UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Data Science Curriculum

Romet Aidla

Comparing Output Modalities in End-to-End Driving

Master's Thesis (15 ECTS)

Supervisors: Tambet Matiisen, MSc

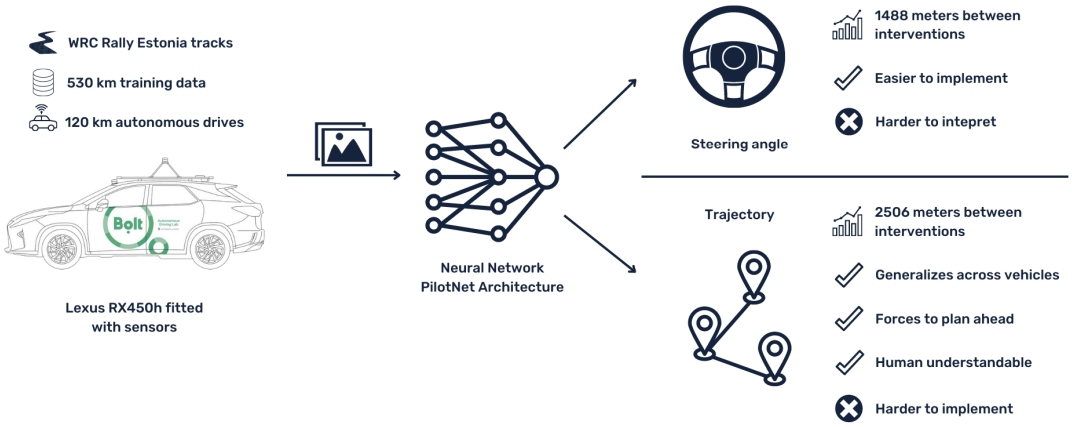
Ardi Tampuu, PhD

Tartu 2022

Comparing Output Modalities in End-to-End Driving

Abstract:
Self-driving car technology has made significant steps in the last ten years with the advancements in neural networks. The first autonomous vehicles are driving in San Francisco and Beijing. One of the promising approaches is end-to-end driving, where a neural network transforms an input image from a camera to output commands to control the vehicle. The most common output modalities are steering angle and trajectory. Both have been extensively benchmarked but not compared in similar settings. Metrics are usually calculated off-policy using a separated test dataset or on-policy using a simulator, but these have proven to correlate weakly with real-life performance. In this thesis, the comparison is made using an autonomous vehicle driving on WRC Rally Estonia tracks. The results show that the trajectory prediction approach is better at road positioning and recovering from non-ideal trajectories, which results in fewer situations where the safety driver has to take over.

Visual abstract:



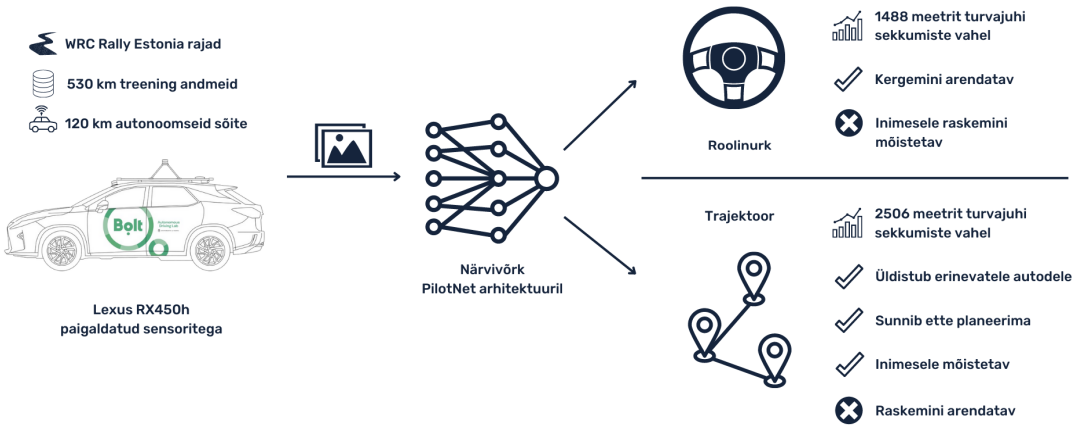
Keywords:
Computer Vision, artificial neural networks, autonomous vehicles, end-to-end-driving, model evaluation

CERCS:
P176 Artificial intelligence
T125 Automation, robotics, control engineering

Närvivõrgu väljundite võrdlus autonoomse sõiduki juhtimises

Lühikokkuvõte:
Isejuhtivate autode tehnoloogia on teinud viimase kümne aasta jooksul suuri edusamme seoses tehisnärvivõrkude arenguga. Esimesed autonoomsed sõidukid liiklevad San Francisco ja Pekingi tänavatel. Üks lootustandvaid lähenemisi on täielikult närvivõrkudel põhinev juhtimine, kus tehisnärvivõrk saab sisendina kaamera pildi ja väljastab käskluse sõiduki juhtimiseks. Levinud väljundid on roolinurk ja trajektoor. Mõlemaid on eraldi põhjalikult analüüsitud, kuid pole omavahel võrreldud sarnases olukorras. Juhtimise edukust väljendavad mõõdikud arvutatakse tavaliselt eraldatud testandmestikul või kasutades simulaatorit, aga on tõestatud nende nõrk korreleerumine tulemustega päris autoga sõites. Selles töös on võrdlus tehtud sõites autonoomse autoga WRC Rally Estonia kiiruskatsetel. Tulemused näitavad, et trajektoori ennustav lahendus on parema sõidujoonega ja taastumisega ideaalsest trajektoorist kõrvalekaldumisel, mille tulemuseks on vähem olukordi, kus ohutusjuht peab kontrolli üle võtma.

Visuaalne kokkuvõte:



Võtmesõnad:
Masinnägemine, tehisnärvivõrgud, autonoomsed sõidukid, otsast lõpuni isejuhtimine, mudeli hindamine

CERCS:
P176 Tehisintellekt
T125 Automatiseerimine, robootika, juhtimistehnika

Contents

1 Introduction

The first recorded case of a driverless car was already in 1925[1] when a vehicle called American Wonder drove down the streets of New York. This car was controlled over the radio by the car following it. Unfortunately, this experiment ended up with a crash into another vehicle and it has been a rough journey from there. There have been attempts to use smart infrastructure to make cars autonomous. In 1970, Citroën DS was made to steer automatically at speeds up to 130km/h using magnetic cables installed on the road[2]. The first vision-based system Navlab 1 [JPKA95] was introduced in 1986. It did not require any changes to the existing infrastructure when doing the first autonomous drive in 1988 with speeds up to 20 mph. In 1995 its successor Navlab 5 drove 2850 miles across the USA with 98% autonomy.

An artificial neural network was first used to control a car in 1988 by a system called ALVINN [Pom88]. This work can be considered the first self-driving system using an end-to-end methodology as a fully connected neural network was trained to predict vehicle turn radius directly from road images. It drove 90 consecutive miles with speeds up to 70 mph. There were additional efforts to build autonomous vehicles, but as hardware was slow to run neural networks, there was little progress. DARPA Challenge in 2004 and DARPA Urban Challenge in 2007 caused new interest in self-driving cars, but these solutions were handcrafted and did not use any learned driving approaches. Only on the arrival of GPUs and with the proliferation of data did neural networks become popular again, which has caused new interest in end-to-end approaches. Nvidia advanced the ideas from ALVINN and DAVE [MBC+05] by introducing research system PilotNet in 2016 [BdTD+16]. A significant amount of research has been done in the end-to-end driving field [TSM+20] since and data-driven approaches are one of the most promising today [HHBK21] [JPGO21].

There is a wide range of inputs used for self-driving, with monocular camera image and LiDAR point-cloud being the most popular. These are often also combined to achieve higher performance and better generalization. There is also a wide choice of output modalities to choose from. Most end-to-end models predict direct control like steering angle and vehicle speed [BdTD+16] [CMD+18] or trajectory waypoints [BCD+20] [HSG+20] the vehicle should follow. A wide range of work is done using different output modalities, but testing is done primarily using a simulator or toy cars. However, transferring models from a simulator to real-life is still an open research problem.

