***, Mean Absolute Error (MAE)*** is shown to be advantageous to ***Mean Squared Error (MSE)*** in that sense. Balanced metrics such as ***Balanced L1 Loss*** correlating even stronger with driving ability [5] [9].

3.4 BENCHMARKS

Online or ***Closed-loop*** evaluation of a driving agents involves deploying a vehicle or a fleet of vehicles in the real world, on which the model is run. This leads to a series of issues from an economic, logistical and safety point of view, already discussed in 3.

The recent CARLA driving simulator allows evaluating driving policies in living towns, populated with vehicles and pedestrians, under different weather and illumination conditions. For this purpose two benchmarks have been developed ***CoRL2017*** [6] and ***NoCrash*** [5], to evaluate and compare different kind of autonomous driving algorithms.

3.4.1 CORL2017 BENCHMARK

CoRL2017 Benchmark [6] is designed to evaluate some basic capabilities of autonomous driving system by using the increasingly difficult tasks within the CARLA Environment. The tasks are set up as goal-directed navigation: an agent is initialized somewhere in town and must reach a destination point. In these experiments, the agent is allowed to ignore speed limits and traffic lights. Tasks are organized in order of increasing difficulty:

- ***Straight***: destination is straight ahead of the starting point, and there are no dynamic objects in the environment. Average driving distance to the goal is 200 m in Town 1 and 100 m in Town 2.
- ***One turn***: destination is one turn away from the starting point; no dynamic objects. Average driving distance to the goal is 400 m in Town 1 and 170 m in Town 2.

- ***Navigation***: no restriction on the location of the destination points relative to the starting point, no dynamic objects. Average driving distance to the goal is 770 m in Town 1 and 360 m in Town 2.
- ***Navigation with dynamic obstacles***: same as the previous task, but with dynamic objects (vehicles and pedestrians).

Experiments are conducted in two towns: Town 1 (seen during training), Town 2 (not seen during training). Six weather conditions are considered organized in two groups. The Training Weather Set, that is the same used during acquisitions of CARLA100 dataset and includes clear day, clear sunset, daytime rain, and daytime after rain. The Test Weather Set is never seen during training and includes cloudy daytime and soft rain at sunset.

For each combination of a task, a town, and a weather set, the testing is carried out over 25 episodes. An episode is considered successful if the agent reaches the goal position within a time budget. The time budget is set to the time needed to reach the goal along the optimal pat at a speed of 10 km/h. Infractions, such as driving on the sidewalk or collisions, do not lead to termination of an episode, but are logged and reported.

3.4.2 NoCrash Benchmark

NoCrash Benchmark [5] focused on testing the ability of the ego vehicles to handle complex events caused by changing traffic conditions (e.g., traffic lights) and dynamic agents present in the scene. To do this, three different tasks are proposed, but also different metrics compared to CoRL2017. Each task corresponding to completing 25 goal directed episodes, that have the same set of start and end positions, as well as an increasing level of difficulty. The tasks are the follow:

- ***Empty Town***: no dynamic objects.
- ***Regular Traffic***: mild number of cars and pedestrians.
- ***Dense Traffic***: large number of pedestrians and heavy traffic.

Like CoRL2017 benchmark, NoCrash has six different weather conditions, where four are seen in training and two are reserved for testing. It also uses Town 1 and Town 2, the first used during training, the second for testing.

The evaluation protocol states that an episode is a failure when any collision occurs. On the other hand, it is considered successful if the agent reaches the goal in a time limit, without any collision. It is also considered the ability of the agent to obey traffic rules such as traffic light. Note that an episode is not terminated when a traffic light violation occurs.

3.5 CONDITIONAL IMITATION LEARNING

A limitation for Imitation Learning is the assumption that actions can be inferred from the perception of the environment alone. In practice this assumption is not verified. For example, observations alone are not enough to predict the direction to take at intersections.

Conditional Imitation Learning (CIL) [5] [7] [9] [8] [10], tries to answer this problem. The idea is to provide the model with not only the perception of the environment, but also a representation of the expert's intention corresponding to a ***high-level command***, such as turn left at next intersection. So that the model is freed from planning, and at testing time can be controlled by a human passenger or a topological planner.

3.5.1 Formulation

Consider a generic controller that interact with the environment over a discrete time. In each time step t, an observation o_t is acquired from environment and given to the controller that outputs an action a_t. As mentioned, Imitation Learning is a supervised approach in which a policy is trained offline to mimic an expert. The training data are ***observation-action*** pairs and are collected in a dataset, $\mathcal{D} = \{\langle o_i,\ a_i \rangle\}_{i=1}^{N}$. The function approximator F is parametrized with θ and must be optimized to fit the mapping between observation and actions:

$$minimize_\theta \sum_i \ell(F(o_i,\ \theta), a_i)$$

The assumption is that the expert's actions are fully explained by observation and, therefore, that there exists a function E that maps observation to expert's actions: $a_i = E(o_i)$. If this assumption holds, a sufficiently expressive approximator will be able to fit the function E given enough data.

Is this true anyway? The answer is no. The assumption of the existence of the E function that maps the observations to the actions is true in the case of simple tasks such as the follow lane but fails in the case of more complex scenarios. To explain this, consider the vehicle approaching an intersection. The subsequent actions of the driver are not explained by the observations but are additionally affected by the driver's internal state, that is, on his intentions about the destination to be reached. The same observations could lead to different actions, depending on this intention. Without this information, the vehicle could navigate the environment, making arbitrary decisions at intersections.

To answer this, the internal state of the driver or expert is modelled with a vector h which, together with the observations, allows the actions of the expert to be explained: $a_i = E(o_i\ ,\ h_i)$. Vector h can contain a lot of information, from destination to previous knowledge. To expose the latent state to the controller, it is mapped through a function c on a further input command to the controller, $c = c(h)$. At training time, this is provided by the expert used to acquire the data, while at testing time, the command can come either from the human driver or from a planner.

The addition of a further input brings about a consequent change in the dataset which becomes $\mathcal{D} = \{\langle o_i,\ c_i,\ a_i\rangle\}_{i=1}^{N}$, and it is possible to formulate Conditional Imitation Learning as follows:

$$minimize_\theta \sum_i \ell(F(o_i, c_i, \theta), a_i)$$

The setting of Conditional Imitation Learning is shown in Figure 9.

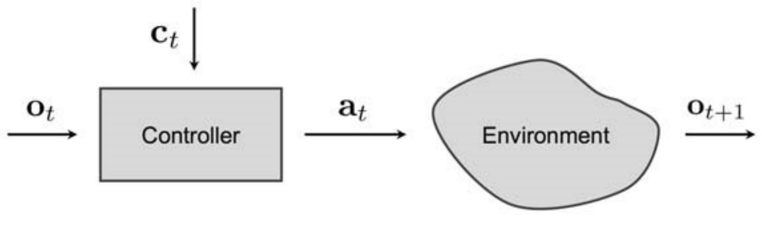

Figure 9: Setting of Conditional Imitation Learning

3.5.2 NETWORK ARCHITECTURE

The controller F can be seen as a ***Deep Neural Network*** that takes as input the observations $o = \langle i, m\rangle$ consisting of images and measurements, and the command c, and produces an action a in terms of actuation signals, steering, throttle, and brake.

The images are processed through the perception module $I(i)$, while measurements are processed through the measurement module $M(m)$. The first is implemented as a convolutional network, while the second as a fully connected network The outputs of these modules are concatenated to form a join representation, given by:

$$J(i,m) = \langle I(i), M(m) \rangle$$

For commands we can assume a discrete set $C = \{c^0, \dots, c^k\}$ and a specialist branch A^i. All these branches are fully connected network and share the join representation. The command acts as selector for the branch that compute the action for the current observation. The output of the network can be written as:

$$F(i, m, c^i) = A^i(J(i,m))$$

In a common driving scenario, the considered c are $\{turn\ left, turn\ right, go\ straight,\ continue\}$, which represent what needs to be done at the next intersection. The network architecture described is shown in Figure 10.

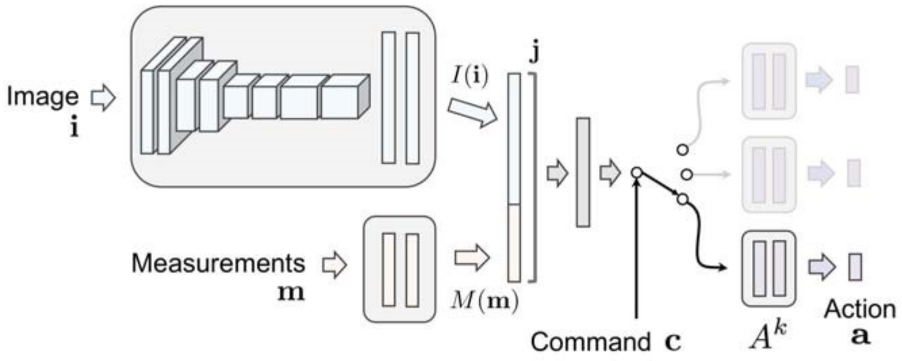

Figure 10: Conditional Imitation Learning architecture.

3.5.3 Network Details

For module I the input i is the currently observed image at $200x88 - pixels$ resolution, while for module M, the input m is the current speed of the car.

In the base CIL model, presented in [7], the I module consists of 8 convolutional and 2 fully connected layers. The convolution kernel size is 5 in the first layer and 3 in the following layers. The first, third, and fifth convolutional layers have a stride of 2. The number of channels increase from 32 in the first convolutional layer to 256 in the last. Fully-connected layers contain 512 units each. The M module is implemented using standard MLP. The ReLU activation function is used after all hidden layers, a Batch Normalization is performed after convolutional layers, a 50% Dropout is applied after fully-connected hidden layers and a 20% Dropout is used after convolutional layers. For last layers, can be considered a Linear activation function.

3.5.4 Loss Function

Actions are two-dimensional vectors, composed by steering and throttle signals, expressible as $a = \langle s, t \rangle$. So, given the predicted action and the ground truth actions a_{gt}, the per-sample loss is determined as:

$$\ell(a, a_{gt}) = \|s - s_{gt}\|^2 + \lambda_a \|a - a_{gt}\|^2$$

3.5.5 Limitations

In [5], some limitations for the CIL model have been highlighted. The ***Distributional Shift Problem*** between the training and testing distributions is the main limitation of this approach, which might require on-policy data collection, obtained by the learning agent. Additionally, datasets are biased because most real-world driving is dominated by common behaviours. In relation to ***Dataset Bias***, CIL can suffer from ***Causal Confusion***: spurious correlations cannot be distinguished from true causes in the observed training demonstration. A typical failure is due to ***Inertia Problem***, where if the ego-vehicle is stopped, the likelihood that it will remain stationary is predominant in training data.

Finally, CIL suffers from ***High Variance***. Once the training set is fixed, the model is expected to always learn the same policy in different training runs. This does not happen since the loss function is optimized through the ***Stochastic Gradient Descent (SGD)***, which assumes that the samples are i.i.d. When a policy is trained on long-term expert demonstrations, the i.i.d hypothesis does not hold. Consequently, it is possible to observe a high sensitivity to the different initializations and to the order in which the samples are presented in input during training.

3.6 CILRS

To address the limitations of the CIL model, some improvements have been proposed in [5]. The robustified model obtained was defined as ***CILRS***, where the ***R*** and ***S*** are due to two modifications for the architecture shown in Figure 11.

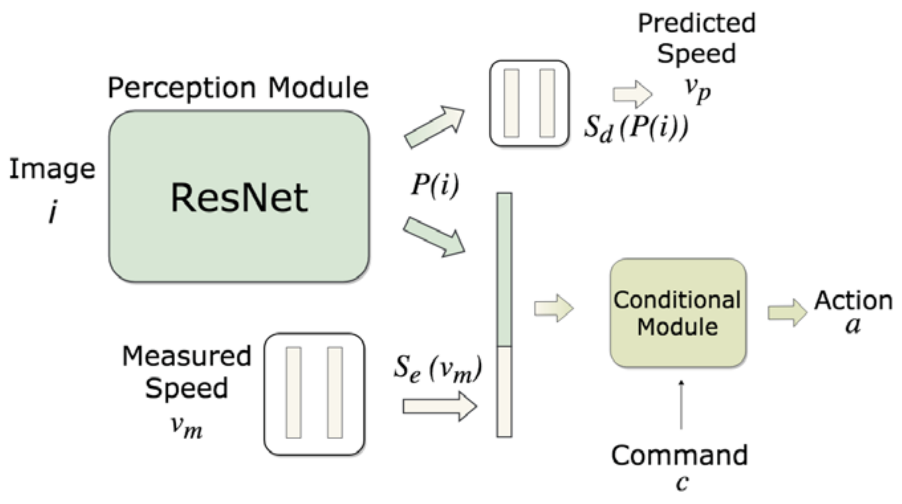

Figure 11: CILRS architecture, with ResNet perception backbone and speed prediction.

3.6.1 Deeper Perception Backbone

For perception backbone, a ***ResNet34*** architecture is used. In the presence of large amounts of data, using deeper architectures can be an effective strategy to improve performance. These can lead to a reduction in variance of training. Moreover, bigger models can have a better generalization performance in complex urban environments.

3.6.2 Weights Initialization

It is shown that using ***ImageNet*** weights initialization for perception backbone leads to a reduction in variability during training, mainly due to less randomness and a more stable policy learned. Thus, it is possible to benefit from generic transfer learning.

3.6.3 Speed Prediction

To overcome the ***Inertia Problem***, the architecture is enriched with an additional branch S specialized in predicting the current speed of the car. This branch shares the join representation obtained from the modules of perception I and measurements M, obtaining:

$$F(i, m, c^i) = (A^i(J(i, m)), S(J(i, m)))$$

What happens is that this joint optimization imposes the extraction of ***speed-related features*** making sure that the input speed is not the only way to obtain the dynamics of the scene, exploiting instead visual signals that are predictive of the speed of the car (e.g. free space, curves, traffic light states, etc.).

Speed prediction is used only during training.

3.6.4 MULTIMODAL INPUT

For perception, instead of using just RGB images, you can consider adding more information, such as depth information. In [9], some input schemes have been proposed. The most effective was ***Early Fusion***, which consists in extending the RGB channels by adding the Depth channel and in putting this chained information in input to the model, instead of using two different perception backbones.

3.6.5 L1 LOSS

As described in [5] and [9], ***L1 loss*** function is used instead of ***Mean Squared Error (MSE)*** adopted for CIL model, as it is more correlated to online driving performance.

Given the predicted action a, its ground truth a^{gt} and a vector of weights w the loss is defined as

$$\ell_a(a, a^{gt}, w) = \sum_i^n |w_i(a_i - a_i^{gt})|$$

with $n = 3$ (steering angle, throttle, brake). The action is computed by the active branch selected by the command c. This is contemplated in the following explanation in the loss $\ell_a(a, a^{gt}, w; c)$.

For speed prediction the error between the predicted speed m and the ground truth speed m^{gt} is computed as:

$$\ell_m(m, m^{gt}) = |m - m^{gt}|$$

To incorporate both errors, a factor β is used to weigh their relevance in the overall loss:

$$\ell(a, a^{gt}, w; c;\ m, m^{gt}) = \beta\, \ell_a(a, a^{gt}, w; c) + (1 - \beta)\, \ell_m(m, m^{gt})$$

3.6.6 EXPERIMENTAL DETAILS

For research on Conditional Imitation Learning, different training procedures are used. We can refer to those that led to relevant results in [5] and [9].

For the first at each iteration, a minibatch is sampled randomly from a dataset of 10 hours and presented to the network for training. If the training error hasn't decreased for over 1000 iterations the learning rate is divided by 10. 2 hours of validation dataset are used to discover when to stop the training process. A validation is performed after $20K$ iterations and if the validation error increases for three iterations, this checkpoint is used to test on benchmarks, both ***CoRL2017*** and ***NoCrash***.