This thesis aims to understand the weaknesses and strengths of the two most common output modalities: steering angle and trajectory. Does predicting trajectory force a model to plan ahead? Which one performs better in crossroads or easier straight sections? What are failure cases for each modality? The self-driving field is vast, ranging from locations

[1] https://www.discovermagazine.com/technology/the-driverless-car-era-began-more-than-90-years-ago

[2] https://www.youtube.com/watch?v=MwdjM2Yx3gU

like gravel roads, city streets, and highways to autonomy levels from driver assistance (Level 2) to full self-driving (Level 5). This study is limited to road-following on Estonian gravel roads. It is a single agent non-dynamic environment. If there are other vehicles or pedestrians, the safety driver takes over the control of the car. Only the vehicle's lateral control (steering) is considered in this study. Longitudinal control (velocity) is outside the scope and is controlled using the speed from the trajectory recorded by the human expert on the same track.

A dataset of over 500 km of human driving on WRC Rally Estonia tracks was used to train models based on Nvidia Pilotnet architecture [BdTD+16] using the supervised methodology. RGB camera images were used as input and steering angle or trajectory as output modality. As these roads contain several crossroads, where the intended driving direction can be ambiguous, network architecture was extended with conditional branches [CMD+18]. The turn signal was used to indicate which road the model should choose. Models were evaluated driving autonomously around 120 km on the rally track near Elva. Driving metrics were calculated from these drives to compare model performances.

The thesis structure is as follows. Section 2 gives the background of end-to-end driving and related research. The reader is expected to be familiar with the main machine and deep learning concepts. However, field-specific concepts like end-to-end driving and conditional imitation learning are introduced to help the reader to understand this study. Next, two central problems in end-to-end autonomous driving are introduced: handling navigational command and countering distribution shift. The section finishes by describing the need for explainability in self-driving and introduces the VisualBackProp visualization technique. Section 3 explains the methods used in the study. First, the Rally Estonia Dataset is described and how it was collected. An overview of network architecture and the training process methodology is detailed. The section finishes with model selection and evaluation procedure. Section 4 reports the results of the experiments conducted. Off- and on-policy metrics are given for chosen models, and prominent failure cases are analyzed. Finally, Section 6 summarizes the work and key findings, followed by ideas for further research in Section 7.

2 Background

This section describes past studies and gives background information needed to understand end-to-end driving. Conditional Imitation Learning is introduced as a solution for handling crossroads. The distribution shift problem is described with possible solutions, followed introduction of VisualBackProp for explaining neural network behavior.

2.1 End-to-End Driving

Autonomous driving aims to find a mapping function from sensory input to actions controlling the vehicle. Input can be a sequence of RGB camera images, LiDAR point clouds, or other high-dimensional data helpful in driving. Action is usually lateral control like steering angle and longitudinal control like throttle and braking. Current autonomous driving solutions can be divided into roughly two different paradigms by how this mapping function is created.

The modular approach is an expansion of the sense-plan-act approach [Sie03] from the robotics field, where functionality is developed in interconnected sub-modules. These modules are usually perception, localization, planning and control. The advantage of this approach is that it is interpretable as the interfaces between modules are human-designed. In case of failure or unexpected behavior, a problematic module can be identified and resolved in isolation. The downside of this approach is that building this pipeline is very costly and after many years of work, full autonomy is still restricted to pre-mapped areas. This method was used successfully by teams in DARPA Urban Challenge [UAB+08] [MBB+08] and is used by companies like Waymo, Cruise, and Zoox.

The end-to-end approach uses one monolithic function to transform sensory input into driving commands. This mapping function is usually implemented as a neural network (Figure 1). The network can learn the driving task as a whole, as there are no human-designed intermediate representations. The advantage of end-to-end learning is that labels are relatively easy to collect, and the system can be trained directly for the end goal of driving itself. The disadvantage of end-to-end learning is that it does not generalize as well as the modular approach. It is also harder to interpret model behavior and understand whether all failure cases are covered.

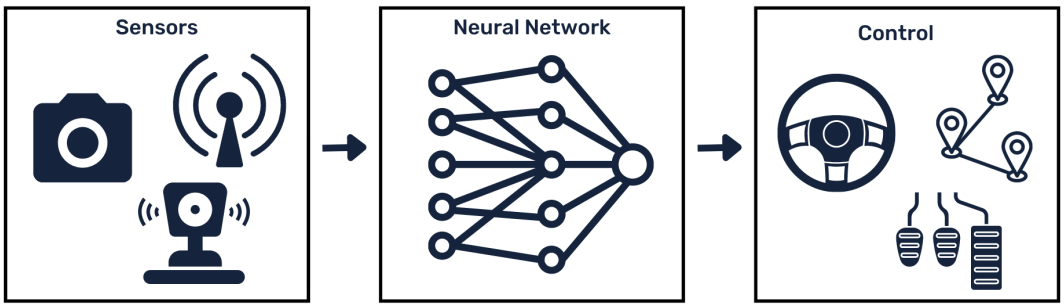


Figure 1. The end-to-end approach uses an artificial neural network to map sensory input directly to the vehicle controls.

There is also a hybrid of the modular and end-to-end approaches called direct perception [CSKX15a] [SSG18] [WB20]. The model is still trained end-to-end but is divided into separate modules, usually decoupling perception from planning and control. There is intermediate representation, which can be visual abstraction like semantic segmentation, depth and optical flow [ZKK19] [BCP+20], affordances [CSKX15b] or driving policy abstraction [MDGK18]. The visual abstraction approach is also by Wayve [HSG+20], Tesla AutoPilot, and Comma OpenPilot. In direct perception, the usual prediction target is affordances [CSKX15b], which can be relative position in driving lanes, distances toward other surrounding cars, or other driving indicators.

The most common approaches for training the neural network for autonomous driving are imitation and reinforcement learning. Imitation learning uses a data-driven approach and learns to mimic human behavior by observing an expert in action or given a dataset of demonstrations. Imitation learning rolls out the sequence of environment states by the action predicted by the model. Behavior cloning [Pom88] [BdTD+16] [BCD+20] [CMD+18] [LMC+19] simplifies this and rolls out states using action used by the expert, which means that finite dataset can be used. The model can be trained using the supervised learning methodology successfully used for other computer vision tasks. A dataset of human driving is collected containing sensor data like camera images and matching vehicle controls like steering angle and velocity. The neural network learns to predict vehicle controls given environment state as sensor data. The downside of this simplification is that the model learns only to act in the states in the dataset and does not learn to behave in a state that would be reached by executing non-perfect action. This problem is called distribution shift and is discussed further in Section 2.3. Reinforcement learning teaches driving policy by interacting with the environment either in real life [KHJ+19] or in a simulator [WAS+22] [AWG+21]. This approach is currently less used as a failure on the roads is costly and transfer from a simulator to real life is an open problem [SFMP21].

There is a wide range of output modalities used successfully in end-to-end driving. The original PilotNet [BdTD+16] solution predicts steering angle and the follow up PilotNet Experiments [BCD+20] trajectory. Models are validated on the physical car, with some of the benefits of trajectory prediction discussed qualitatively. However, neither work describes any experimental results that would make these solutions comparable. [CMD+18] predicts steering angle and acceleration, experiments are done in Carla simulator and in real-life using a 1/5 scale robotic truck. A trajectory is also predicted in [HSG+20] and [LMC+19], however, it is impossible to compare the results as different environments and network architectures are used. [TWB20] offers the Bézier curve as a novel output modality and compares it to steering angle and trajectory waypoints. The study is done in the simulator and with a 1/10 scale model car in a circuit racing setting. This environment does not have crossroads, which is an important scenario for every autonomous vehicle and could influence the choice of the most suitable output modality. Their results show that predicting trajectory waypoints gives considerably better results than predicting steering angle. A trajectory can also be represented as a motion plan, a sequence of future steering angles and accelerations [ZWL+21]. Driving has inherent uncertainty and there are probabilistic approaches like predicting a mixture of Gaussians [ARKR19].

An end-to-end approach with behavior cloning is used in this work. As testing all output modalities would be unfeasible, the most common options steering angle and trajectory waypoints are used as output modalities.

2.2 Conditional Imitation Learning

Cloning human behavior makes a vehicle follow the road, but the problem arises when the vehicle arrives at a crossroad. We need to inform the model somehow about which direction to take. In a fully-autonomous system, the navigational command would be sent out by a global planner, which plans the route on a map. The turn signal is used for our current implementation to simulate the correct direction. Two possible architectures were proposed for handling this command [CMD+18]. The first technique concatenates the navigation command with image features, while the second option uses the command to switch between different branches of the neural network (Figure 2). This article showed that branched architecture is potentially better and has been successfully used in other works [BCD+20] [HBWZ17]. [HSG+20] extended the first technique by concatenating route command with the output of every layer in the control module to avoid vanishing signal and showed that it is possible to get good results even in an urban driving scenario (Figure 3).