In the second, a 25-hour dataset is used to train the CILRS model. In addition to the frontal camera, lateral cameras are adopted to simulate recovery episodes from error situations. The RGB images are enriched with depth information, according to the ***Early Fusion*** technique, explained in 3.6.4.

In the experiments, it is shown that photometric and geometric data augmentation techniques do not lead benefits, therefore they are not used. However, the data are ***balanced***: this concerns both weather conditions and high-level commands contained within the training minibatches.

The Loss function used is L1, for which the weight vector $w = (0.5, 0.45, 0.05)$, while the actions is balanced with respect to the speed by a $\beta = 0.95$ factor.

Driving performance are assessed through ***CoRL2017 Benchmark***. To select the best model to test a specified protocol is developed: a V_p validation performance is monitored every $500\,K$ iterations. V_p is defined to balance training and testing conditions, both in terms of maps and weathers. In particular, $V_p = 0.25\,V_w + 0.25\,V_t + 0.50\,V_{wt}$ where V_w is the success rate when validating in Town 1 and 'soft rainy sunset' weather (not included in training data), V_t is the success rate when validating in Town 2 (not included in training data) and 'clear noon' weather (included in training data), and V_{wt} stands for success rate when validating in Town 2 and 'soft rainy sunset' (neither town, nor weather are part of the training data). Therefore, note that V_p is a weighted success rate based on 75 episodes.

3.7 LEARNING BY CHEATING

Currently the ***State of the Art*** is achieved by this learning approach [11]. This assumes that the direct Imitation Learning from expert trajectories to vision-based driving is hard, so the learning process can be decomposed into ***two stages***.

Conceptually, direct sensorimotor learning conflates two difficult tasks: ***learning to see*** and ***learning to act***. The described procedure tackles these in turn. For the privileged agent, in the first stage, perception is solved by providing direct access to the environment's state, and the agent can thus focus on learning to act. In the second stage, the privileged agent acts as a teacher, and provides abundant supervision to the sensorimotor student, whose primary purpose is to learn to see. This is illustrated in Figure 12.

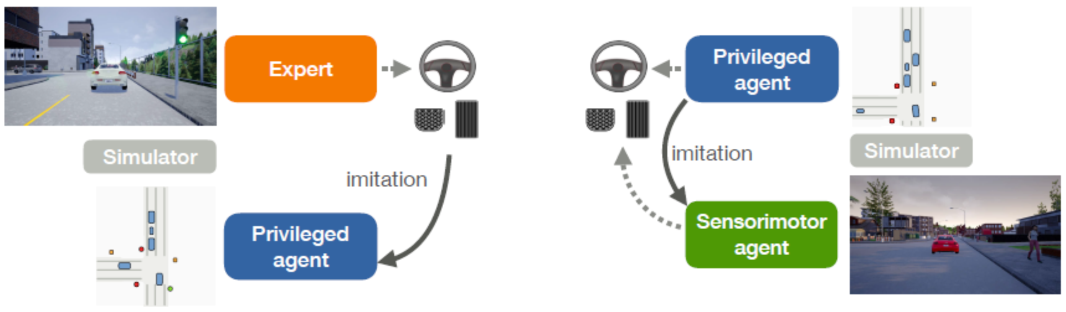

Figure 12: Learning by cheating process. (Left) An agent with access to privileged information learns to imitate expert demonstration. (Right) A sensorimotor agent vision-based without access to privileged information learns to imitate the privileged agent.

Concretely, the decomposition provides three advantages. First, the privileged agent operates on a compact intermediate representation of the environment *(a bird's-eye view)* and can thus learn faster and generalize better. Second, the trained privileged agent can provide much stronger supervision than the original expert trajectories. This enables automatic DAgger-like training in which supervision from the privileged agent is gathered adaptively via online rollouts of the sensorimotor agent [12]. The third advantage is that the privileged agent produced in the first stage is a ***white box***: if the privileged agent is trained via Conditional Imitation Learning, it can provide an action for each possible command (e.g., "turn left", "turn right") in the second stage. Thus, all conditional branches of the privileged agent can train all branches of the sensorimotor agent in parallel. The architecture is illustrated in Figure 13. The privileged agent receives a *bird's-eye view image* and produces a set of *heatmaps* that go through a *soft-argmax layer (SA)*, yielding waypoints for all commands. The input command selects one conditional branch. The corresponding waypoints are given to a *low-level controller* that outputs the steering, throttle, and brake. The sensorimotor agent receives image from a forward-facing camera and produces waypoints in that are selected based on the command and passed to the low-level controller.

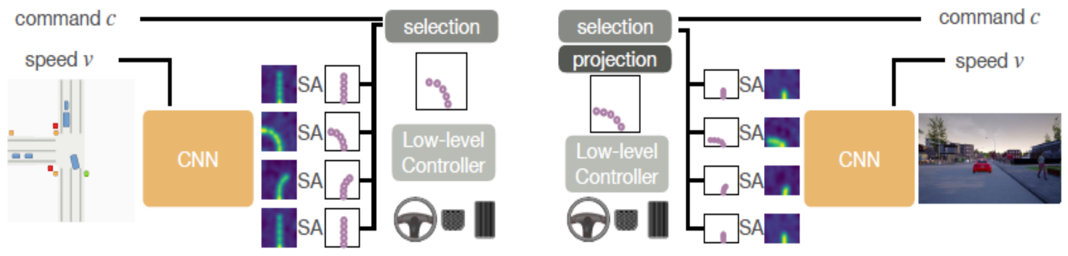

Figure 13: Learning by cheating architecture.

3.8 A comparison between State of the Art

Here are the results for the main approaches analyzed, with Learning by cheating, currently the ***State of the Art*** [11].

TASK	SCENARIO	CIL	CILRS	LBC
STRAIGHT	**Training**	98	96	100
ONE TURN		89	92	100
NAVIGATION		86	95	100
NAV. DYNAMIC		83	92	100
STRAIGHT	**Test**	98	96	100
ONE TURN		90	96	100
NAVIGATION		84	96	100
NAV. DYNAMIC		82	96	96

Figure 14: Results on CoRL2017 Benchmark for CIL, CILRS and Learning by cheating. These results are based on CARLA 0.9.x.

TASK	SCENARIO	CIL	CILRS	LBC
EMPTY	**Training**	79 ± 1	87 ± 1	100 ± 0
REGULAR		60 ± 1	83 ± 0	99 ± 1
DENSE		21 ± 2	42 ± 2	95 ± 2
EMPTY	**Test**	83 ± 2	87 ± 1	100 ± 0
REGULAR		55 ± 5	88 ± 2	99 ± 1
DENSE		13 ± 4	70 ± 3	97 ± 2

Figure 15: Results on NoCrash Benchmark for CIL, CILRS and Learning by cheating. These results are based on CARLA 0.9.x.

This thesis focuses on one-stage learning, therefore the obtained results will be compared with those of the CILRS [5] ***architecture, since Learning by cheating*** [11] ***uses a two-stage learning approach.***

3.9 TECHNOLOGIES

In this paragraph we are going to discuss about the technologies used during this work, from both a *software* and *hardware* point of view.

3.9.1 CARLA SIMULATOR

It has already been introduced in section 3.1. Currently different versions are available from 0.8.1 up to 0.9.12. For this work CARLA version 0.8.4 will be used to evaluate the performance of driving models in an urban environment. The choice was guided by the fact that it is a stable version and many scientific works including those presented in [10] are made using this.

3.9.2 TENSORFLOW

TensorFlow [13] is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is developed by the Google Brain team. Google release the updated version of TensorFlow, named TensorFlow 2.0 . This is used to build our network architectures. In particular, the CILRS [5] architecture is developed using the ***PyTorch*** framework, and a platform conversion is carried out.

3.9.3 NVIDIA TITAN XP

Figure 16: NVIDIA TITAN Xp

To run the artificial intelligence algorithms, we use the ***NVIDIA TITAN Xp***. This is shown in Figure 16. This is equipped with a *Pascal* architecture with 3840 *CUDA Cores*, a *Frame Buffer* of 12 GB *GDDR5X*, a memory speed of 11.4 $Gbps$.

Chapter 4.

PERSONAL CONTRIBUTION TO THE SOLUTION OF THE PROBLEM

In this chapter the developed solution will be presented, starting from the CILRS architecture, CARLA100 dataset and CARLA Simulator. Furthermore, the protocols followed to train and evaluate the candidate solutions will be introduced.

4.1 NETWORK ARCHITECTURE

4.1.1 PERCEPTION BACKBONE

Starting from the ***CILRS*** model presented in [5], we compare the same backbone ***ResNet34*** and the deeper ***ResNet50V2***. As indicated in the same study, a deeper network can help reduce variance in training, as well as improve generalization capabilities in complex dynamic scenarios. *Although, as noted in* [10], *for a deeper network like there is still a chance of overfitting the training data, resulting in worse results.*

ResNetV1 and ResNetV2

The idea behind ***ResNet*** models is that increasing the depth of a network cannot be done simply by stacking the layers one behind the other. Deep networks are difficult to train due to the ***Gradient Vanishing Problem***, which can make the gradient infinitely small. As a result, as the network deepens, its performance becomes saturated or even begins to degrade rapidly.

With ***ResNet***, so-called ***"identity shortcut connections"*** were introduced that skip one or more layers, as in Figure 17. The authors of [14] [15] argue that stacking layers shouldn't degrade network performance, since we could simply stack the mappings of the network ***identity*** (layer that does nothing) on the current network and the resulting architecture would work the same way. This indicates that the deeper model should not produce a higher training error than its shallower counterparts. They speculate that letting stacked layers fit into a residual mapping is easier than letting them fit directly into the underlying desired mapping. The ***Residual Block*** in Figure 17 explicitly allows it to do just that.

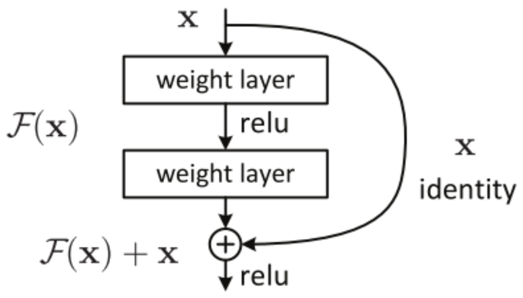
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 17: A Residual Block.

Figure 18 shows the differences between ***ResNetV1*** and ***ResNetV2***. While ResNetV1 adds the second non-linearity after the addition operation is performed between x and $F(x)$, ResNetV2 has removed the last non-linearity, therefore, clearing the path of the input to output in the form of identity connection. Furthermore, ResNetV2 applies Batch Normalization

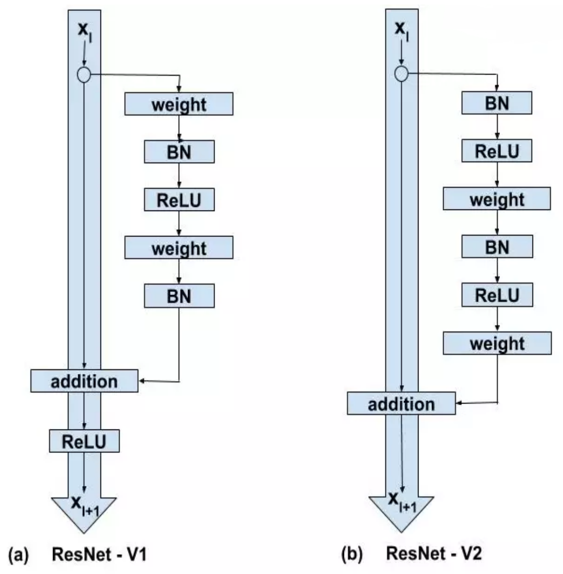

Figure 18: ResNetV1 and ResNetV2

and ReLU activation to the input before the convolution operation, while ResNetV1 performs the convolution followed by Batch Normalization and ReLU activation.

ResNet has many variants that run on the same concept but have a different number of levels such as ***Resnet34*** the variant that can work with 34 layers of neural network and ***Resnet50*** with 50 layers.

4.1.2 ACTIVATION FUNCTIONS

As already indicated, the model's output consists of the steering, throttle and brake actuation signals. Except for the steering which can assume values in the continuous range $[-1, 1]$, for throttle and brake the reference range is $[1, 0]$. This is a ***regression problem***, in which values must be limited to a specific range.

A typical activation function used on the last layer of an ANN in the case of regression is the ***Linear*** one. This is the case of the CILRS model, in which this function is used for all output layers. The outputs, therefore, can take an arbitrary value in an unbounded continuous range, from negative to positive. If we wanted to incorporate a priori knowledge about data into the construction of the model, we know that the steering, throttle and brake values are limited and, therefore, negative values will always be wrong. Furthermore, at testing time, potentially, the model can output negative values and so, to drive the vehicle, some manipulations are required, to ensure that these values fall within the specific range. This is not an efficient practice.

To address the problem the practical solution is to limit the outputs of the model. We proceeded by making some changes to the architecture:

- for ***steering*** outputs, the ***Hyperbolic Tangent*** activation function is added, to limit the values in the range $[-1, 1]$.
- for ***throttle*** outputs, ***ReLU*** activation function is added to limit values to positive range.
- for ***brake*** outputs, ***ReLU*** activation function is added to limit values to positive range.

Hyperbolic Tangent:

$$tanh\,(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

ReLU:

$$f(x) = \max\,(0, x)$$

These are shown in Figure 19.

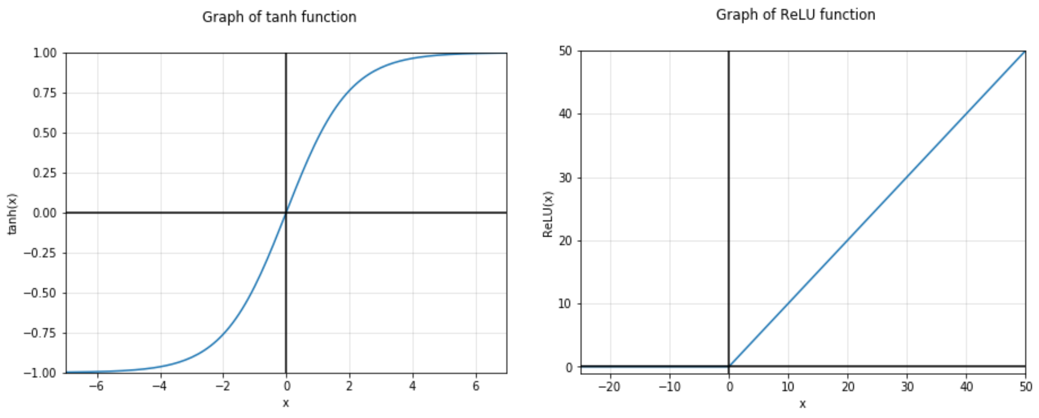

Figure 19: On the left the Hyperbolic Tangent activation function, on the right the ReLU.

The ***ReLU (Rectified Linear Unit)*** is preferred over the ***Sigmoid*** activation function. The latter suffers from ***Gradient Vanishing Problem***, that is, when the function saturates close to 0 or 1, the value of the gradient goes close to 0.

Furthermore, ***ReLU*** is computationally more efficient and tend to show better convergence performance than ***Sigmoid***.

Experiment ResNet50V2 2 which uses this architecture, showed a stuck vehicle, unable to move from the initial position, i.e. with zero accelerator and brake values. This means that, again, the model did not come up with a correct policy from the data. This pointed to the ***Dying ReLU*** problem as a potential cause: ***ReLU*** neurons can sometimes be pushed into states in which they become inactive for essentially all inputs since the weighted summation of the inputs for neurons is less than or equal to zero ($x \leq 0$). The gradient for that layer is zero, but also for all the layers upstream. In this state, no gradients flow backward through the neuron, and so it becomes stuck in a perpetually inactive state and ***"dies"***.