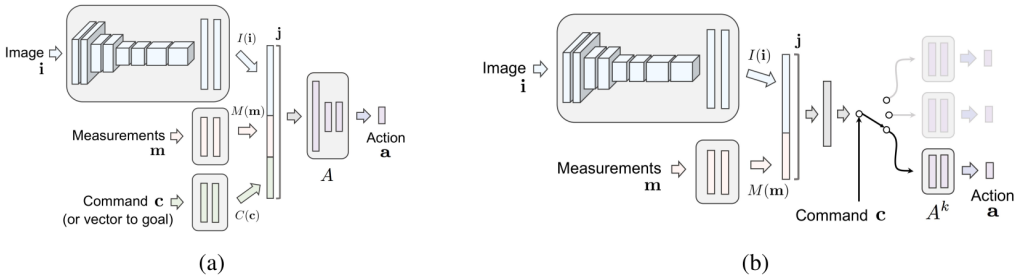


Figure 2. Two possible architectures for conditional imitation learning. (a) concatenates image features with the command. (b) uses command as a switch between different branches of the network. Image adapted from [CMD+18].

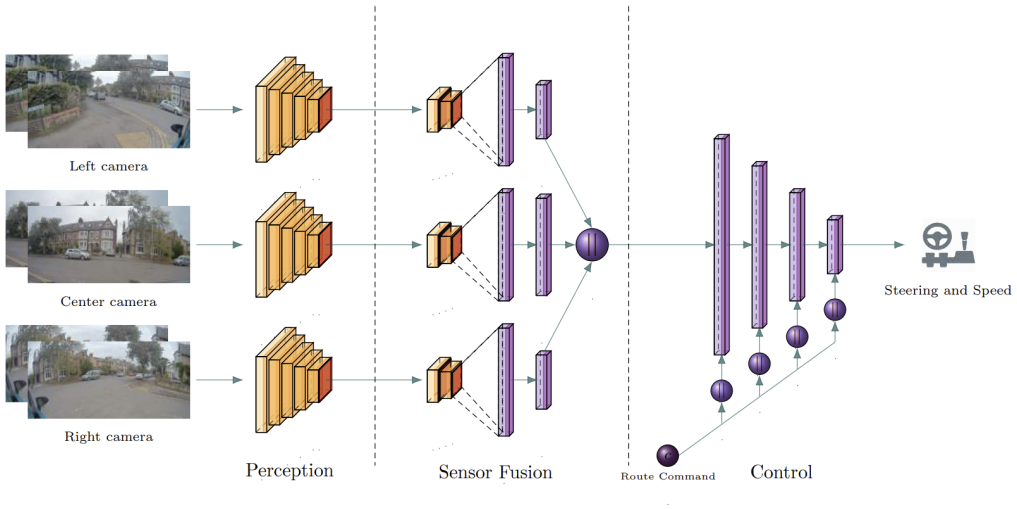


Figure 3. Urban Driving with Conditional Imitation Learning concatenates directional command with the output of every dense layer to avoid vanishing signal. Image adapted from [HSG+20].

Using a turn signal as a navigation command can be limited as it has only three states (straight, left, right), and some road crossings could have more choices. In addition, the meaning can be somewhat ambiguous as sometimes it is not clear which road is going straight and which one is turning left or right. To improve this, a simple roadmap with desired route [ARKR19] [CSU21] [SCR+20] or directional vector [BCD+20] can be used instead.

2.3 Distribution Shift

One of the main problems with behavior cloning is the distribution shift, where test data differs from training data. During training, the model is trained only using expert trajectories. Still, during driving, the vehicle can slowly drift from this trajectory and get into states it has not seen during the training, as the expert rarely deviates from the good trajectory. These mistakes will compound until the vehicle is in a state where the model will behave unexpectedly, in the worst case turning off the road.

2.3.1 Side Camera Augmentation

One of the most straightforward solutions for the model to learn to recover from mistakes and decrease the effect of distribution shift is to augment training data with additional images that simulate situations where the car shifts from the ideal trajectory or rotates from the direction of the road. Cameras positioned to the left and right in relation to the central camera can be used to show the perspective of the off-center shift. The left camera images imitate drifting to the left and respectively, the right camera images imitate drifting to the right. As the steering angle used by the expert aim to keep the car driving on the current trajectory, correct steering angles must be calculated for the side camera perspectives to recover the vehicle towards the good trajectory by steering right or left. Precise rotations are simulated by viewpoint transformation, which requires 3D scene knowledge and is out of the scope of this work. Side cameras are only used during collecting training data and not while driving with the trained model (Figure 4). During testing on the vehicle, only the central camera is used to predict the steering angle. This method was successfully used in [BdTD+16].

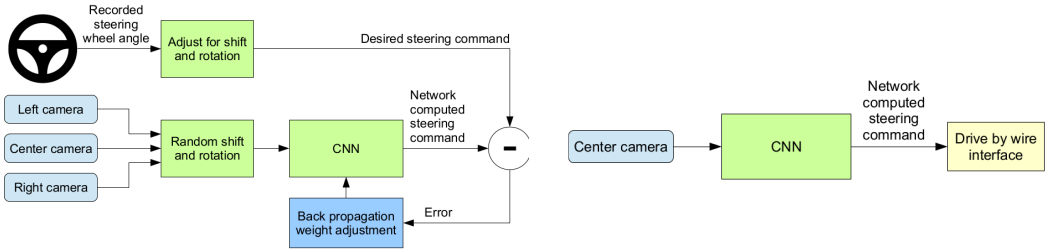


Figure 4. **Left**: during training, images recorded using side cameras are used to imitate drifting from the expert trajectory. **Right**: autonomous driving is done using only the central camera. Images adapted from [BdTD+16].

2.3.2 Data Balancing

A driving dataset is naturally unbalanced as most driving is near-straight, with considerably fewer sharp turns. Biases in a dataset can decrease model generalization ability. Datasets are usually balanced using upsampling frames with rarely occurring steering angles or downsampling the common ones. [HSG+20] balanced data weighting samples by steering angle and speed, proposing this was sufficient to fight the distribution shift problem without needing additional augmentation or data synthesis. First, data is split into bins by steering values. Bin edges are defined such that the product of the number of data in each bin and the width of that bin is equal for all bins. This balancing process is recursively applied to each bin w.r.t. speed. Samples with infrequent steering angle and speed will fall into wide bins and get significant weight, while driving straight samples will fall into narrow bins and get small weight decreasing their influence.

2.3.3 Other options not used

DAgger [RGB11] [PBOB+20] addresses this problem by collecting additional training data by driving the model. Humans label this new data to have optimal action in these situations, which in the case of self-driving would be corrective turning towards the center of the road. This new data is then merged with the existing dataset and used to train a new model that would recover better from a non-ideal trajectory. It can be tricky for humans to label the optimal actions. In the case of end-to-end driving, labeling steering angles for every frame is challenging and is not used in this work.

An additional option is introducing noise to the steering wheel and letting the human expert demonstrate recovering [CMD+18]. It is very tiring for humans to collect data this way and is not used in this work.

2.4 Explainability

Autonomous driving is a safety-critical task and understanding why models make certain decisions is critical. The most common approach in computer vision is gradient-based solutions [ZKL+16] [SDV+16], but these are not widespread in the end-to-end driving field as they lack pixel-level accuracy. Intermediate representations [ZKK19] and auxiliary outputs [HSG+20] can help to investigate why models fail but are outside of the scope of this work.

2.4.1 VisualBackProp

The main idea of VisualBackProp [BCC+16] is to find salient image regions that lead to high activations, as seen in Figure 5. They proved that this method finds the most important regions for steering by shifting the salient objects, which resulted in a linear

change in steering angle nearly as large as when shifting the entire image [BYC+17]. Shifting just the background pixels had a much smaller effect on the steering angle.

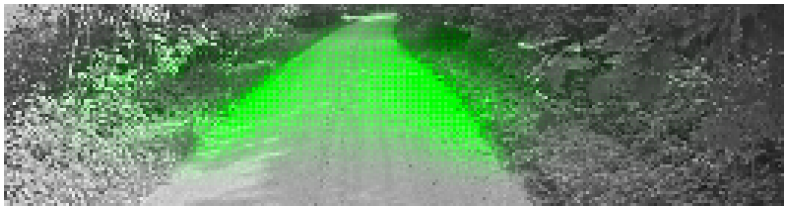


Figure 5. The VisualBackprop method can highlight salient objects of the input image.

The forward pass is done with the input image, and the feature maps' activations are averaged. The averaged map of the last layer is scaled up to the size of the previous layer using deconvolution. This scaled-up map is multiplied by the previous layer's activation map, resulting in an intermediate map. This process is repeated until input is reached and the last map is the size of the input image (Figure 6). The result can highlight the most important regions of the input image contributing the most to the prediction of the network. VisualBackProp will be used throughout this work to analyze the model's behavior, such as the main reasons for failures.

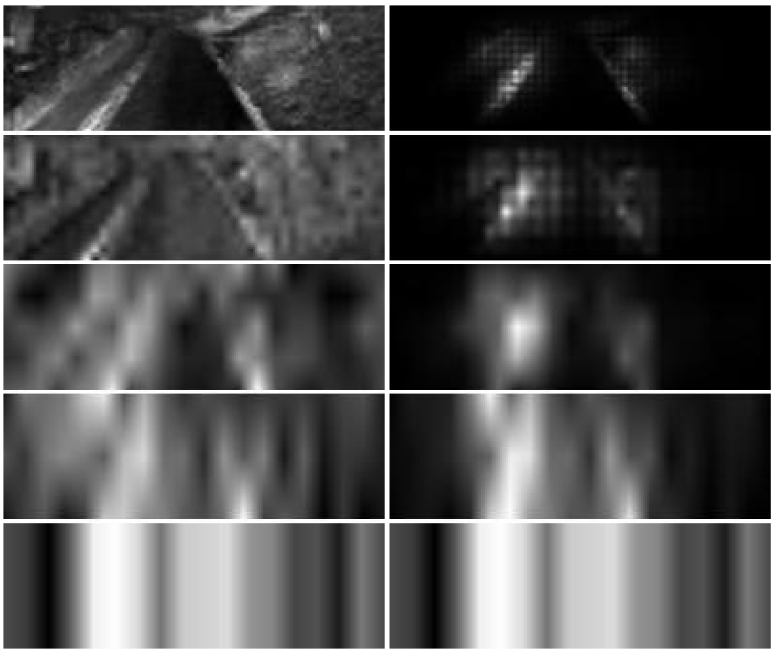


Figure 6. **Left**: Averaged feature maps for each level of the network. **Right**: Intermediate visualization mask for each level of the network. [BYC+17]

3 Methods

This section describes the methodology used to train models predicting different output modalities and evaluate these in the experiments. First, chosen output modalities are introduced. The vehicle platform is presented, followed by the data collection and preparation steps. The subsequent section describes network architecture and training methodology. Lastly, the experimental setup and evaluation criteria are defined.

3.1 Output Modalities

Steering angle and trajectory waypoints are used and compared in this work as output modalities as these are the most common approaches.

3.1.1 Steering Angle

The most straightforward solution is to predict the steering angle of the driving wheel as a single numerical value in either degrees or radians. This value can then be converted into the angle of the front wheels by multiplying with the vehicle-specific steering ratio constant. Sometimes the angle of the front wheels is predicted directly, but as these values are linearly dependent, it does not make a difference. The front wheel angle can then be fed into the vehicle's control system for achieving lateral control. The downside of predicting steering angle directly is that the model will be vehicle specific and not work in other vehicles. This work predicts the steering wheel angle as a single number in radians.

3.1.2 Trajectory Waypoints

A more complex solution is predicting waypoints the vehicle should follow on the road. This forces the model to learn long-term planning, which could improve driving ability thanks to looking ahead and reacting to challenging situations earlier. Predicting trajectory also has the benefit of decoupling planning from control, which means the same model can be used on different cars. It has been shown [BCD+20] that learning from human trajectories does not work well and that using a human-labeled center line as a learning target is better. Labeling center lines is significant work and is not used in this thesis.

There are different ways to represent trajectory. In this work, each waypoint is encoded as x- and y-offsets in relation to the viewing direction of the vehicle, as was done in the Observational Imitation Learning paper [LMC+19]. Z offset is ignored and is always set to zero. This makes the trajectory similar to Birds Eye View (BEV), as shown in Figure 7.

The waypoints' longitudinal part (x-offset) can be encoded as distance or time difference. The benefit of encoding it as a time difference is that model learns to control speed. In this work, the driving speed is not controlled by the model. To make predicting easier for the model, the x-offset is always after every 5 meters. Prediction of the x-offset is only learned as a generic solution is implemented to support distance- and time-based trajectories, but only distance-based trajectories are used in this work.

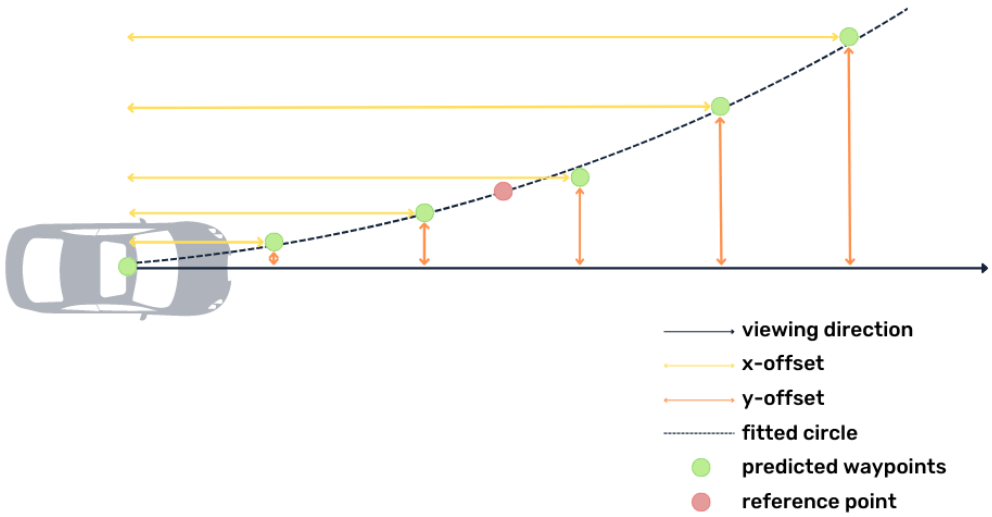


Figure 7. The trajectory is represented as x- and y-offset in relation to the viewing direction of the vehicle. A circle is fitted to the predicted waypoints and the vehicle is steered towards the reference point on this circle.

The model sees the world from the perspective of the camera used as the input sensor. The trajectory driven by the expert used for training is transformed into the same camera frame, which makes predicting easier for the model as it does not need to learn the spatial configuration of the vehicle. As the predicted trajectory is in the camera frame, it must be transformed back to the vehicle frame before it can be used to drive the vehicle. Transforming trajectory between vehicle and camera frames is a simple transformation as relative positions of cameras to the vehicle are known. As an added benefit, this makes it possible to train camera-agnostic models by training models using images from all mounted cameras. For every image, trajectories are transformed into a given camera frame. The model trained with different camera viewpoints will also be more robust to small changes in camera position. During this work, preliminary tests with camera-agnostic models were done and the results were promising, but this needs further research.

A vehicle cannot be controlled directly with a predicted trajectory, so the trajectory

must be transformed into direct steering and velocity controls. The most common controllers used in the field are Pure Pursuit [Cou92], Stanley Controller [HTMT07] and Model Predictive Control [Sni09]. Off-the-shelf Stanley Controller was tried, but it had problems following the predicted trajectory, and as this controller works only on-policy, these problems were hard to resolve. In a road-following use case, the vehicle is always at the beginning of the predicted trajectory, so these controllers are more complex than necessary. Furthermore, as these controllers accumulate errors, they cannot be used to calculate metrics off-policy, which is helpful for training and choosing a correct model to be deployed on the car.

A custom controller was implemented following ideas from Learning by Cheating article [CZKK19]. Instead of directly steering the vehicle toward one of the predicted waypoints, a circle is fitted onto a selected number of trajectory waypoints. The vehicle is steered towards a reference point on the curve to smooth out prediction errors in individual waypoints (Figure 7). The number of selected waypoints and reference point distance are hyper-parameters optimized with training data by minimizing loss. The best configuration found were three waypoints and a reference distance of 6.5 meters. It was also found that it is better to include the vehicle's current position in the set of waypoints.

3.2 Vehicle Platform

The vehicle used for collecting expert driving recording and testing models is Lexus RX 450h (Figure 8) fitted with sensors needed for autonomous driving. Four Sekonix cameras (3x SF3324, 1x SF3325) are used to record RGB images of the drive. These cameras are connected to the NVIDIA DRIVE AGX Xavier computer. AStuff Spectra computer is used to run models predicting steering angles and trajectories. NovAtel PwrPak7D GNSS device determines the location of the car using satellite positioning and corrections from nearby base stations. The system makes use of Real-Time Kinematic (RTK) system with ESTPOS base station network to achieve 5-10 cm positioning accuracy consistently.