It may be mitigated by using alternative function, such as the ***PReLU (Parametric ReLU)*** shown in Figure 20.

$$f(x_i) = \begin{cases} x_i & if\ x_i > 0 \\ \alpha_i\, x_i & otherwise \end{cases}$$

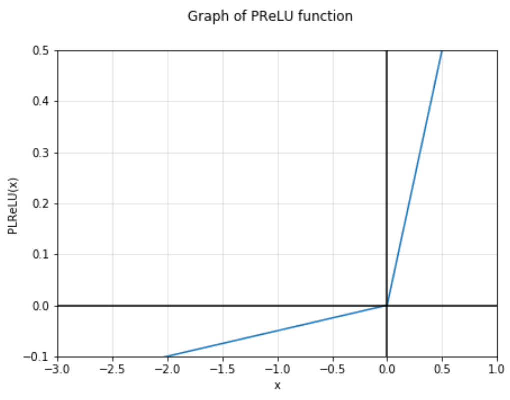

Figure 20: PReLU activation function.

The idea is to provide an α parameter ***slope*** for negative inputs, so that the outputs are set to very small, but non-zero values. The α value is learned from data and varies for each channel. When α_i values are learned to be zero, ***PReLU*** behaves like ***ReLU***. To learn α_i an optimization is performed through

Momentum Gradient Descent. According to the authors [16] it can lead to a significant reduction in errors.

Also for speed branch ***PReLU*** activation on output layer is experimented, showing better driving performance respected to ***Linear*** one, especially due to a less incident ***Inertia Problem***. For the comparison you can refer to Experiment ResNet50V2 8 and Experiment ResNet50V2 9, or to Experiment ResNet34 1 and Experiment ResNet34 2, respectively.

The final mapping for the output activation functions for each network branch is shown in Table 1:

OUTPUT	***ACTIVATION FUNCTION***	***PARAMETERS***
STEER	*Hyperbolic Tangent*	/
THROTTLE	*PReLU*	$starting\ \alpha = 0.5$
BRAKE	*PReLU*	$starting\ \alpha = 0.5$
SPEED	*PReLU*	$starting\ \alpha = 0.5$

Table 1: Activation functions for last layers.

4.1.3 Weights Initialization

The ***ResNet50*** perception backbone has been pre-trained on ***ImageNet*** as indicated in [5] to reduce training variance and obtain more stable policies.

For the remaining layers of the network, an appropriate initialization of the weights must be found. In the original CILRS model, all the fully-connected hidden layers have the ***ReLU*** activation function, except for the terminal layers for which, if we refer to the architecture defined in 4.1.2, ***PReLU*** and ***Hyperbolic Tangent*** are added.

The practices followed for the initialization of these weights are reported below.

- ***Hyperbolic Tangent-activated layers***: ***Uniform Xavier Initialization*** is a standard for this activation. Weights are drawn from:

$$w = U\left[-\sqrt{\frac{6}{(fanin + fanout)}}\ ,\sqrt{\frac{6}{(fanin + fanout)}}\right]$$

where U is a ***Uniform probability distribution*** and $fanin$ and $fanout$ are respectively the number of inputs and the number of outputs for a layer. The logic of Xavier's initialization method [17] is to set an equal variance of the inputs and outputs of each layer such that the variance of activations is the same across every layer. This can help to prevent the gradient from ***exploding*** or ***vanishing***.

- ***ReLU and PReLU-activated layers***: the ***"Xavier"*** initialization is found to have problems when used to initialize networks that use ***ReLU*** activation function. ***He Normal Initialization*** is used instead. Weights are drawn from:

$$w = G\left(0.0\,, \sqrt{2/_{fanin}}\right)$$

where G is a ***Gaussian probability distribution*** with mean 0.0 and standard deviation $\sqrt{2/_{fanin}}$, and where $fanin$ is the number of nodes in the prior layers. [18] It is almost similar to Xavier's initialization, except for the use of different scaling factor for the weights. In this case it is necessary to model the non-linearity of the ReLU which makes deep models ($> 30\, layers$) difficult to converge. The idea is to balance the variance of the activation to avoid reducing or magnifying the magnitudes of inputs signals exponentially, again preventing the gradient from ***exploding*** or ***vanishing.***

4.2 DATASET

The dataset used is ***CARLA100*** [5], introduced in 3.2. Below are the considerations that led to the choices regarding the data. To create an End-To-End Multimodal system, not all the data contained in it can be used, since the depth information are not present for all samples of the dataset. The hours that can be used are approximately ***25 hours***, and, therefore, about 900000 samples, considering the simulator running at 10 fps during acquisitions.

4.2.1 DATA DISTRIBUTION

Analysing the ***usable data***, it was noticed that samples are not perfectly balanced in terms of high-level command as shown in Figure 21. The highest concentration of samples can be found for the Follow Lane command, corresponding to driving without approaching any intersection since it is the most common situation, even if the number of samples for the other commands are not too far.

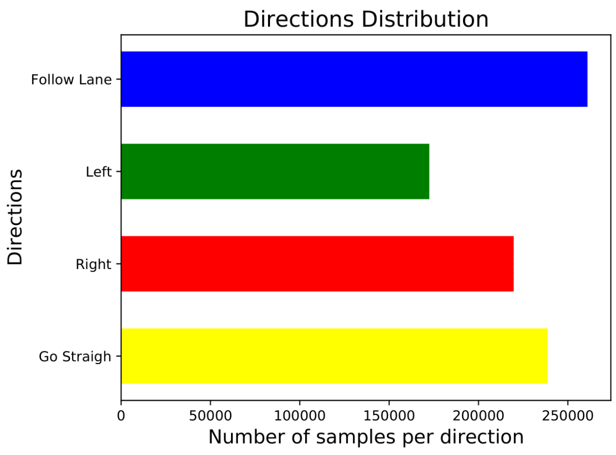

Figure 21: Number of samples per direction in the usable part of CARLA 100 dataset.

Other considerations can be made regarding the distribution of data for the different training weather conditions in Figure 22. In this case it is possible to notice a certain homogeneity on the number of samples per weather.

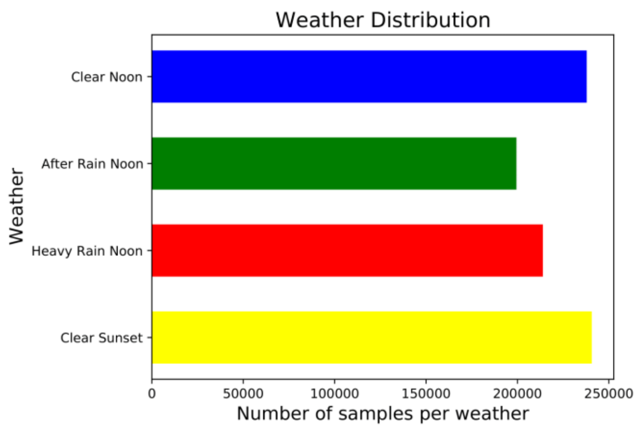

Figure 22: Number of samples per weather condition in the usable part of the CARLA 100 dataset.

If we sample data partitions for model training, we observe that the distribution of samples for the high-level commands does not change significantly, since the episodes statistically have a comparable number of intersections. This reasoning does not extend to the distribution of samples per weather conditions: each episode has a weather and the number of samples contained within it is related to the length of the path and to dynamic events of the scene (e.g. traffic lights). Therefore, sampling to fill partitions can induce a change in the distribution.

In this thesis, the models will be trained without any kind of manipulation and balancing of the data. *We are aware that the non-balancing of data could result in worse driving performance, but it is believed that in the construction of a data-based driving system, it is necessary to foresee that not all conditions can be adequately covers. In this context, an attempt is made to work out a satisfactory solution.*

4.2.2 TRAINING AND VALIDATION SIZE

As demonstrated in [5] in which CARLA100 dataset is introduced, the best results are obtained with 10 hours of training data, in particular on the Dense Traffic tasks and novel conditions such as New Weather and New Town. The risk in increasing the number of training samples is the ***overfitting*** the data that lacks diversity ***(Dataset Bias Problem)***, due to the spatial limitation and visual variety of the environment, even in terms of dynamic objects. Furthermore, it is shown that an increase in data corresponds to an increase in the ***Inertia Problem***. If we consider the entire size of the dataset, only 10% is used. On other hand, in [9] 25 hours of data are used for training, but balances are made in terms of weather conditions and direction commands.

To evaluate ***offline*** the prediction capacity during training, a ***validation set*** is added to the training set, used at the end of each learning step to evaluate the loss of the model on unseen data but also to select the best model from an experiment.

Different data split are considered to evaluate the impact of the ***Dataset Bias Problem***. These are reported in Table 2.

TRAINING	***VALIDATION***
5	*2*
8	*8*
10	*5*
12	*3*
16	*4*

Table 2: Number of hours for Training set and Validation set.

4.2.3 DATA SAMPLING

The data in CARLA100 are organized by episodes, each of which is different for weather conditions, actors, starting and destination points. In this work two different ways of sampling data are analyzed:

- training and validation sets are filled using the episodes completely, i.e. we can find samples from an episode only in one of the two partitions as shown in Figure 23. We will refer to this data sampling as ***non-shuffled***.

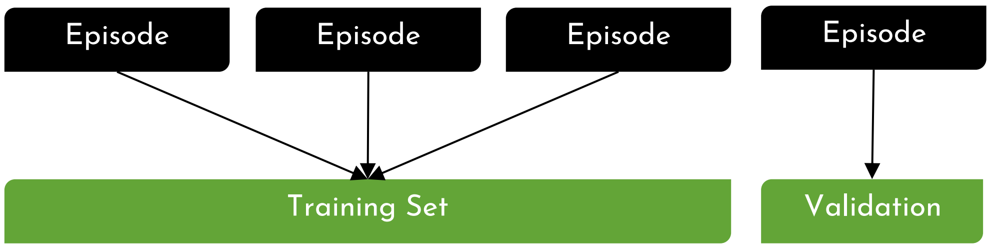

Figure 23: Non-shuffled partitioning of the dataset.

- training and validation sets are filled following a shuffle of usable data, i.e. in the two partitions we can find samples from the same episodes as shown in Figure 24. We will refer to this data sampling as ***shuffled***.

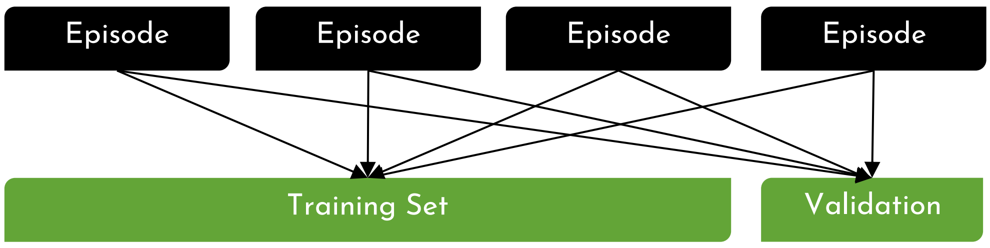

Figure 24: Shuffled partitioning of the dataset.

In any case, during the training a random sampling of the data is carried out online. We found that in general, ***non-shuffled*** sampling reported better results in the experiments.

4.2.4 DATA DIVERSIFICATION

To build the Training set and Validation set, we apply the data diversification protocols [8]: we incorporate in both partitions data affected by the addition noise to the trajectory during the acquisition to simulate drift-away (Figure 25 – Left), and data acquired through perception sensors rotated to the right and left with the corresponding steering values to simulate recovery situations (Figure 25 – Right). The paper states that these two practices can increase the correlation between Open-loop and Closed-loop metrics.

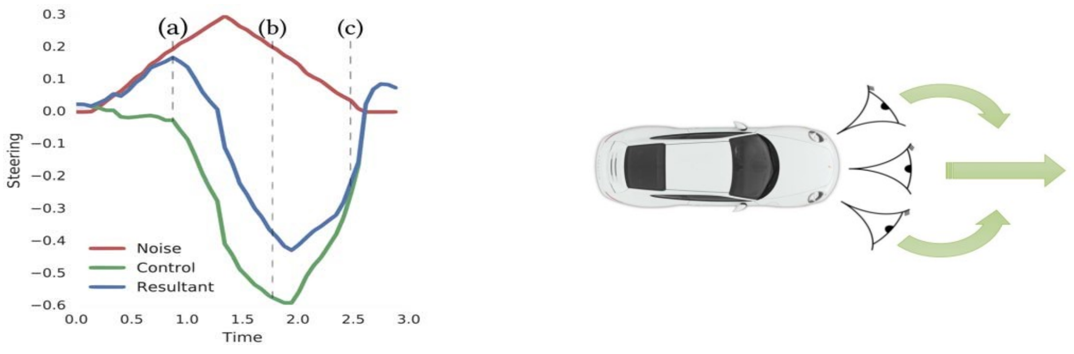

Figure 25: (Left) Noise injection. (Right) Lateral cameras and corresponding recovery actions.

4.2.5 DATA AUGMENTATION

In [5] [9], photometric and geometric data augmentation policies are not used because they do not bring benefits. In this thesis, the data augmentation is revaluated for the purpose of limiting some of the causes that lead to failures, related to shadows, light conditions, sun, puddles and traffic lights. So, some policies are developed to be used online when training a model. To make them, the ***Automold*** library [19] is used. It is a driving-oriented library, which allows you to manipulate images to simulate different lighting or weather conditions, but also to slightly modify the environment in which the ego-vehicle drives. The allowed transformations are***: Random Hue Variation, Add Gravel, Add Shadow, Add Fog, Add Rain, Darken, Brighten, Darken & Brighten***.

Some examples of these can be seen in Figure 26.

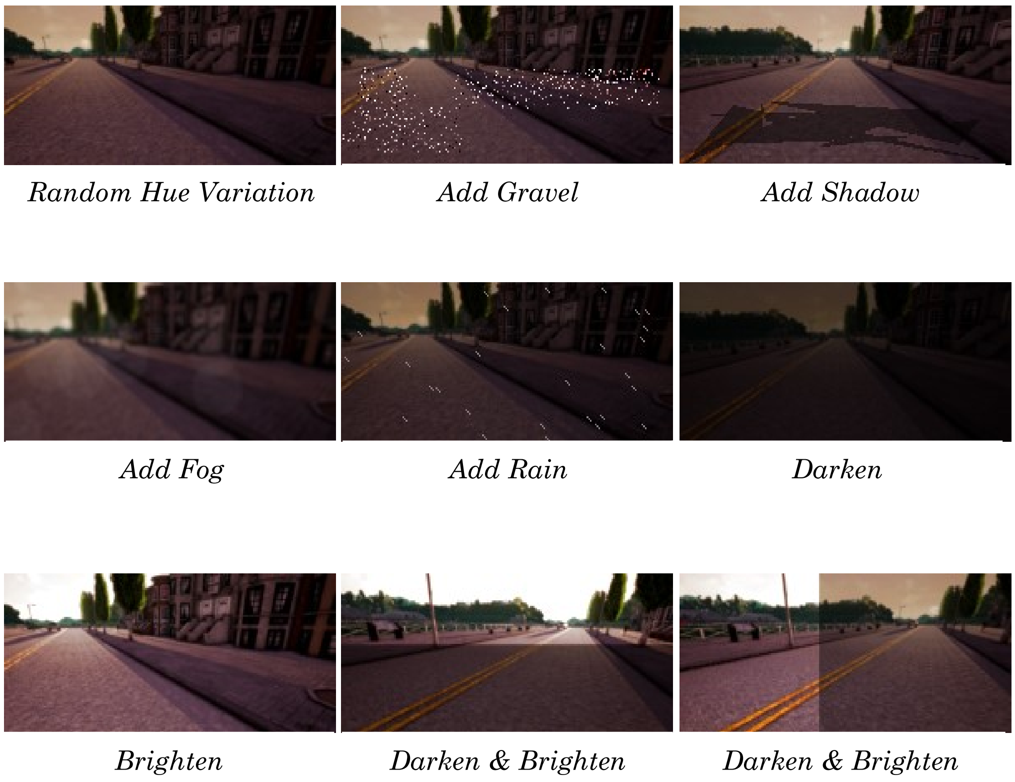

Figure 26: Some Data augmentation examples.

Three different sets of transformations are employed during training:

- ***Weak*** (Random Hue Variation, Add Shadow, Add Fog, Darken, Brighten).
- ***Medium*** (Random Hue Variation, Add Shadow, Add Fog, Darken, Brighten, Darken & Brighten).
- ***Strong*** (Random Hue Variation, Add Shadow, Add Fog, Darken, Brighten, Darken & Brighten, Add Rain, Add Gravel).

For each image to be presented to the network, data augmentation can be applied with a certain probability, by selecting a random subset of transformation with random sampled magnitudes.

Another policy proposed is the ***Horizontal Flip*** of the images together with the inversion of steering and a change of command from Turn Left to Turn Right and viceversa, or no change in the case of Follow Lane and Go Straight.

4.2.6 INPUT MODALITY

The purpose of this thesis is to build an End-To-End Multimodal System, therefore, to evaluate if adding the depth channel can help to obtain a better performing model, we carry out a comparison between RGB and RGBD inputs modalities obtained by ***Early Fusion*** [9].

4.2.7 DATA NORMALIZATION

During the training, the inputs to the perception and measurements modules are normalized. There are two main reasons for doing this:

- the first is that the model has multiple inputs, with different scales: the speed can assume values in the range $[0, 35]$ km / h while for RGB images, the pixels can assume values in the range $[0, 255]$. This situation could give rise to greater influence in the final results for some of the inputs, with an imbalance not due to the intrinsic nature of the data but simply to their original measurement scales. Normalizing all features in the same range avoids this type of problem.

- the second is related to the gradient problem we mentioned in 4.1.2. The rescaling of the input within small ranges gives rise to even small weight values in general, and this makes the output of the units of the network near the saturation regions of the activation functions less likely.

4.3 Loss Function

4.3.1 L1 Loss

The L1 loss introduced in [5] and [9] is characterized by four different parameters that must be specified to weight the steering, throttle, brake and speed components. The per-sample loss is defined as

$$\ell_a(a, a^{gt}, w; c;\ m, m^{gt}) = \beta\ \ell_a(a, a^{gt}, w; c) + (1 - \beta)\ \ell_m(m, m^{gt})$$

The used weights in papers are reported in Table 3.

OUTPUT	***WEIGHT***
STEER	*0.50*
THROTTLE	*0.45*
BRAKE	*0.05*
$\boldsymbol{\beta}$	*0.95*

Table 3: Starting weights for the L1 loss function.

These are the starting point and are modified during the various experiments in section 5. In particular, it is observed that steer, throttle and brake weights are good choices. Instead, β used to balance the speed in the overall loss, needed more attention: one of the main causes of episode failure is ***Inertia Problem***, for which the speed prediction is proposed as a possible solution in CILRS.

4.3.2 L1 Loss Distribution Weight

The L1 loss function in [9] is used successfully since data are balanced in high-level commands per batch. Unbalanced batches can bias the loss function towards the most frequent data, so that the losses of the most penalized branches are spread over the overall batch loss, and therefore not very perceptible. This can lead to unsatisfactory training for such branches.

In this thesis we propose a variant of the L1 loss, capable of capturing the distribution of commands within batches. To do it, the distribution p_c of the command c within the batch of origin is determined and is used to weigh the loss.

$$\ell_a(a, a^{gt}, w; c;\ m, m^{gt}) = \beta\,(1 - p_c)\ell_a(a, a^{gt}, w; c) + (1 - \beta\,)\,\ell_m(m, m^{gt})$$

This additional term allows you to increase the influence on batch loss of those less frequent commands. Also, it can be seen as possible alternative to the manipulation of data that is carried out in the reference paper. The benefits of its use in terms of driving performance will therefore be assessed.

4.4 TRAINING, VALIDATION AND TESTING PROTOCOLS

4.4.1 SETTING

For a trained model, it is necessary to evaluate the actual driving performance, in order to evaluate whether the loss values obtained lead to a vehicle capable of circulating in an urban environment. Table 4 shows the weather conditions and maps used during training, validation and testing of the models. The training scenarios are those on which the ***CARLA100*** dataset data are acquired, while for testing the scenarios are those of ***CoRL2017 Benchmark*** and ***NoCrash Benchmark***. For validation, however, the scenarios are selected following the criteria reported in section 4.4.3.

Weather	***Training (Dataset)***	***Validation (Simulator)***	***Testing (Benchmark)***
Wet cloudy noon	▨	▨	*Town 1 & 2*
Soft rainy sunset	▨	*Town 2*	*Town 1 & 2*
Clear noon	*Town 1*	*Town 2*	*Town 1 & 2*
Clear after rain	*Town 1*	▨	*Town 1 & 2*
Clear sunset	*Town 1*	▨	*Town 1 & 2*
Heavy rain noon	*Town 1*	▨	*Town 1 & 2*

Table 4: Weather conditions and Towns used for Training, Validation and Testing protocols.

4.4.2 TRAINING PROTOCOL

To train the model, the typical procedure is used which consists in a ***training set*** and a ***validation set***. These are of different sizes as reported in 4.2.2. A ***test set*** is not employed since driving performance are evaluated online on simulator. During training runs we adopt ***Adam Optimizer*** [20] and ***minibatch***, with different batch size, mainly 128, with demonstrations randomly sampled from the training set. Different starting ***learning rate*** are considered with a lower bound of 0.000125. This divided by 2 if for a certain number of epochs validation loss does not fall by a certain fixed delta. In addition, we use validation set to discover when to stop the training process.

To feed the model, original sensors channel with a size of $800 \ x \ 600$ pixels, are trimmed to remove sky and very close areas, and thus reshaped to obtain channels of $200 \ x \ 88$ pixels. Finally, if necessary, the data augmentation policies presented in 4.2.5 can be applied.

4.4.3 VALIDATION PROTOCOL

It is meant to select a models to benchmark and is inspired by the one in [9] and explained 4.4.1. A success score V_p is determined balancing the training and testing conditions shown in the Table 4. In particular, we consider ***Town 2*** (not seen in training data), in combination with ***Clear Noon*** (seen in training data) and ***Soft Rain Sunset*** (not seen in training data). The Closed-loop metric adopted is the ***success rate***, since it is strongly correlated to average completion of paths and with the number of Km travelled between infractions. The success rate for each combinations is calculated over 25 episodes, in a dynamic scenario with 15 vehicles and 50 pedestrians. To make a comparison of the proposed models with those of the [5], the validation is done using both the metrics proposed in ***CoRL2017 Benchmark*** and ***NoCrash Benchmark***. In the first, an episode is considered to be successful if the target position is reached within the time limit, while in the second no collision are allowed. The final value is defined weighing the score on ***New Town*** scenario V_t by 0.33 and the score on ***New Weather&Town*** V_{wt} scenario by 0.66.

$$V_p = 0.33\, V_t + 0.66\, V_{wt}$$

This is computed for both metrics. We will refer to them as V_{pc} and V_{pn}.

4.4.4 TESTING PROTOCOL

Only the best model is benchmarked. The benchmark considered is ***NoCrash***, as it is better able to capture the vehicle's performance in dynamic urban environments compared to ***Corl2017***, in which only one of the four tasks provides a dynamic scenario. Therefore, the driving takes place under all possible combination of weather and towns reported in Table 4.

Chapter 5.

EXPERIMENTAL VALIDATION

This chapter provides a view of the experiments carried out to build a model capable of interacting effectively with the environment, providing an explanation of the results and a comparison with those of [5].

5.1 EXPERIMENTS

For each experiment two tables are reported:

- the first one contains the experimental setting as follow: training hours, validation hours, the sampling for data, the batch size, channels, loss, loss weights, activation functions used for all output layers, if we apply data augmentation and the set of transformations, the learning rate used for Adam optimizer;
- the second one, instead, reports the result coming from the training and validation procedures. For the experiments based on ResNet50V2, the protocols reported in Training, validation and testing protocols 4.4 are followed, so the table report the model selected through validation loss, for which the two validation on simulator are performed. For the experiments based on ResNet34, the protocols in 4.4 are compared with those used in [9]: the table shows the model with the highest V_{pn} value obtained either through the protocols in 4.4 or through the procedures in [9] i.e. a validation on simulator after a certain number of training epochs for instance 20.

Experiment ResNet50V2 1

Setting	
Training hours	*5*
Validation hours	*2*
Data sampling	*Non-shuffled*
Batch size	*120*
Channels	*RGB*
Loss	*L1*
Loss weights	*Steer = 0.60, Throttle = 0.35, Brake = 0.05, β = 0.95*
Activation function	*Steer: Linear, Throttle: Linear, Brake: Linear, Speed: Linear*
Data augmentation	*No*
Learning rate	*0.005*

Table 5: Setting for Experiment 1 based on ResNet50V2.

Results	
Best Training Epoch	*64 / 100*
Training Loss	*0.0544*
Validation Loss	*0.0520*
Validation score V_{pc}	*0.0396*
Validation score V_{pn}	*0.0000*

Table 6: Results for Experiment 1 based on ResNet50V2.

This experiment is intended to evaluate the driving performance of the starting model in the case of a few hours of training. V_{pc} and V_{pn} show the inability of the model to drive in a dynamic scenario: main problems are curves and braking to avoid static and dynamic collisions.

Experiment ResNet50V2 2

Setting	
Training hours	*5*
Validation hours	*2*
Data sampling	*Non-shuffled*
Batch size	*64*
Channels	*RGB*
Loss	*L1*
Loss weights	*Steer = 0.55, Throttle = 0.40, Brake = 0.05, β = 0.90*
Activation function	*Steer: Tanh, Throttle: ReLU, Brake: ReLU, Speed: Linear*
Data augmentation	*No*
Learning rate	*0.01*

Table 7: Setting for Experiment 2 based on ResNet50V2.

Results	
Best Training Epoch	*29 / 59*
Training Loss	*0.1239*
Validation Loss	*0.1138*
Validation score V_{pc}	*0.0000*
Validation score V_{pn}	*0.0000*

Table 8: Results for Experiment 2 based on ResNet50V2.

V_{pc} and V_{pn} tell us that adding ReLU for throttle and brake never succeeds. This can be explained through ***Dying ReLU Problem*** which lead to constantly zero values for throttle and brake.

Experiment ResNet50V2 3

Setting	
Training hours	*8*
Validation hours	*2*
Data sampling	*Non-shuffled*
Batch size	*64*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.60, Throttle = 0.35, Brake = 0.05, β = 0.90*
Activation function	*Steer: Tanh, Throttle: ReLU, Brake: ReLU, Speed: Linear*
Data augmentation	*No*
Learning rate	*0.01*

Table 9: Setting for Experiment 3 based on ResNet50V2.

Results	
Best Training Epoch	*51 / 51*
Training Loss	*0.1534*
Validation Loss	*0.1445*
Validation score V_{pc}	*0.0000*
Validation score V_{pn}	*0.0000*

Table 10: Results for Experiment 3 based on ResNet50V2.

We can observe that neither the addition of depth channel nor the increase in the hours of demonstration have induced an improvement, always obtaining a vehicle unable to move. This is still due to the ***Dying ReLU Problem***.

Experiment ResNet50V2 4

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGB*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.95*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: Linear*
Data augmentation	*No*
Learning rate	*0. 0002*

Table 11: Setting for Experiment 4 based on ResNet50V2.

Results	
Best Training Epoch	*38 / 50*
Training Loss	*0.0196*
Validation Loss	*0.0262*
Validation score V_{pc}	*0.3960*
Validation score V_{pn}	*0.3036*

Table 12: Results for Experiment 4 based on ResNet50V2.

Here we present a possible solution to the ***Dying ReLU Problem***. As described in 4.1.2, the ***ReLU*** for throttle and brake outputs are replaced with ***PReLU***. This is used in conjunction with more hours of training demonstrations. At first glance, a clear reduction in losses can be observed on the two datasets partitions compared to previous models. About driving, the vehicle is currently able to move and complete episodes. On both validations, we get few collisions, and few off-road. The main cause of failure are linked to the ***Causal Confusion***.

Experiment ResNet50V2 5

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.95*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: Linear*
Data augmentation	*No*
Learning rate	*0. 0002*

Table 13: Setting for Experiment 5 based on ResNet50V2.

Results	
Best Training Epoch	*45 / 65*
Training Loss	*0.0191*
Validation Loss	*0.0285*
Validation score V_{pc}	*0.4752*
Validation score V_{pn}	*0.4356*

Table 14: Results for Experiment 5 based on ResNet50V2.

Here we highlight the improvements introduced by adding the Depth channel through ***Early Fusion***. We get a clear increase in driving performance on both validations due to the reduction of the ***Causal Confusion***. Conversely, an increase in collisions is observed, mainly on the New Weather&Town scenarios.

Experiment ResNet50V2 6

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.95*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*No*
Learning rate	*0.0002*

Table 15: Setting for Experiment 6 based on ResNet50V2.

Results	
Best Training Epoch	*25 / 62*
Training Loss	*0.0246*
Validation Loss	*0.0280*
Validation score V_{pc}	*0.7128*
Validation score V_{pn}	*0.5676*

Table 16: Results for Experiment 6 based on ResNet50V2.

To limit the ***Inertia Problem***, the ***PReLU*** activation is proposed for the output of speed branch. Results show that, despite similar loss values to Experiment ResNet50V2 5, improvement in terms of V_{pc} and V_{pn} are observed. In general, the number of collisions and off-road decrease, showing the ability of model to react to dynamic scenarios. ***Causal Confusion*** remains but it represents a smaller percentage of failure.

Experiment ResNet50V2 7

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*64*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.95*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*No*
Learning rate	*0. 0002*

Table 17: Setting for Experiment 7 based on ResNet50V2.

Results	
Best Training Epoch	*34 / 47*
Training Loss	*0.0220*
Validation Loss	*0.0293*
Validation score V_{pc}	*0.6600*
Validation score V_{pn}	*0.5412*

Table 18: Results for Experiment 7 based on ResNet50V2.

The same setting of Experiment ResNet50V2 6 is proposed again with a reduction in the batch size from 128 to 64 samples. This can affect the losses values, due to the non-balancing of the samples in terms of direction commands. The values of the losses are quite similar. The same cannot be said for driving performance even if the distance is short and can be partially explained through the variability in the simulations.

Experiment ResNet50V2 8

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.45, Throttle = 0.45, Brake = 0.10, β = 0.90*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Weak*
Probability	*0.35*
Learning rate	*0.0004*

Table 19: Setting for Experiment 8 based on ResNet50V2.

Results	
Best Training Epoch	*53 / 57*
Training Loss	*0.0258*
Validation Loss	*0.0248*
Validation score V_{pc}	*0.7260*
Validation score V_{pn}	*0.5676*

Table 20: Results for Experiment 8 based on ResNet50V2.

It is observed how shadows, light conditions and traffic lights can lead to episode failure. Data augmentation is experimented to limit the impact of these factors. But the goal is not achieved: respect to Experiment ResNet50V2 6 results are almost the same, both for V_{pc} and V_{pn}, but they are dependent on a lower rate of collisions, increasing other causes of failure.

Experiment ResNet50V2 9

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.45, Throttle = 0.45, Brake = 0.10, β = 0.90*
Activation function	*Steer: Linear, Throttle: Linear, Brake: Linear, Speed: Linear*
Data augmentation	*Weak*
Probability	*0.35*
Learning rate	*0.0004*

Table 21: Setting for Experiment 9 based on ResNet50V2.

Results	
Best Training Epoch	*54 / 54*
Training Loss	*0.0300*
Validation Loss	*0.0295*
Validation score V_{pc}	*0.5280*
Validation score V_{pn}	*0.5150*

Table 22: Results for Experiment 9 based on ResNet50V2.