The vehicle has additional sensors not used in this work. The short-range LiDAR Ouster OS1-128 helps to detect close-by and small obstacles around the vehicle. The long-range LiDAR Velodyne VLP-32C lidar helps to detect far-away obstacles, e.g. cars and pedestrians. Aptiv ESR 2.5 radar uses reflections of radio waves to estimate the distance to the other vehicles ahead. The Allied Vision Mako G-319C cameras keep an eye on traffic lights. The left camera is pointed forward to see the farther traffic lights. The right camera is pointed slightly to the right to see the closest traffic lights next to the stop line.

[3] https://adl.cs.ut.ee/lab/vehicle

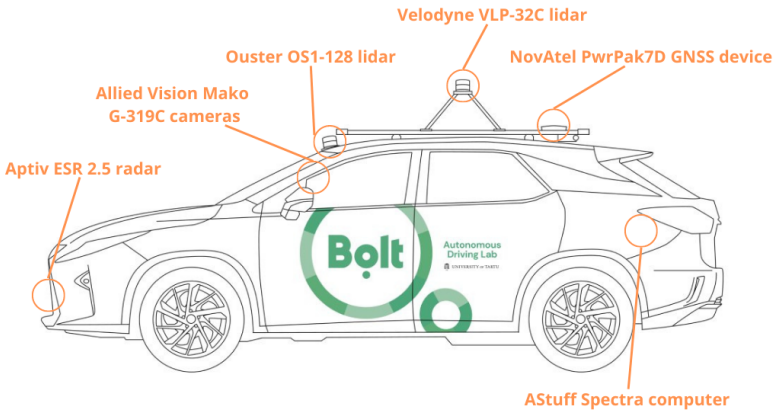


Figure 8. The Lexus RX 450h from the University of Tartu's Autonomous Driving Lab (ADL)[3]

3.3 Data

3.3.1 Data Collection

Between May 2021 and June 2022, driving data from WRC Rally Estonia 2020 and 2021 tracks was collected. Driving was only done in daylight, but weather conditions were very diverse, including heavy rain, deep snow, and sunshine. The road type was mostly narrow gravel roads, but there is a small section of two-way tarmac roads. The dataset was collected during all seasons, including winter, but only non-snow seasons were used in this work.

Three Sekonix SF3324 120-degree FOV cameras and one Sekonix SF3325 60-degree FOV camera were used to collect RGB images. LiDAR point clouds were collected using the Ouster OS1-128 sensor. Positional data were recorded using the NovAtel PwrPak7D-E2 GNSS device. All tracks were recorded three times, the first time in spring and early summer, the second time in autumn, and finally during the winter. Test track near Elva was recorded additionally for a fourth time to train overfitted models. The dataset contained 853 km of driving, from which the non-winter recording was 530 km.

3.3.2 Data Preparations

Models were trained only on images from centrally placed Sekonix SF3324 120-degree FOV camera. Winter recordings were not used as driving in the winter is more challenging and was out of scope for this work. The dataset was divided into training and validation sets, with the validation set roughly 21.8% of the whole dataset. Recording of the test track in Elva in spring and autumn 2021 and spring 2022 were included only in

the validation set except when training overfitted models, in which case the Elva track recordings from 2022 were moved into the training set. Driving on the test track is used to validate the model instead of using a separate test dataset.

Input RGB images were cropped to include only areas important for driving. Vertically only pixels between the vehicle's hood and horizon were included. Horizontally only around 90 degrees of view were used as a preliminary test with a wider crop showed that the side of the images is not essential for driving and does not contribute to better performance. The resulting image with size 258x66x3 was used as an input to the neural network. No additional prepossessing was done to the images.

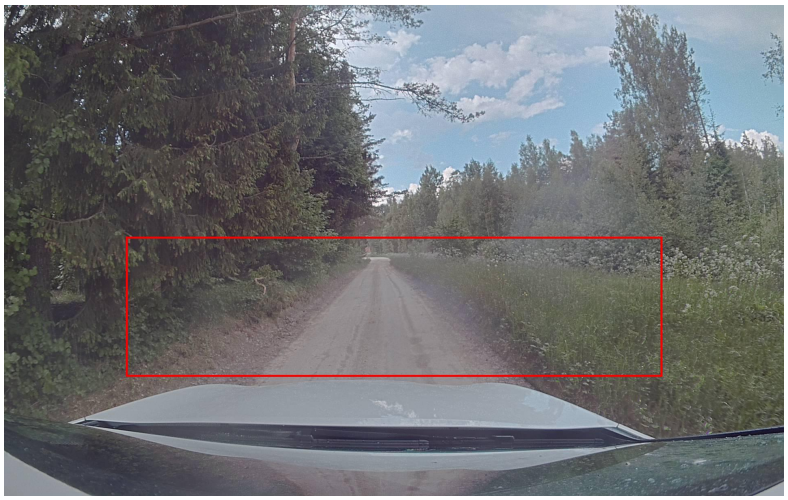


Figure 9. Input image. Cropped area are is 258x66 pixels and is marked with a red box.

The dataset contains a small amount of driving that is not good for training. For example, when a crossroad is missed, when another vehicle is encountered on a narrow road, or when the vehicle needs to be reversed. There can also be technical problems like GPS positioning being inaccurate. All drives were checked manually and 14.9% of frames were removed during the cleaning process. Off-policy metrics showed around 10% improvement resulting from the cleaning process. During the process, also some of the turn signal labels were improved. If the turn signal is not shown during a turn on a crossroad or the signaling time is not correct, these labels were improved in the dataset manually.

3.4 Network Architecture

This work aims to compare output modalities and not to find the best network architecture. Road following is a simple non-dynamic task and a simple model provides a better

understanding of the effects of using different prediction targets. Convolutional neural network architecture based on classical PilotNet architecture [BdTD+16] is used. The five convolutional layers extract image features and form the perception part of the network. The four fully connected layers are designed to work as a steering controller. This division is mainly logical as the network is trained end-to-end and there is no rigid boundary between which functions different parts of the network fulfill.

The control part of the network was extended using the approach described in Conditional Imitation Learning [CMD+18]. In addition to the general road following branch, separate network branches are created for turning left or right on the crossroad. Turn indicator is used to control which branch is used for the prediction. Architecture based on Urban Driving with Conditional Imitation Learning [HSG+20] was also tried, but testing it on the vehicle showed it to have inferior driving performance. It could be caused by only one central camera being used instead of the three cameras as in the paper.

Batch normalization is added after every layer except the last two dense layers, as this has been shown to give more stable training and better convergence [STIM18]. LeakyReLU activation is used instead of ReLU as it has been shown to have better performance [XWCL15]. Different variations of the models were tried, for example, using dropout regularisation. However, no significant performance benefit was noticed and as it is out of the scope of this work, no neural architecture search and ablation study was done.

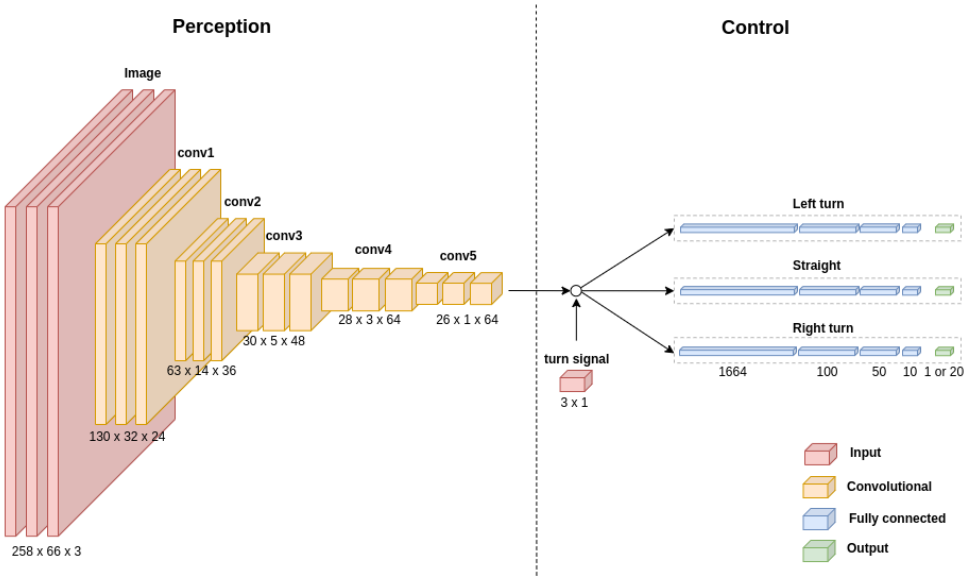


Figure 10. Network architecture is based on PilotNet and Conditional Imitation Learning.