This experiment is carried out in order to determine the best configuration for the activation functions for the output layers. Respect to Experiment ResNet50V2 8, we change the configuration of the output activations. As a result, the ***Tanh-PReLU*** configuration is better. We find more collisions and some out of lane, while the incidence of ***Causal Confusion*** is almost the same.

Experiment ResNet50V2 10

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.95*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Horizontal Flip for Turn Left and Turn Right Commands*
Probability	*0.50*
Learning rate	*0.0002*

Table 23: Setting for Experiment 10 based on ResNet50V2.

Results	
Best Training Epoch	*35 / 40*
Training Loss	*0.0195*
Validation Loss	*0.0272*
Validation score V_{pc}	*0.6864*
Validation score V_{pn}	*0.5676*

Table 24: Results for Experiment 10 based on ResNet50V2.

In this experiment for the Turn Left and Turn Right commands we flip horizontally the perception input, together with a reverse steering and command inversion, from Turn Left to Turn Right or conversely. The losses values remain close to those of Experiment ResNet50V2 6. We get a better generalization capability in the New Weather&Town scenarios on both validation but also more collisions due to a lane invasion problem. ***Causal Confusion*** is still present.

Experiment ResNet50V2 11

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1 Distribution Weight*
Loss weights	*Steer = 0.45, Throttle = 0.45, Brake = 0.10, β = 0.90*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Weak*
Probability	*0.35*
Learning rate	*0.0002*

Table 25: Setting for Experiment 11 based on ResNet50V2.

Results	
Best Training Epoch	*65 / 73*
Training Loss	*0.0167*
Validation Loss	*0.0165*
Validation score V_{pc}	*0.5544*
Validation score V_{pn}	*0.4620*

Table 26: Results for Experiment 11 based on ResNet50V2.

Starting from Experiment ResNet50V2 8, we test the loss function introduced in 4.3.2 with data augmentation. This setting, however, made online driving worse, probably due to a more limited correlation between offline and online metric. More in detail, on validations, good performances are obtained in the New Town scenarios, and a degradation of these in New Weather&Town mainly due to the ***Causal Confusion***. We also find a high ability to avoid collisions.

Experiment ResNet50V2 12

Setting	
Training hours	*16*
Validation hours	*4*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1 Distribution Weight*
Loss weights	*Steer = 0.45, Throttle = 0.45, Brake = 0.10, β = 0.90*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Weak*
Probability	*0.30*
Learning rate	*0.0005*

Table 27: Setting for Experiment 12 based on ResNet50V2.

Results	
Best Training Epoch	*52 / 55*
Training Loss	*0.0156*
Validation Loss	*0.0146*
Validation score V_{pc}	*0.3432*
Validation score V_{pn}	*0.2376*

Table 28: Results for Experiment 12 based on ResNet50V2.

For a further analysis on the modified loss from section 4.3.2, the Experiment ResNet50V2 11 is carried out by increasing the amount of training data. We observe a clear decline in driving performance on validations due to both poor correlation between online and offline performance and increased training data, which, according to [5] corresponds to increased failures and ***Inertia Problem***.

Experiment ResNet50V2 13

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.95*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Medium*
Probability	*0.30*
Learning rate	*0.0002*

Table 29: Setting for Experiment 13 based on ResNet50V2.

Results	
Best Training Epoch	*66 / 67*
Training Loss	*0.0174*
Validation Loss	*0.0161*
Validation score V_{pc}	*0.3696*
Validation score V_{pn}	*0.4092*

Table 30:: Results for Experiment 13 based on ResNet50V2.

Here a different way of sampling the data is experimented, the one described in 4.2.3, as well as an enriched set of possible transformations for data augmentation. A comparison with Experiment ResNet50V2 8 shows that the same architecture, same number of training hours and the same loss have generated a marked decrease in driving performance. In particular, for both validations good results are obtained on the New Town scenario, with few collisions and little incidence of ***Causal Confusion***, but a worsening on the New Weather&Town scenario. This behaviour can be explained by the presence of samples belonging to the same episodes in both datasets partitions which lead to low loss values, but also a low correlation with online driving.

Experiment ResNet50V2 14

Setting	
Training hours	*12*
Validation hours	*3*
Data sampling	*Shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.95*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Horizontal Flip for all Commands*
Probability	*0.30*
Learning rate	*0.0002*

Table 31: Setting for Experiment 14 based on ResNet50V2.

Results	
Best Training Epoch	*63 / 65*
Training Loss	*0.0200*
Validation Loss	*0.0169*
Validation score V_{pc}	*0.3168*
Validation score V_{pn}	*0.2640*

Table 32: Results for Experiment 14 based on ResNet50V2.

In this experiment is evaluated the Flip augmentation described in 4.2.5, but extended to all commands, and therefore also to Go Straight and Follow Lane, for which samples we use only the horizontal flip of perception input and a reversal of the steering. As a result, on both validations the model does not perform well overall, showing better capabilities on the New Town scenario, and a marked worsening on New Weather&Town. This is similar behaviour to that found in the

Experiment ResNet50V2 13 and, therefore, due to the sampling method but, in addition to ***Causal Confusion***, there is also the lane invasion problem that leads to a timeout termination.

Experiment ResNet50V2 15

Setting	
Training hours	*10*
Validation hours	*5*
Data sampling	*Shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.88*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Horizontal Flip for Turn Left and Turn Right Commands*
Probability	*0.25*
Learning rate	*0.0002*

Table 33: Setting for Experiment 15 based on ResNet50V2.

Results	
Best Training Epoch	*63 / 65*
Training Loss	*0.0191*
Validation Loss	*0.0167*
Validation score V_{pc}	*0.1980*
Validation score V_{pn}	*0.1848*

Table 34: Results for Experiment 15 based on ResNet50V2.

From [5]we know that the ***Inertia Problem*** is related to the number of training samples and speed prediction. Therefore, in this experiment, the influence of speed in the loss is increased and less data are used for training. Furthermore, the size of the validation set has been increased in order to obtain a loss that can be better representative of the driving performance and Flip augmentation is provided for the Turn Left and Turn Right commands. This configuration generated a model with poor generalization capabilities on both New Town and New Weather&Town scenarios. In particular, there is a high percentage of failures due to collisions, but at the same time the lane invasions problem. This is further confirmation of the fact that observing samples belonging to the same episodes both in the training and in the validation set leads to a marked worsening of driving performance.

Experiment ResNet50V2 16

Setting	
Training hours	*10*
Validation hours	*5*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.915*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Strong*
Probability	*0.30*
Learning rate	*0.000125*

Table 35: Setting for Experiment 16 based on ResNet50V2.

Results	
Best Training Epoch	*58 / 62*
Training Loss	*0.0179*
Validation Loss	*0.0272*
Validation score V_{pc}	*0.7920*
Validation score V_{pn}	*0.5808*

Table 36: Results for Experiment 16 based on ResNet50V2.

For this experiment we further enrich data augmentation with other transformations, adding rain and gravel. The latter having the purpose of simulating puddles and the reflections of the sun on them. Furthermore, as in [5] we use a smaller number of training samples, obtaining an improvement on both validations.

Experiment ResNet34 1

Setting	
Training hours	*10*
Validation hours	*5*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.92*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*No*
Learning rate	*0.0002*

Table 37: Setting for Experiment 1 based on ResNet34.

Results	
Best Training Epoch	*64 / 70*
Training Loss	*0.0182*
Validation Loss	*0.0336*
Validation score V_{pn}	*0.5808*

Table 38: Results for Experiment 1 based on ResNet34. Here the best model is selected through validation loss.

We experiment ResNet34 in combination with the ***Tanh-PReLU*** configuration for the activation functions of the output layers. Following the training protocol in [9], we perform a validation at 20, 40, 60 and 70 epochs but the highest V_{pn} arise from the model with the best validation loss following the protocol in 4.4. This value is the same reported by the Experiment ResNet50V2 16. Results for New Town Scenarios and New Weather&Town scenarios expanding V_{pn} are summarized in Figure 27, in which we perform a comparison of all performed validations. You can see how the selected model far exceeds the others.

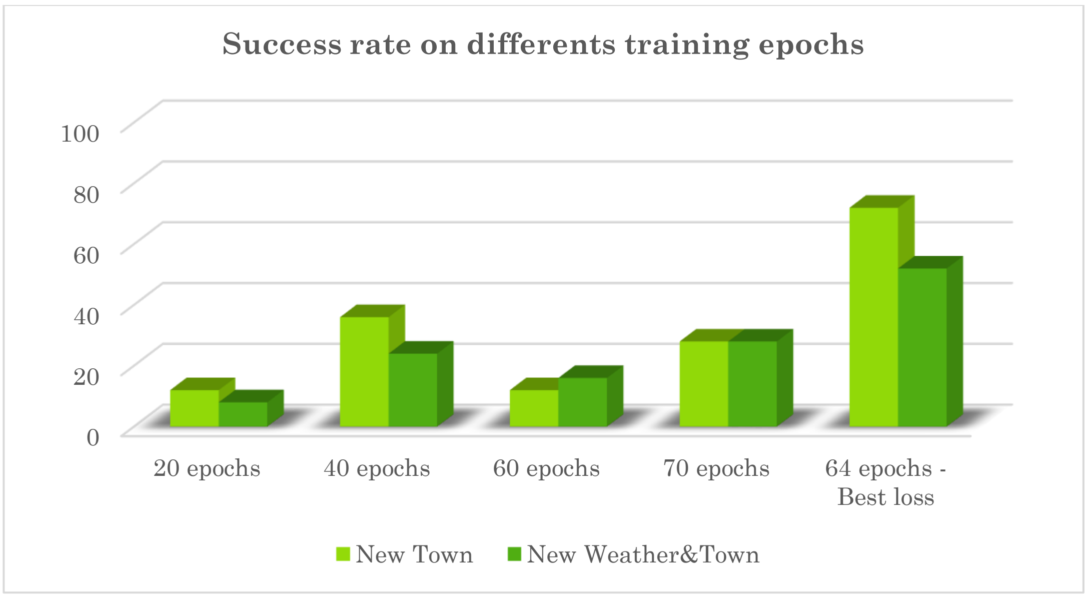

Figure 27: Success rate at different epochs. The highest score was obtained at the epoch 64.

Experiment ResNet34 2

Setting	
Training hours	*10*
Validation hours	*5*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.05, β = 0.92*
Activation function	*Steer: Linear, Throttle: Linear, Brake: Linear, Speed: Linear*
Data augmentation	*No*
Learning rate	*0.0002*

Table 39: Setting for Experiment 2 based on ResNet34.

Results	
Epoch	*40 / 60*
Training Loss	*0.0300*
Validation Loss	*0.0376*
Validation score V_{pn}	*0.5016*

Table 40: Results for Experiment 2 based on ResNet34. Here the best model comes from epoch 40 and not from the model with the lowest validation loss.

Again, we want to evaluate the impact of the ***Tanh-ReLU*** configuration for the output layers of the model against the ***Linear*** one. The highest value of V_{pn} is obtained at epoch 40, and not at epoch 55 which has the lowest loss on validation set. This value, however, is smaller compared to that of the Experiment ResNet34 1. The results for all validations expanding V_{pn} are reported in Figure 28.

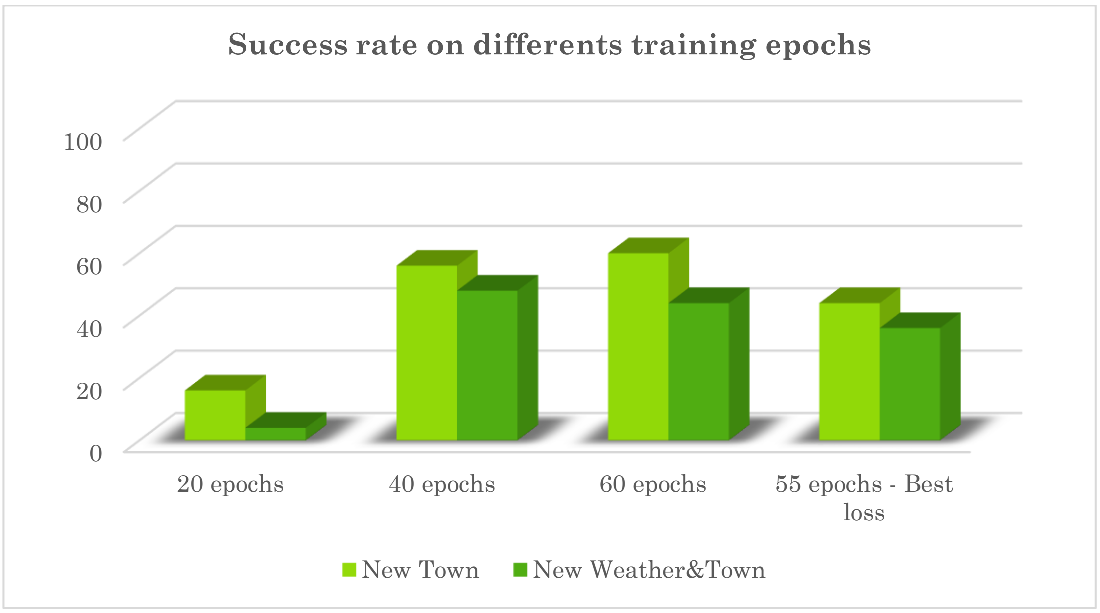

Figure 28: Success rate at different epochs. The highest validation score is obtained at epoch 40, and not at the epoch with the lowest loss value on the validation set.

Experiment ResNet34 3

Setting	
Training hours	*10*
Validation hours	*5*
Data sampling	*Non-shuffled*
Batch size	*128*
Channels	*RGBD*
Loss	*L1*
Loss weights	*Steer = 0.50, Throttle = 0.45, Brake = 0.08, β = 0.92*
Activation function	*Steer: Tanh, Throttle: PReLU, Brake: PReLU, Speed: PReLU*
Data augmentation	*Strong*
Probability	*0.30*
Learning rate	*0.000125*

Table 41: Setting for Experiment 3 based on ResNet34.

Results	
Epoch	*20/80*
Training Loss	*0.0290*
Validation Loss	*0.0413*
Validation score V_{pn}	*0.4490*

Table 42: Results for Experiment 3 based on ResNet34. Here the best model comes from epoch 40 and not from the model with the lowest validation loss.

Here we evaluate the same setting of Experiment ResNet50V2 16, adding strong data augmentation in combination with the ***Tanh-PReLU*** configuration for activation functions of the output layers. Th best results are obtained at epoch 20. Figure 29 summarizes the results for all validations, decomposing V_{pn}. We can see a clear decrease in performance as the epochs grow and, in general, a higher success rate in the New Weather&Town scenarios, due probably to rain transformation. Therefore, the data augmentation does not bring benefits but, on the other hand, leads to a specialization of the model on particular weather conditions.

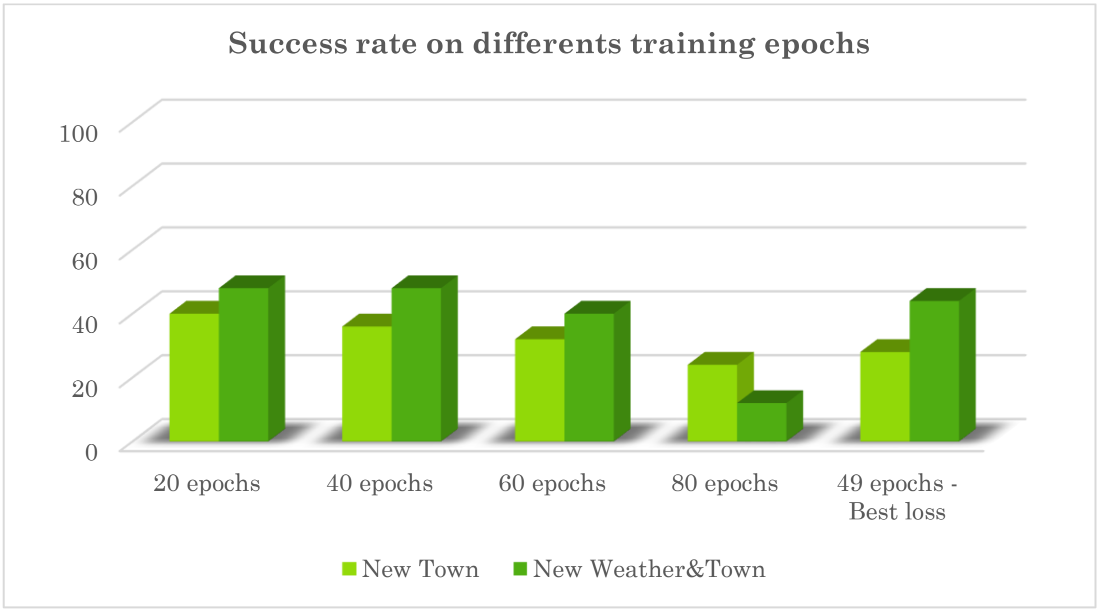

Figure 29: Success rate at different epochs. The highest validation score is obtained at epoch 20, and not at the epoch with the lowest loss value on the validation set.

5.2 Experiment analysis

5.2.1 Dataset Bias Problem

Looking at ResNet50V2 based experiments, the highest validation comes out from Experiment ResNet50V2 16 with 10 hours of training, as in [5]. This is a manifestation of the ***Dataset Bias Problem***. In Figure 30 we compare Experiment ResNet50V2 8 and Experiment ResNet50V2 16: an increase in hours involves a reduction in the success rate when greater generalization capabilities are required in New Weather&Town scenario, increasing it, instead, in New Town scenario with training weather conditions. Other analysis in Figure 31 comes from Experiment ResNet50V2 11 and Experiment ResNet50V2 12 for which a further increase in training exacerbate the problem.

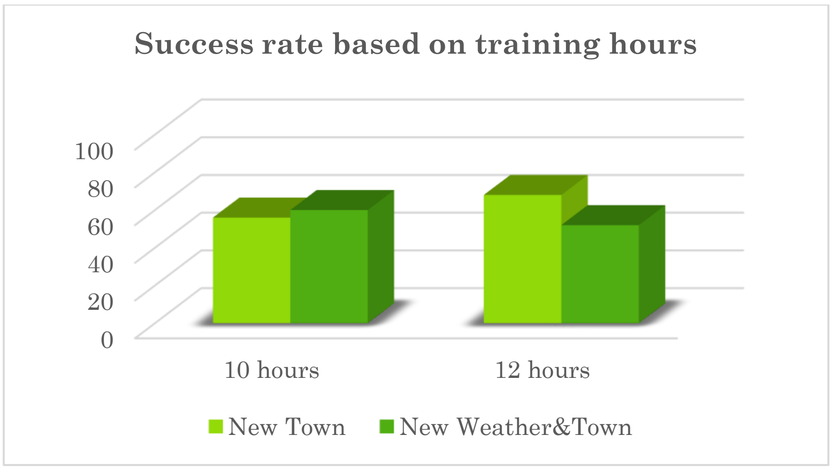

Figure 30: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2-Tanh-PReLU architecture trained on 10 and 12 hours using L1 loss.

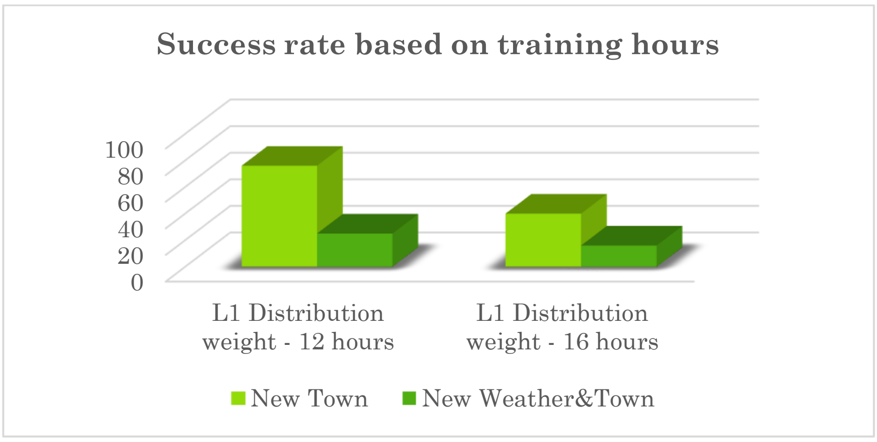

Figure 31: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2-Tanh-PReLU architecture trained on 12 and 16 hours with L1 loss distribution weight.

5.2.2 MULTIMODALITY

A comparison between Experiment ResNet50V2 4 and Experiment ResNet50V2 5 show that RGBD perception input modality can improve generalization ability compared to RGB only as happens in [9]. Figure 32 summarizes this comparison, analysing the components of V_{pn} of the two experiments. In particular, in both scenarios RGBD modality achieves a higher success rate.

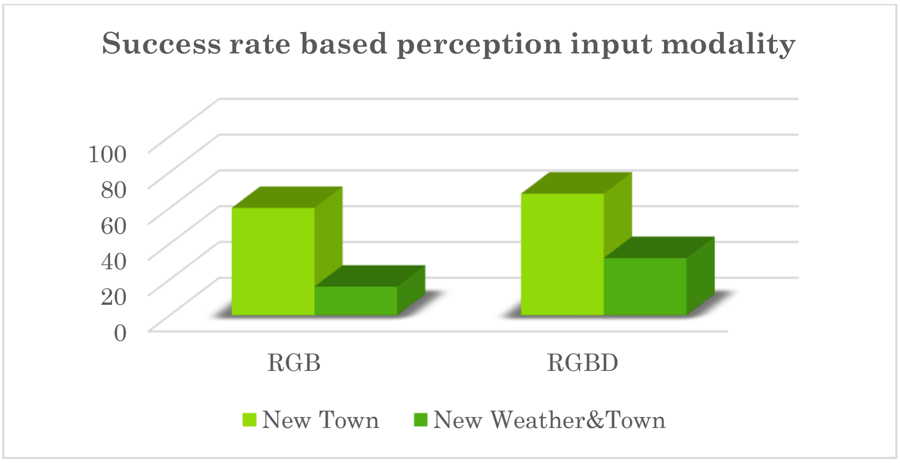

Figure 32: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2 with Tang-PReLU configuration for actions only between RGB and RGBD perception input modality.

5.2.3 Inertia Problem

Speed prediction is added in order to reduce the impact of the ***Inertia Problem***. From Experiment ResNet50V2 5 and Experiment ResNet50V2 6 in Figure 33 appears that the use of PReLU on output of speed prediction branch leads to a reduction in failures due to this cause mainly in New Weather&Town scenarios.

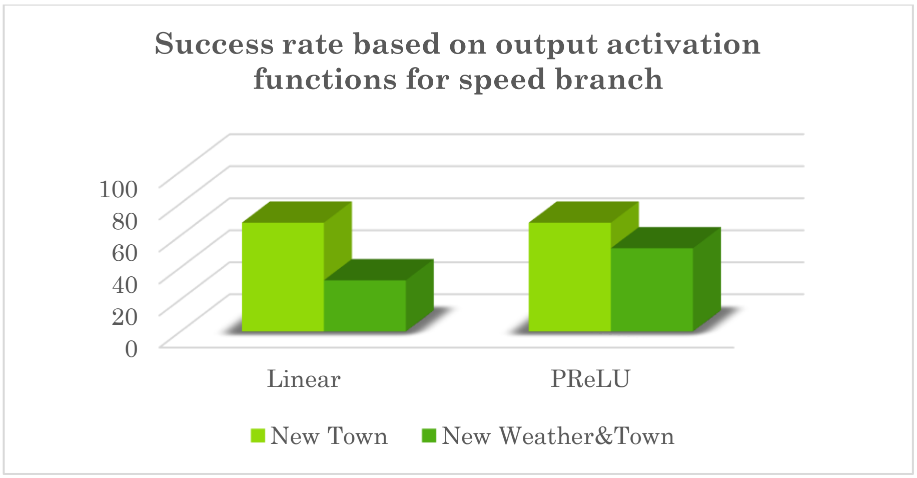

Figure 33: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2 with Linear and PReLU for speed prediction.

5.2.4 Activations

From Experiment ResNet50V2 8 and Experiment ResNet50V2 9, Experiment ResNet34 1 and Experiment ResNet34 2 is clear how the ***Tanh-PReLU*** configuration for outputs activation functions have brought advantages compared to the use of simple ***Linear*** activations. This is summarized in Figure 34 and Figure 35 showing the components of the V_{pn} in the experiments.

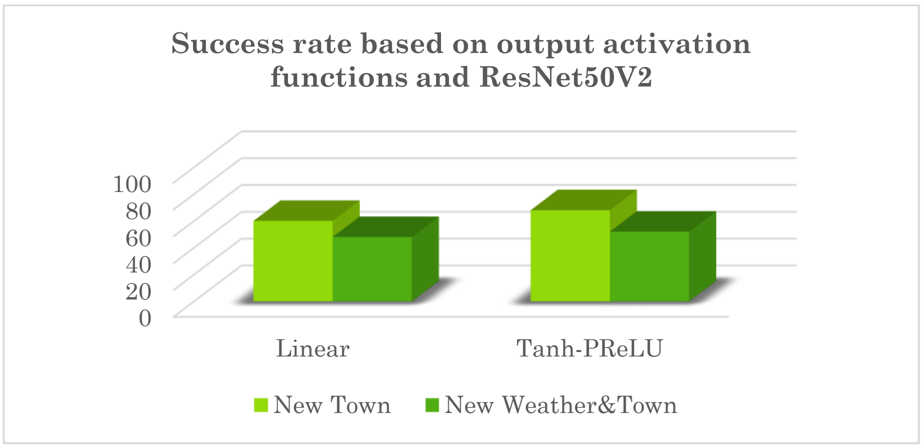

Figure 34: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2 with Linear and Tanh-PReLU configurations for output activation functions.

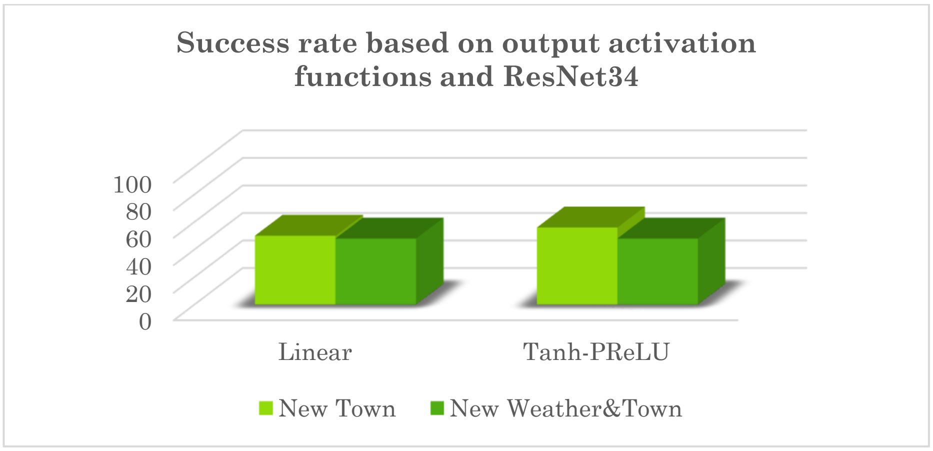

Figure 35: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet34 with Linear and Tanh-PReLU configurations for output activation functions.

5.2.5 DATA AUGMENTATION

In general, the data augmentation brings few benefits, or even worse in driving performance. In Figure 36 a small performance boost is highlighted by Experiment ResNet50V2 6 and Experiment ResNet50V2 8. These are due to a great ability to avoid collisions. Instead, it had less impact on the resolution of ***Causal Confusion***. On the other hand, lower generalization capability comes out from Experiment ResNet34 1 and Experiment ResNet34 3 in Figure 37.

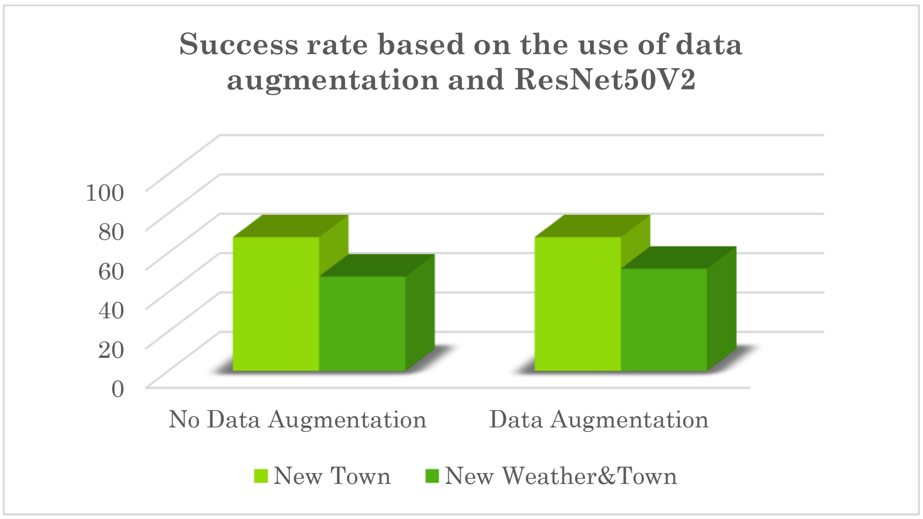

Figure 36: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2-Tanh-PReLU with and without Data Augmentation.

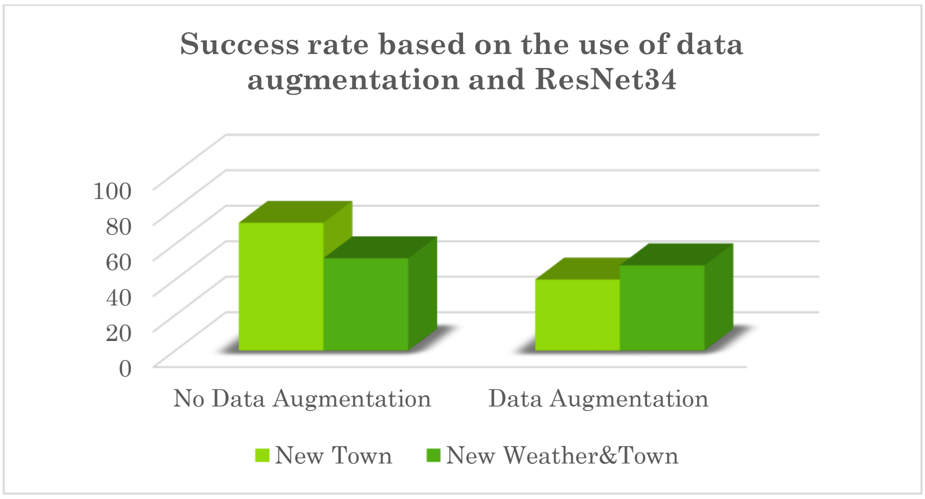

Figure 37: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet34-Tanh-PReLU with and without Data Augmentation

5.2.6 DATA SHUFFLING

By analysing the experiments with shuffled training datasets, the models obtain good driving performance in New Town scenarios, while in New Weather&Town we get a marked worsening. This is evidenced by the comparison between Experiment ResNet50V2 8 and Experiment ResNet50V2 13 in Figure 38.