The driving task would benefit from temporal context and recurrent network architecture would seem to be a great fit. Still, it has been shown to lead to causal confusion [dHJL19] and would need additional work to counter this effect as demonstrated in [HSG+20].

3.5 Training

3.5.1 Training Process

Each training batch has 512 frames, and each epoch has 1024 batches, which adds to 262144 data points used in every training cycle. The usual practice is to use all training data with each epoch. However, the downside is that training set size can change by either collecting new data or cleaning existing data, which changes epoch size and makes different training runs incomparable. Training is stopped after validation loss has not improved after 20 epochs. The model weights used for off- and on-policy evaluation are saved at the end of the epoch, which was the last one to improve the validation loss.

MAE loss is used instead of MSE as previous work has shown it has a better correlation with real-life performance [CSLG19]. The loss is the difference between predicted and actual steering angles for steering angle. For trajectory prediction, the loss is the distance of predicted and actual x- and y-offsets. Further waypoints contribute more to the loss as their lateral offsets usually have a more significant deviation from the viewing direction of the vehicle. Discounting the loss of farther waypoints by a discount factor of 0.2 did not show a clear advantage in the off-policy metrics and was not used. However, it could potentially improve the driving ability as shown in [HSG+20].

Models are trained using behavior cloning, which means models learn only from expert trajectories and are bad at recovering after drifting from the ideal trajectory. Both augmenting data with side cameras and data balancing were used to address this issue. The second problem is that while rally roads are relatively curvy, there is still only a limited amount of crossroads where navigation command is used. Left and right conditional branches were initialized with pre-trained weights to get a smoother drive.

3.5.2 Data Balancing

Data balancing was implemented using the data-sampling solution described following ideas from [HSG+20]. Data is split into twenty bins by steering values. The edges of the bins were found using the Limited-memory BFGS minimization algorithm by minimizing the difference in product of width and number of samples for each bin (Figure 11). For simplicity, bins were not balanced by speed, although there is a possibility of additional improvement as described in the original work. Each sample is assigned a weight that is proportional to its bin width. Bigger steering angles are oversampled as these are in sparse bins and driving straight samples are undersampled as these are in dense bins. The

resulting data distribution is more balanced but still not precisely normal distribution. Weight is equal for each sample in the same bin, but there are more samples of small steering angles within the same bin. The balancing solution could be improved by using more bins, but it is unknown whether it would improve driving performance.

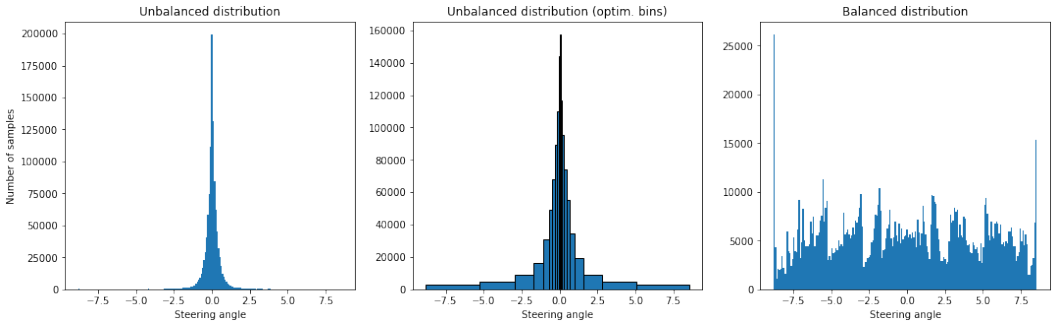


Figure 11. **Left**: Driving data is unbalanced as most samples are near-straight driving. **Middle**: Driving data distribution divided into bins used for balancing. Each bin's area (product of bin width and number of samples) is equal. **Right**: Big steering angles (sharp turns) are oversampled compared to small steering angles (driving straight), resulting in a more balanced distribution. All figures are sampled with the same number (1.09M) of data points as there are samples in the training dataset.

3.5.3 Augmentation Using Side Cameras

Using images from side cameras helps with recovering from a non-ideal trajectory. As prediction labels are in the frame of the center camera, then new labels had to be calculated for side cameras. For trajectory prediction, it is easier as trajectory can easily be transformed from one camera frame to others using camera location and viewing direction. There is no true method to convert the steering angle from between camera frames, but in this case, the trajectory controller was used to calculate the steering angle from the already transformed trajectory. This adds reliance on the accuracy of the used controller but should be a good enough approximation.

3.5.4 Initializing Conditional Branches

As there are not enough turns at the crossroads in the dataset, the left and right branches of the network's control module do not see enough driving data and remain quite uncertain in their predictions. A standard Pilotnet model with no conditional branches was trained using complete training data. Then conditional model with left and right turn branches was initialized with the weights from the pre-trained model with all conditional branches

having the same weights. This model was then trained using conditional imitation learning, meaning all conditional branches were fine-tuned for their specific purpose.

3.5.5 Model selection

Testing models in real-life on a vehicle is time-consuming and it is impossible to test each model extensively. For each output modality, three models were trained with techniques described above: training with data balancing, augmentation using side cameras and initializing conditional branches. Additionally, a baseline model was trained without using these techniques. These four candidate models were tested using short drives and the best model was chosen for each output modality to be evaluated in more extended experiments. The main criteria for selection were a low number of interventions and predictable behavior judged qualitatively by the safety driver.

After the correct model training procedure was determined, two additional models were trained for each output modality. The second version of the models were trained using the same configuration used to train the candidate model to validate the stability of the training pipeline and driving performance. The third version of the models were trained to evaluate generalization performance to a new track, with the evaluation track recorded in the 2022 spring included in the training set. As these models are exposed to the roads on the evaluation track, they are called "overfit", although every drive exposes models to new conditions. This process resulted in six models, three for each modality.

3.6 Evaluation

The two main evaluation methods for autonomous vehicles are off- and on-policy evaluation (Figure 12). The off-policy evaluation uses observations from the existing dataset to make predictions. The accuracy of this prediction is calculated by comparing it to the ground truth showing the model's predictive ability. There is no driving using the trained policy, hence the name off-policy. This methodology is often called open loop evaluation as the policy does not interact with the environment. The observations are not generated from taken actions, so the feedback loop is not closed. Off-policy accuracy is a bad predictor of actual driving performance as models with the highest off-policy metrics are not always the best at the actual driving [CLKD18]. It is impossible to test all models using on-policy evaluation, so off-policy evaluation is used as a cheap heuristic to select the promising models deployed on the vehicle.

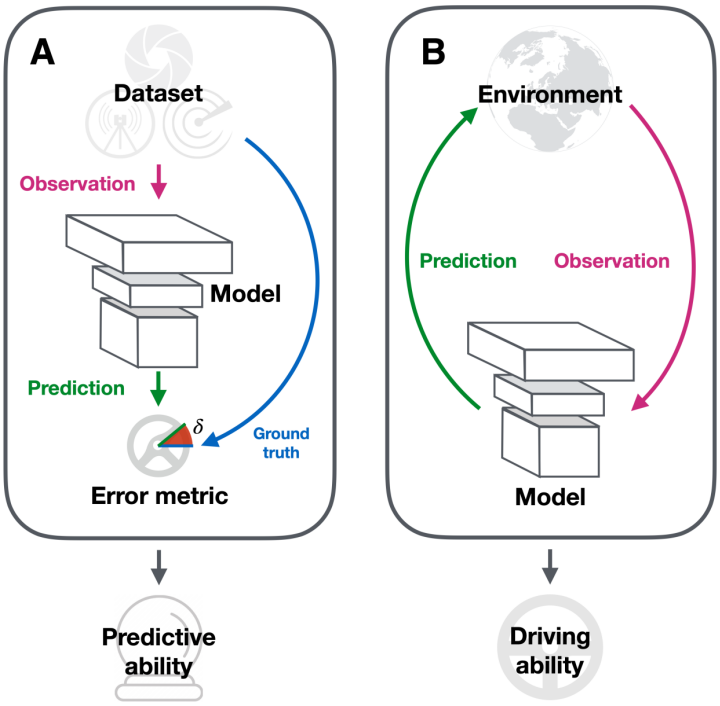


Figure 12. Off and on-policy evaluation. A: Off-policy evaluation uses a part of the original dataset to evaluate the model. The similarity between predicted and ground truth values is measured. No actual driving is done using the trained policy. B: On-policy evaluation deploys the model in an environment. The model's predictions are used as driving actions. The resulting driving behavior is observed and quantified. The figure and caption are adapted from [TSM+20].

On-policy evaluation deploys a trained model into the real world and the vehicle is driven using actions predicted by the model, hence the name on-policy. The last action chosen by the policy changes the environment state and new observation from the environment is used to predict the following action. For this reason, this methodology is sometimes called also closed-loop evaluation. It is slow and expensive to do an on-policy evaluation in the real world and one solution often used is using an on-policy evaluation in a simulator as an intermediary step. The most popular simulator is CARLA [DRC+17], but it is not used in this work as sim-to-real techniques need to be used to bridge the gap between the simulator and the real world. Data-driven simulators [AWG+21] are gaining traction to lower the cap between simulation and the real world and would be a good fit for this work. However, no free data-driven simulator was available during this work.

3.6.1 Off-policy Evaluation

Models are first evaluated using off-policy metrics as these are cheap to calculate. The main metric is the mean absolute error (MAE) between human command and model predictions. It has been shown to have a favorable correlation with driving ability [CLKD18] compared to the often used mean squared error (MSE). MAE is also broken down into directional commands (straight, left, right) to understand model accuracy at crossroads better. In addition, following [TAvGM22] [EMH17] [Fer18], whiteness is calculated to measures the smoothness of the sequence of predicted actions. It is calculated as:

$$W = \sqrt{\frac{1}{D}\sum_{i=1}^{D}(\frac{\delta P_i}{\delta t})^2}, \tag{1}$$

where D is the size of the dataset, δP_i is the change in predicted steering angle between each timestep, and δt is the duration of each timestep. As cameras work in 30hz configuration, δt is 0.033. Whiteness can be calculated for both on- and off-policy recordings. In the case of off-policy, whiteness is calculated from a sequence of predictions using human driving recordings from the evaluation track.

3.6.2 On-policy Evaluation

Models are evaluated on the 10 km Elva track (SS10/14 during WRC Rally Estonia 2022), driving it both ways (Figure 13). Kulbilohu rally-cross circuit at the beginning of the track is excluded as it needs special permission. Models are tested at 50% of the speed used by a human at the exact location on the track while collecting the dataset. This dynamic speed control allows to slow down in sharp turns and drive faster in wide straight sections. Driving at 100% human speed was successfully used with steering angle models, but the controller used to convert waypoints to steering angle causes oscillation. Also, as rally tracks are mostly relatively narrow and the margin of error is small, a lower speed is preferred to avoid premature overtaking by the safety driver.

Figure 13. All models are evaluated on the 10 km WRC Rally Estonia Elva stage.

The primary on-policy metric is distance per intervention. During driving on the test route, safety drivers will take over when there is a risk of driving off the road. Distance per intervention is calculated by counting these situations and dividing by the total distance traveled. Safety driver takes over as late as possible to let models recover, sometimes producing weird trajectories but reducing human subjectivity on what is good driving. Models are not trained to handle other vehicles and pedestrians, and the safety driver will take over when there is an accident risk. As different drives can have a different number of these situations, these are excluded from the results. As this is a country road, there can be other obstacles on the road like birds, fallen tree branches or big rocks, causing intervention, which is excluded from calculations as well.

The driving trajectory of the model was also compared to the human trajectory on the same route. Trajectories were recorded using NovAtel Pwr-Pak7 GNNS receiver and combined with inertial navigation (INS) system data with real-time kinematic positioning (RTK) with centimeter-level precision. For each position in the model-driven trajectory, the two closest positions in the human-driven trajectory are found and a line is constructed between these. The shortest distance between the line and a position from the model-driven trajectory is calculated as a lateral error. Additionally, the failure rate is calculated as the proportion of the time the lateral error was more than 1 meter.

Similar to off-policy evaluation, whiteness is used to evaluate driving performance. Here whiteness is calculated from the sequence of actions the policy takes while driving. During testing the models, high whiteness often leads to intervention and can be considered a measure of uncertainty. Model not responding to two similar frames (e.g., consecutive frames) with consistent predictions reveals its inability to deal with the situation, as can be clearly observed at the crossroads.

A big part of this work was also to use directional commands to handle crossroads. The performance metric used to measure this is a percentage of crossroads completed. Here only crossroads handled using the navigation command were included as going straight on the crossroad is just general road following. There are six crossroads in forwards drive and five in backward. The backward drive has one less as one junction is Y-shaped and the navigation command is not used in this case.

4 Results

This section compares models' predictive and driving performance using off- and on-policy metrics, followed by the analysis of the most common failure cases.

4.1 Candidate Models

The baseline steering angle model trained with no additional training technique already performed quite well, as also shown in our previous work [TAvGM22]. A model trained with data balancing performed surprisingly worse than the baseline model, cutting corners on the crossroads and turning off randomly on the straighter road sections. Similar effects were magnified with the model trained using augmented data from side cameras. The assumption can be made that this was caused by having too many frames with big steering angles and the model started to be biased to steer too much. With side camera augmentation, problems could also be attributed to possible incorrectness introduced by calculating steering angle labels from the trajectory. The steering angle model with pre-trained conditional control branches had driving performance similar to the baseline model but drove smoother on the crossroads with fewer interventions as left and right branches had been trained with more data.

The waypoints baseline model drove badly with many interventions, with the main problem drifting off the good trajectory and the inability to recover. Improving training with images from side cameras and balancing data distribution by steering angle seemed like good directions that could be used to solve this problem. The baseline model also struggled with taking crossroads with cutting corners and behaving unstable when the navigational command was given. The model trained with images from side cameras had the same problems as the steering angle model trained with this technique and performed poorly with a high number of interventions and did not improve the baseline. Balancing the dataset improved the driving performance significantly, with the model able to recover when drifting off a good trajectory, although sometimes doing it too aggressively. This model also had a good performance at the crossroads, with the leading cause of interventions being caused by starting the turn too early and cutting the curb. Initializing conditional branches did not improve driving performance at the junctions compared to the model trained with balanced data. As training techniques were not combined to limit the number of models, data balancing and pretraining conditional branches were not used in conjunction, although it could have improved the model further.

The best model for steering angle was trained with initializing conditional branches. The only model working well for waypoints prediction was using the data balancing technique, which again did not work well for steering angle. A different technique was used to train the models for each output model to not handicap modalities with a training methodology that does not benefit them.

4.2 Off-policy Metrics

Off-policy metrics are bad predictors for driving performance but are needed to train and choose models for evaluation in the real life. Off-policy metrics are given in Table 1. For the waypoints models, the trajectory is transformed into a steering angle with the controller described in Section 3.1.2. Metrics are calculated only with Elva test tracks, excluding the recording used for over-fitting to remove the effect of memorization and only see the generalized performance. During training, there were more tracks in the validation set, but these are excluded here to compare the off-policy metrics to the on-policy metrics calculated in the next section.

Model	**MAE total (°)**	**95th perc. (°)**	**MAE left (°)**	**MAE straight (°)**	**MAE right (°)**	**Whiteness (°/s)**
Steering angle v1	7.70	20.47	57.26	5.77	21.35	76.62
Steering angle v2	7.26	19.79	51.56	5.47	22.16	78.53
Steering angle overfit	6.56	18.27	39.39	5.18	20.43	63.80
Steering angle mean	7.17	19.51	49.40	5.47	21.31	72.98
Waypoints v1	10.82	29.88	88.81	7.40	48.76	150.25
Waypoints v2	12.01	31.88	86.68	8.59	54.87	176.25
Waypoints overfit	10.30	27.96	68.69	7.44	52.27	117.07
Waypoints mean	11.04	29.91	81.39	7.81	51.97	147.86

Table 1. Results of off-policy evaluation. MAE total is the mean difference between predicted and expert steering angles. 95th percentile shows the value from which MAE total is 95 percent of the time below. MAE left and right are mean errors using only frames originating from turns on crossroads. MAE straight excludes turns on crossroads. Whiteness measures the mean smoothness of the sequence of commands generated.

Steering angle models demonstrate a better predictive ability across all metrics. MAE total shows the mean difference between predicted and expert steering angles. Waypoints models have an MAE total of around 54% higher, showing considerably lower predictive ability. Although this difference should be taken with skepticism as it could be caused by converting the predicted trajectories to steering angles via the controller module, there is no other reasonable way to compare the metrics of two different output modalities. The maximum steering angle error is very high for all models, but the majority of top errors are during turning on a crossroad with limited visibility (left picture on Figure 14).