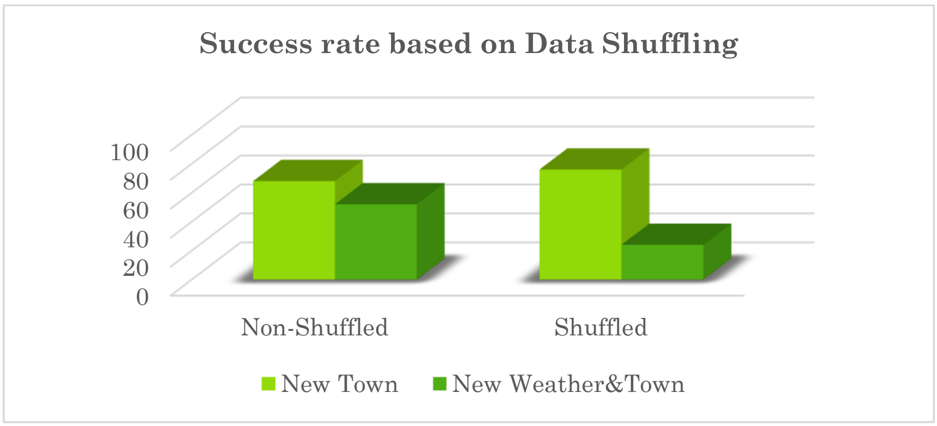

Figure 38: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2-Tanh-PReLU architecture trained with a shuffled and a non-shuffled dataset.

5.2.7 LOSS DISTRIBUTION WEIGHT

The modified loss introduced in section 4.3.2 does not improve overall driving performance. This is applied to Experiment ResNet50V2 11 and Experiment ResNet50V2 12. In Figure 39 we show the comparison with Experiment ResNet50V2 8. With 12 hours of training, the success rate on the New Town scenario is really high, lowering significantly on the New Weather&Town scenario. By increasing the number of hours the problem is exacerbated.

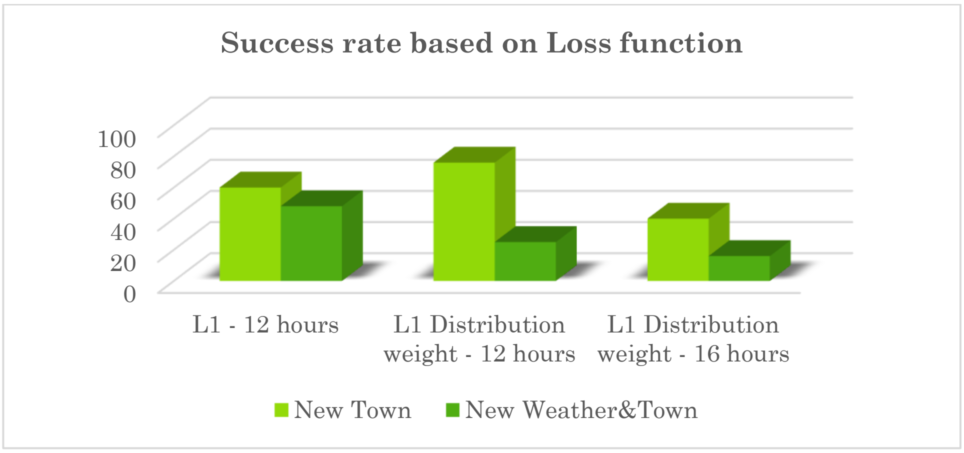

Figure 39: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet50V2-Tanh-PReLU architecture trained using L1 Loss end L1 Distribution weight.

5.2.8 Perception Backbone

In this thesis a comparison is made between the CILRS architecture based on ResNet34 and ResNet50V2 and a ***Tanh-PReLU*** configuration for output layers. Experiment ResNet50V2 16 and Experiment ResNet34 1 obtain the same V_{pn}. Figure 40 shows the success rate in New Town and New Weather & Town scenarios, analysing the components of V_{pn}. While ResNet50V2 has a better ability to generalize in never observed weather conditions, ResNet34 is better if the weather conditions are already observed during training.

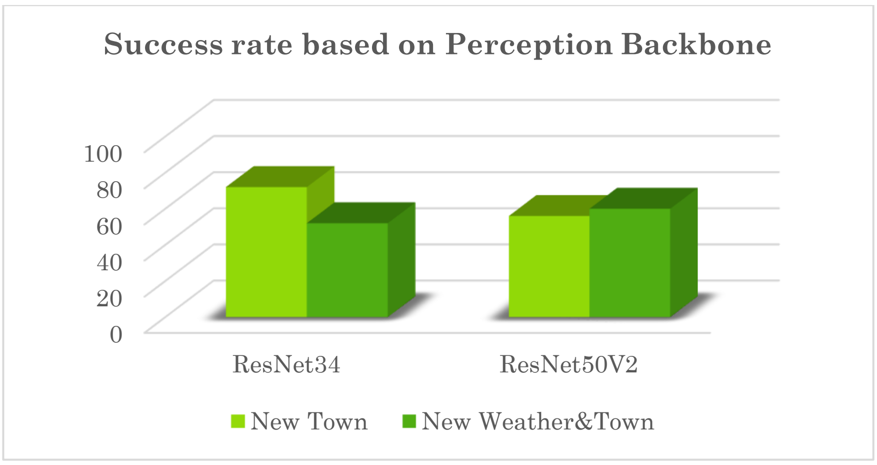

Figure 40: A comparison of success rate in New Town and New Weather & Town scenarios for ResNet64-based and ResNet50V2-based architectures.

5.3 Benchmark Evaluation

Following the testing protocol in section 4.4.4, ***NoCrash Benchmark*** is applied to the models selected through the validation scores. For the purpose of a comparison between architectures with different perception backbones, we consider the models that come from Experiment ResNet50V2 16 and Experiment

ResNet34 1. We refer with ***CILRS50*** to the architecture based on ResNet50V2 perception backbone, while with ***CILRS34*** to the one based on ResNet34.

Primarily, a view of the success rate is provided on the basis of the tasks, weathers and maps, and then a comparison is made with the results from literature.

5.3.1 NoCrash Benchmark for CILRS34

TOWN 1 - TRAINING WEATHERS

	Empty	***Regular Traffic***	***Dense Traffic***
CLEAR NOON	*96*	*80*	*48*
AFTER RAIN NOON	*92*	*80*	*32*
HEAVY RAIN NOON	*100*	*80*	*44*
CLEAR SUNSET	*96*	*92*	*48*

Table 43: Success rates on training weathers on Town 1 for the ResNet34-based architecture.

TOWN 1 - TEST WEATHERS

	Empty	***Regular Traffic***	***Dense Traffic***
RAINY AFTER RAIN	*92*	*84*	*44*
SOFT RAIN SUNSET	*92*	*72*	*36*

Table 44: Success rates on testing weathers on Town 1 for the ResNet34-based architecture.

TOWN 2 - TRAINING WEATHERS

	Empty	***Regular Traffic***	***Dense Traffic***
CLEAR NOON	*84*	*60*	*28*
AFTER RAIN NOON	*52*	*36*	*8*
HEAVY RAIN NOON	*60*	*36*	*4*
CLEAR SUNSET	*80*	*60*	*24*

Table 45: Success rates on training weathers on Town 2 for the ResNet34-based architecture.

TOWN 2 - TEST WEATHERS

	Empty	*Regular Traffic*	*Dense Traffic*
RAINY AFTER RAIN	*48*	*40*	*4*
SOFT RAIN SUNSET	*48*	*56*	*16*

Table 46: Success rates on testing weathers on Town 2 for the ResNet34-based architecture

5.3.2 NoCrash Benchmark for CILRS50

TOWN 1 - TRAINING WEATHERS

	Empty	*Regular Traffic*	*Dense Traffic*
CLEAR NOON	*100*	*92*	*44*
AFTER RAIN NOON	*96*	*88*	*56*
HEAVY RAIN NOON	*100*	*88*	*40*
CLEAR SUNSET	*100*	*84*	*36*

Table 47: Success rates on training weathers on Town 1 for the ResNet50V2-based architecture.

TOWN 1 - TEST WEATHERS

	Empty	*Regular Traffic*	*Dense Traffic*
RAINY AFTER RAIN	*76*	*76*	*40*
SOFT RAIN SUNSET	*96*	*80*	*28*

Table 48: Success rates on testing weathers on Town 1 for the ResNet50V2-based architecture.

TOWN 2 - TRAINING WEATHERS

	Empty	*Regular Traffic*	*Dense Traffic*
CLEAR NOON	*76*	*40*	*20*
AFTER RAIN NOON	*68*	*48*	*8*
HEAVY RAIN NOON	*76*	*76*	*12*
CLEAR SUNSET	*92*	*44*	*28*

Table 49: Success rates on training weathers on Town 2 for the ResNet50V2-based architecture.

TOWN 2 - TEST WEATHERS

	Empty	***Regular Traffic***	***Dense Traffic***
RAINY AFTER RAIN	*32*	*36*	*4*
SOFT RAIN SUNSET	*60*	*44*	*20*

Table 50: Success rates on testing weathers on Town 2 for the ResNet50V2-based architecture.

5.3.3 Results comparison

In Table 51 we compare our results with those obtained in [5] and [10] on ***NoCrash Benchmark***. We denote with ***CIL*** the architecture coming from [7] and with ***CILRS*** to the one coming from [5].

SCENARIO	*TASK*	*CIL*	*CILRS*	*CILRS34*	*CILRS50*
TRAINING	***Empty***	*79 ± 1*	*97 ± 2*	*96*	***<u>99</u>***
	Regular	*60 ± 1*	*83 ± 0*	*83*	***<u>88</u>***
	Dense	*21 ± 2*	*42 ± 2*	*43*	***<u>44</u>***
NEW WEATHER	***Empty***	*83 ± 2*	***<u>96 ± 1</u>***	*92*	*86*
	Regular	*55 ± 5*	*77 ± 1*	***<u>78</u>***	***<u>78</u>***
	Dense	*13 ± 4*	***<u>47 ± 5</u>***	*40*	*34*
NEW TOWN	***Empty***	*48 ± 3*	*66 ± 2*	*69*	***<u>78</u>***
	Regular	*27 ± 1*	*49 ± 5*	*48*	***<u>52</u>***
	Dense	*10 ± 2*	***<u>23 ± 1</u>***	*16*	*17*
NEW WEATHER&TOWN	***Empty***	*24 ± 1*	***<u>90 ± 2</u>***	*48*	*46*
	Regular	*13 ± 2*	***<u>56 ± 2</u>***	*48*	*40*
	Dense	*2 ± 0*	***<u>24 ± 8</u>***	*10*	*12*

Table 51: A comparison of the success rates obtained on No Crash Benchmark for the CIL model, CILRS with ResNet34 from papers and CILRS with ResNet34 and ResNet50V2 proposed in this thesis. For the first two, mean and standard deviation are reported for three run of the benchmark. In bold and underlined there are the highest success rates for each scenario.

5.3.4 RESULTS ANALYSIS

Clearly the Table 51 states that CILRS50 improves the results of CILRS and CILRS34 on Training conditions. The same happen in New Town. In New Weather scenarios, on the other hand, it is CILRS34 that comes closest to the results of CILRS, and on the Regular task, together with CILRS50, they improve the success rate of CILRS. These considerations do not apply to New Weather&Town scenarios: for both, CILRS34 and CILRS50, we get a consistent worsening in the success rate, especially on the Empty task. Finally, it is noted that both CILRS34 and CILRS50 markedly improve the results of CIL on all scenarios.

Generalization in presence of dynamic objects

Also, for CILRS34 and CILRS50, we observe a generalization issues when the control policies have to deal with dynamic objects. Table 51, indeed, shows a large drop in performance as we change to tasks with more traffic, e.g., −53% and −38% from Empty to Dense traffic in NoCrash Training / New Weather&Town conditions respectively for CIRS34 and −55% and −34% for CILRS50. Therefore, the learned policies have a much harder time dealing robustly with a large number of vehicles and pedestrians. Success rate drops are shown in Figure 41.

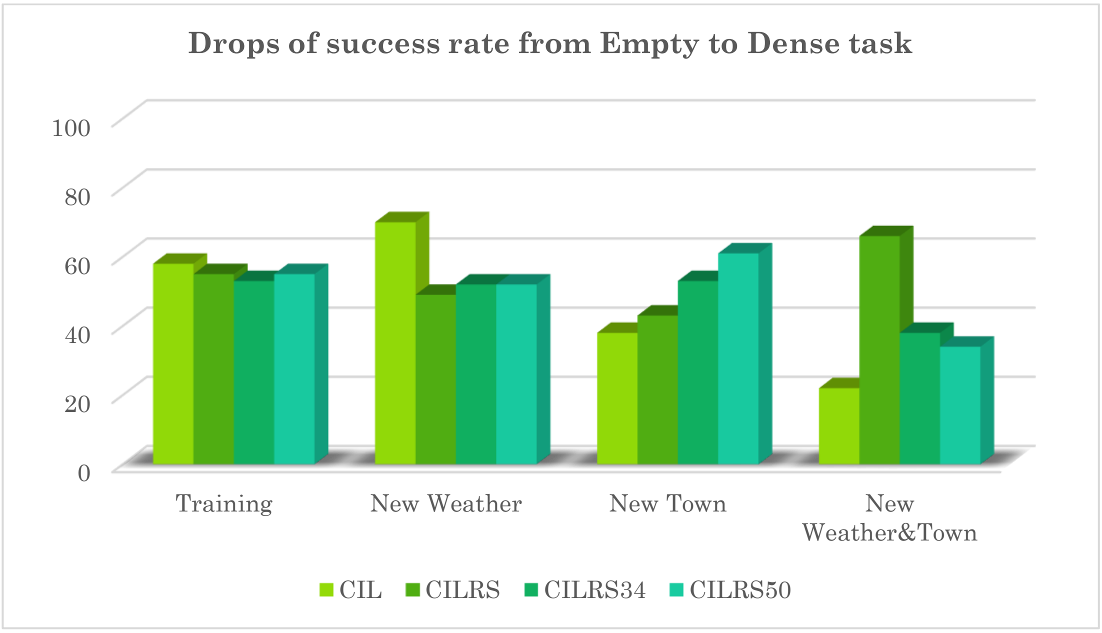

Figure 41: Drops of success rate from the Empty task to the Dense task is reported, for all scenarios and for the CI, CILRS, CILRS34 and CILRS34 architectures.

Generalization changing weather conditions

In less challenging scenarios, i.e. *training weather conditions*, CILRS50 is, in general, better than CIL and CILRS and CILRS34. This is not true for the *test weather conditions*, resulting in a worse success rate. Indeed, considering the Regular Traffic task, we get a drop in performance of −10% and −48% from training to New Town / New Weather&Town scenarios respectively for CILRS50 and of −5% and −35% for CILRS34. Observing Figure 42 the CILRS drops are of −6% and −27%, and you can understand how, at least for the most challenging scenarios, this architecture shows better generalization capabilities than the counterparts proposed in this thesis. We believed that this may be due to the lack of balancing of the data from the point of view of weather conditions.

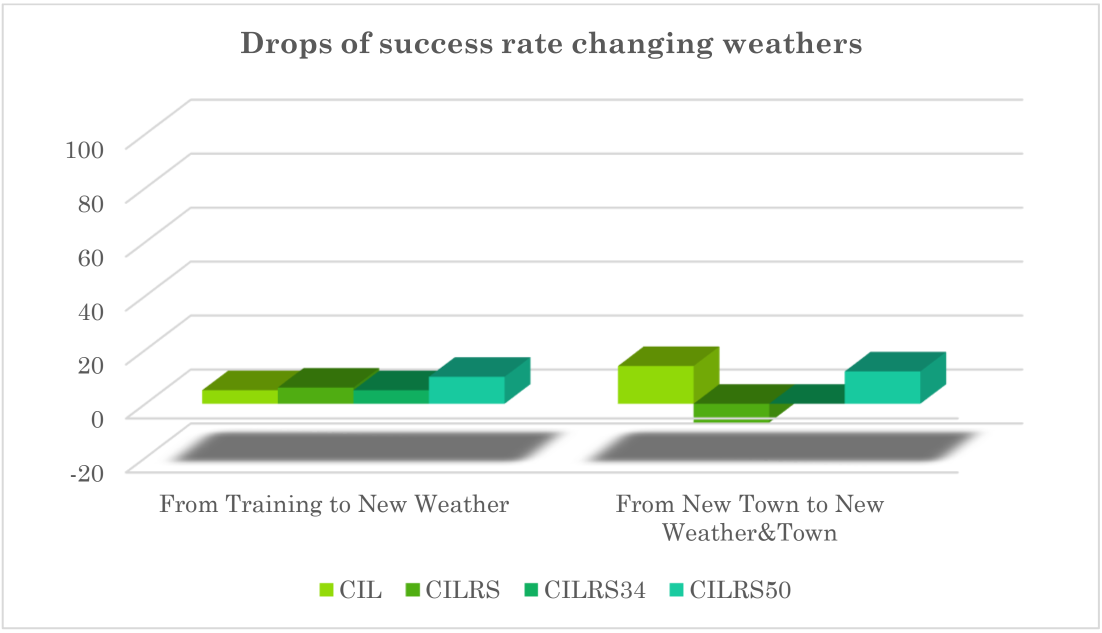

Figure 42: Drops of success rate on Regular task changing weather conditions. We report the drops when we pass from the Training scenario to those of New Weather and New Weather & Town for the CIL, CILRS, CILRS34 and CILRS50 architectures.

Generalization changing towns

In the *training map*, Town 1, CILRS34 and CILRS50 performs really good and sometimes they go beyond CILRS. When moving to the *test map*, Town 2, the results are closely related to the weather conditions. For CILRS34, the success rate drop is -35% from Training to New Town scenarios and -30% from New Weather to New Weather&Town scenarios, considering the Regular Traffic task. On the other hand, for CILRS50, the drops are -36% and -38% . This demonstrates that with both shallow and deeper architecture, is possible to obtain similar generalization capabilities on different maps. But these reductions are in any case greater than those obtained by CILRS, even more capable to generalize. This is shown in Figure 43.

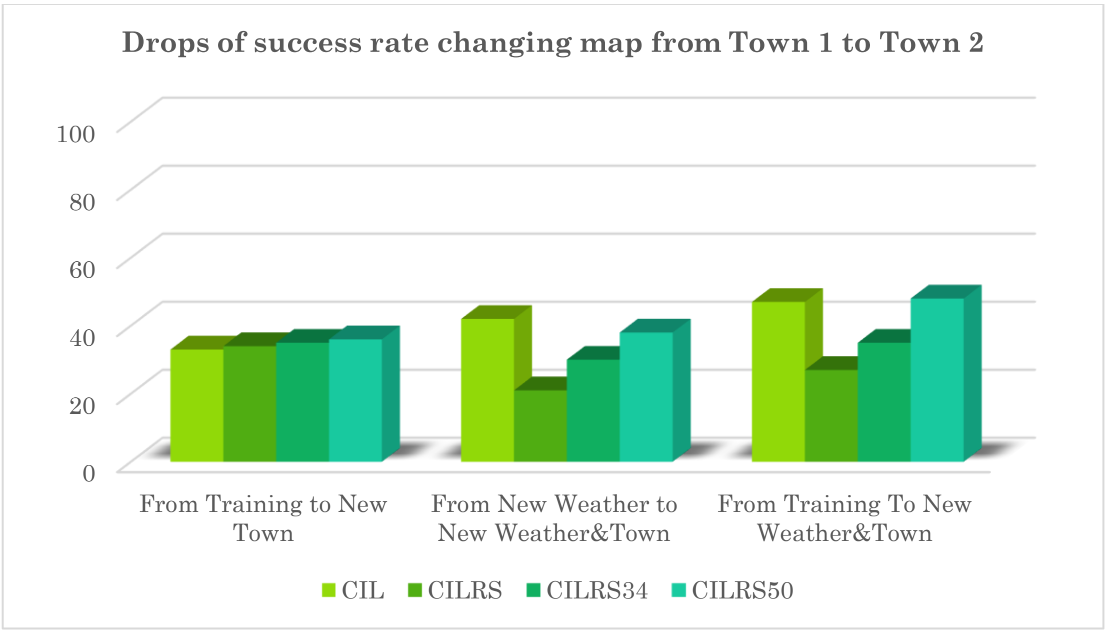

Figure 43: Drops of success rate on Regular task changing the map from Town 1 to Town 2. We report the drops when we pass from Training to New Town scenarios, from New Weather to New Weather & Town and from Training to New Weather & Town for CILR, CILRS, CILRS34 e CILRS50.

5.3.5 MAIN CAUSES OF FAILURE

In Table 52 we show the models with regard to their cause of failure. Values for CILRS come from [10].We specify the percentage of episodes that ended due to collisions and timeouts. As you can see, for all architectures, moving from the Empty task to the Dense task increases the number of collisions against vehicles and pedestrians. Moreover, it should be noted that the main cause of failure for CILRS are collisions with vehicles, while for CILRS34 and CILRS50 in the New Town and New Weather&Town scenarios the failures highlighted in section 5.3.4 are mainly due to ***timeouts***. These can occur for different reasons, from the failure to complete the task in time up to the difficulty in restarting once the controller has stopped ***(Inertia Problem)***.

TASK	*METRIC*	*TRAINING*			*NEW WEATHERS*		
		CILRS	*CILRS34*	*CILRS50*	*CILRS*	*CILRS34*	*CILRS50*
EMPTY	***Success Rate***	***97***	***96***	***99***	***99***	***92***	***86***
	Col. to Pedestrian	/	/	/	/	/	/
	Col. to Vehicle	/	/	/	/	/	/
	Col. to Other	*1*	*1*	*0*	*0*	*2*	*2*
	Timeout	*2*	*3*	*1*	*1*	*6*	*12*
REGULAR	***Success Rate***	***83***	***83***	***88***	***77***	***78***	***78***
	Col. to Pedestrian	*4*	*9*	*6*	*2*	*8*	*8*
	Col. to Vehicle	*8*	*1*	*0*	*17*	*0*	*2*
	Col. to Other	*5*	*3*	*0*	*3*	*6*	*6*
	Timeout	*0*	*4*	*6*	*1*	*8*	*6*
DENSE	***Success Rate***	***42***	***43***	***44***	***47***	***40***	***34***
	Col. to Pedestrian	*22*	*20*	*30*	*12*	*24*	*26*
	Col. to Vehicle	*21*	*14*	*15*	*26*	*8*	*8*
	Col. to Other	*12*	*15*	*7*	*11*	*16*	*20*
	Timeout	*3*	*8*	*4*	*4*	*12*	*12*
TASK	***METRIC***	***NEW TOWN***			***NEW WEATHERS&TOWN***		
		CILRS	***CILRS34***	***CILRS50***	***CILRS***	***CILRS34***	***CILRS50***
EMPTY	***Success Rate***	***66***	***69***	***78***	***91***	***48***	***46***
	Col. to Pedestrian	/	/	/	/	/	/
	Col. to Vehicle	/	/	/	/	/	/
	Col. to Other	*21*	*8*	*8*	*5*	*8*	*6*
	Timeout	*13*	*23*	*14*	*4*	*44*	*48*
REGULAR	***Success Rate***	***50***	***48***	***52***	***56***	***48***	***40***
	Col. to Pedestrian	*8*	*12*	*21*	*7*	*2*	*22*
	Col. to Vehicle	*9*	*7*	*3*	*23*	*0*	*0*
	Col. to Other	*22*	*12*	*7*	*8*	*10*	*8*
	Timeout	*12*	*21*	*17*	*6*	*40*	*30*

DENSE	***Success Rate***	***23***	***16***	***17***	***25***	***10***	***12***
	Col. to Pedestrian	*15*	*17*	*25*	*14*	*6*	*24*
	Col. to Vehicle	*39*	*24*	*16*	*37*	*30*	*20*
	Col. to Other	*17*	*21*	*21*	*8*	*14*	*12*
	Timeout	*5*	*22*	*21*	*17*	*40*	*32*

Table 52: Success rate and the main causes of failure for CILRS, CILRS34 and CILRS50 due to collisions or to the timeouts. Each cell reports the average of the success rates for a certain scenario and task. For each task the sum in a scenario is 100.

For the most challenging task, Dense traffic in New Weather Town scenario, in Figure 44 we go into detail. Here it is even clearer how the higher number of vehicle collisions for CILRS is offset by the higher incidence of timeouts for CILRS34 and CILRS50.

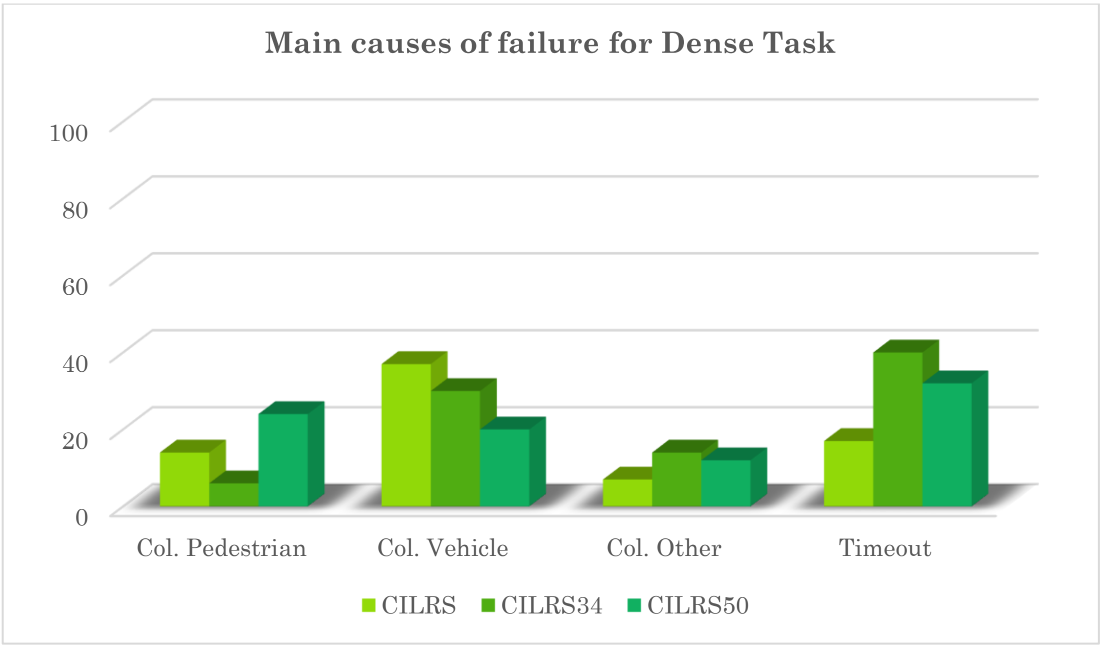

Figure 44: The success rates and the main causes of failure are reported for the Dense task and the New Weather & Town scenarios, comparing those obtained by CIL, CILRS, CILRS34 and CILRS50.

Chapter 6.

CONCLUSIONS AND FUTURE WORKS

6.1 CONCLUSIONS

With this work, the scientific literature around Conditional Imitation Learning is further enriched. The use of multimodality in input to the perception module, already experimented by [9] on the CIL architecture, and tested on ***CoRL2017 Benchmark***, is extended to the deeper CILRS, for which in [5] only the of RGB input modality is explored, evaluating the most reliable ***NoCrash Benchmark***.

Increasing the depth of the perception backbone could prove to be a double-edged sword: on the one hand, it can help to increase attention to scene detail, and on the other hand, it can over-specialize on training data, reducing generalization capability in new scenarios.

We have also faced some problems that remain open. First of all, the amount of dynamic objects in the scene directly hurts all policy learning methods. Second, the self-supervised nature of ***Behavior Cloning*** enables it to scale to large datasets of demonstrations, but with diminishing or worse returns due to driving-specific dataset biases that create ***Causal Confusion*** (e.g., the ***Inertia Problem***). Third, the poor correlation between offline evaluation and online driving performance, which makes difficult to evaluate architectures. Correlation can be affected by the sampling order and Open-loop metric used. In response to this, a more robust and standardized validation procedure is needed, based on

Open-loop and Closed-loop metrics, which allows the identification of the best policies.

6.2 FUTURE WORKS

Controllability. The set of commands used to implement ***Conditional Imitation Learning*** is quite limited, and this is due to the simplicity of the simulated environment available. However, it would be interesting to extend this idea to more complex scenarios, incorporating e.g. more complex intersections or lane change behaviour.

Evaluation. Currently evaluating an Imitation Learning based system is very complicated due to the discrepancy between Open-loop and Closed-loop metrics. Therefore, the development of a standardized validation methodology that allows to accurately provide a qualification of the real driving performance of a system could help.

Learning. It is observed that the current diversification and augmentation policies used to acquire data are not sufficient. A policy ***"correction"*** step may therefore be necessary using ***On-policy learning*** methodologies, i.e. the acquisition of new data using the learner under the supervision of the expert or applying Online learning through ***Reinforcement Learning***.

Limitations. Pure driver imitation produces end-to-end driving agents that lack knowledge of driving objectives such as traffic rules and safety. By adding traffic and safety rules to the objective function, it would be possible to obtain models that generalize better. Also, a different training strategy that detects incorrect causality from the data could be very beneficial for this type of training.

ACKNOWLEDGMENT

INDEX OF FIGURES

INDEX OF TABLES

BIBLIOGRAPHY

[1] Synopsys, “What is an Autonomous Car?,” [Online]. Available: https://www.synopsys.com/automotive/what-is-autonomous-car.html.

[2] A. Tampuu, M. Semikin, N. Muhammad, D. Fishman and T. Matiisen, “A Survey of End-to-End Driving: Architectures and Training Methods,” *IEEE Transactions on Neural Networks and Learning Systems,* 2020.

[3] P. Norvig and S. J. Russel, Artificial Intelligence, A Modern Approach, University of Michigan Press, 2003.

[4] Z. Zeyu and Z. Huijing, “A Survey of Deep RL and IL for Autonomous Driving Policy Learning,” 2021.

[5] F. Codevilla, E. Santana, A. M. López and A. Gaidon, “Exploring the Limitations of Behavior Cloning for Autonomous Driving,” 2019.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *1st Conference on Robot Learning (CoRL)*, 2017.

[7] F. Codevilla, M. Müller, A. López, V. Koltun and A. Dosovitskiy, “End-to-end Driving via Conditional Imitation Learning,” in *International Conference on Robotics and Automation (ICRA)*, 2018.

[8] F. Codevilla, A. M. López, V. Koltun and A. Dosovitskiy, "On Offline Evaluation of Vision-based Driving Models," in *ECCV 2018 conference*, 2018.

[9] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu and A. M. López, "Multimodal End-to-End Autonomous Driving," in *EEE Transactions on Intelligent Transportation Systems 2020*, 2020.

[10] F. Codevilla, "On Building End-to-End Driving Models Through Imitation Learning," Univeritat Autònoma de Barcelona, 2019.

[11] D. Chen, B. Zhou, V. Koltun and P. Krähenbühl, "Learning by Cheating," in *CoRL2019*, 2019.

[12] S. Ross, G. J. Gordon and A. J. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, 2010.

[13] G. Brain, "TensorFlow," Google Brain , 2015. [Online]. Available: https://www.tensorflow.org/.

[14] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2015.

[15] K. He, X. Zhang, S. Ren and J. Sun, "Identity Mappings in Deep Residual Networks," 2016.

[16] K. He, X. Zhang, S. Ren and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” 2015.

[17] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[18] K. He, X. Zhang, S. Ren and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” 2015.

[19] U. Saxena, “Automold--Road-Augmentation-Library,” 2018. [Online]. Available: https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library. [Accessed 10 2021].

[20] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference for Learning Representations (ICLR)*, San Diego, 2015.