Left and right MAE measure mean losses when predicting left and right turns at crossroads. MAE straight shows loss during frames where no directional command is given, although it can still include some sharp turns. Turning on crossroads has significantly higher losses than other sections. The first reason is that there are fewer crossroad turns than road following sections in the dataset, making it harder for the model to learn. Secondly, using only the primary camera does not give a sufficiently wide viewing angle to correctly predict steering angle or trajectory at most crossroads, as seen in Figure 14. Humans have to turn their heads in sharp turns, indicating the need to use side cameras as was done successfully in an urban environment by Wayve

[HSG+20]. There is also a considerable difference in errors between left and right turns, but this comes from the peculiarities of the test track as there are more difficult left turns. In contrast, right turns usually have better visibility. The models were tested on the test track by driving both ways, but junction difficulty can differ depending on which direction it is approached.

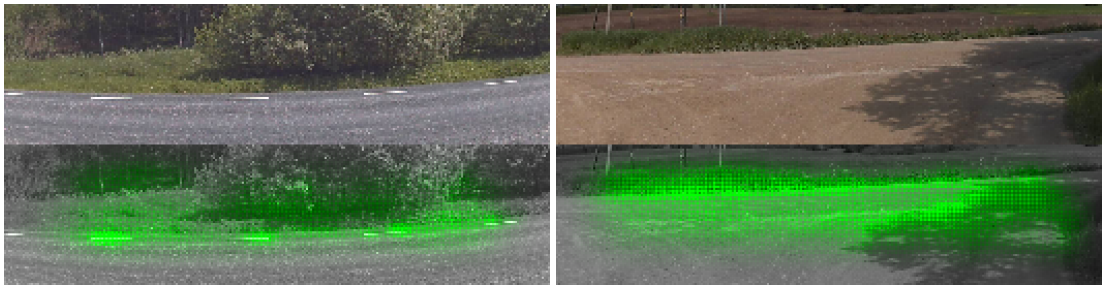


Figure 14. Crossroads are challenging for the models because of limited visibility. **Left**: A model cannot see where it is turning and fails to predict the correct driving command. **Right**: Visibility is better and models are able to predict a more accurate steering angle. The green color shows what pixels were considered most important by the model when making the prediction. The most critical regions of the input image are calculated using the VisualBackProp method and highlighted with green.

The considerable difference in whiteness shows that the models predicting waypoints do not respond consistently to consecutive similar frames. For both output modalities, it can be seen that over-fitting models to the test track have improved performance, but the difference is not very big. Off-policy metrics indicate steering angle models are better than waypoints models; however, driving performance estimated by on-policy metrics is more meaningful.

4.3 On-policy Metrics

Six models, three for each output modality, were tested on the test track driving autonomously both ways resulting in around 20km for each model. All the drives with resulting interventions can be seen in Figure 15, as well as driving videos on YouTube [4].

As seen in Table 2, all models could drive with an average distance per intervention over 1km. Models predicting waypoints could recover better when the vehicle got off the ideal trajectory, which resulted in fewer interventions. Both models were quite good at taking crossroads that needed navigational commands using turn signals, but waypoints models also had a clear advantage here.

[4] https://youtube.com/playlist?list=PLgvCZzErl12Y6qw7S2aKRjV42XUHlSIiV

Model	Distance (m)	Interventions	DiP (m)	Lat. error (m)	Failure rate (%)	Crossroads (%)	Whiteness (°/s)
Steering angle v1	19685	18	1094	0.50	11.97	54.55	33.05
Steering angle v2	19848	12	1653	0.54	13.31	81.82	31.94
Steering angle overfit	19990	10	1999	0.57	13.01	81.82	32.04
Steering angle total	59523	40	1488	0.54	12.77	72.73	32.50
Waypoints v1	20178	6	3363	0.45	8.45	90.91	39.13
Waypoints v2	19962	10	1996	0.44	7.52	72.73	44.27
Waypoints overfit	20004	8	2500	0.44	8.56	81.82	36.20
Waypoints total	60144	24	2506	0.44	8.18	81.82	40.04

Table 2. Results of on-policy evaluation. Distance is autonomous distance driven by the model. Interventions are the number of times the safety driver takes control of the vehicle. Interventions caused by other vehicles or obstacles on the road are excluded. The DiP is the autonomous distance driven per intervention. Lateral error defines the mean deviation from the expert trajectory. The failure rate is the time lateral error percentage was greater than 1m. Crossroads percentage defines success rate on crossroads needing navigation command (turn left, turn right). Whiteness measures the smoothness of the sequence of predicted action.

As seen from the whiteness metric, one aspect where steering angle models were better was the drive's smoothness. This was empirically felt during drives with an over-fitted steering angle model having the best human experience. It was especially noticeable during the forward drive with the overfitted model, where a long section of the track was completed without interventions.

The worst performing model was steering angle v1, but this was mainly caused by interventions on crossroads, and performance on the crossroad is currently quite variable due to limited viewing angle.

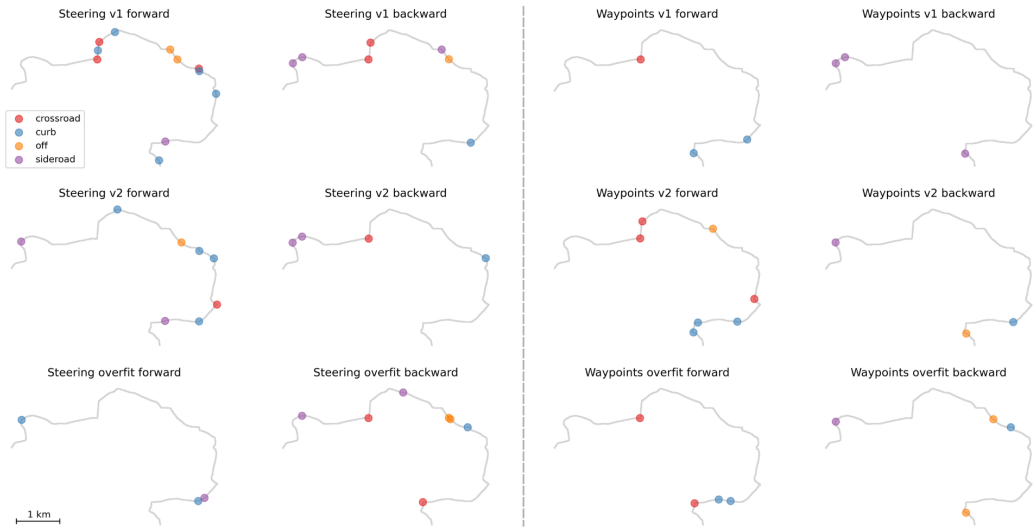


Figure 15. Six models were tested on the track near Elva by driving 120km autonomously. Failure cases needing the safety driver to take over command of the vehicle are marked on the figure with the color indicating the type of intervention.

4.4 Failure cases

Causes of interventions were similar for both output modalities and can be roughly divided into four categories as seen in Figure 16.

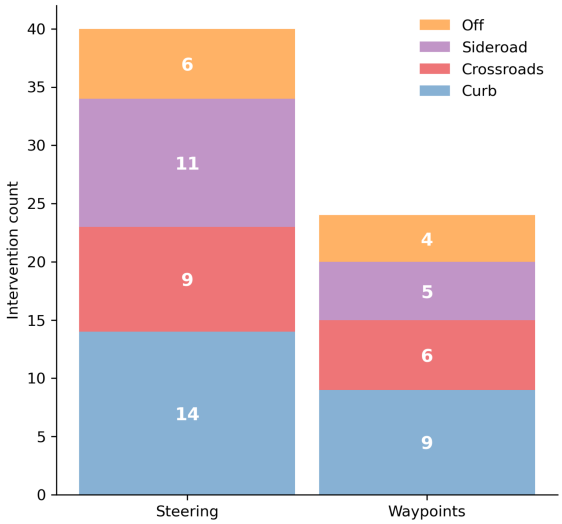


Figure 16. Intervention counts for both output modalities during drives with the models.

4.4.1 Curb Cutting

The most frequent intervention for both steering angle and trajectory models is cutting the curb of the road. This problem can be explained by limited visibility, as only the central camera was used, combined with relatively narrow roads. Interventions happened mainly on the turns but occasionally on straight roads when the vehicle drifted slowly towards the curb.

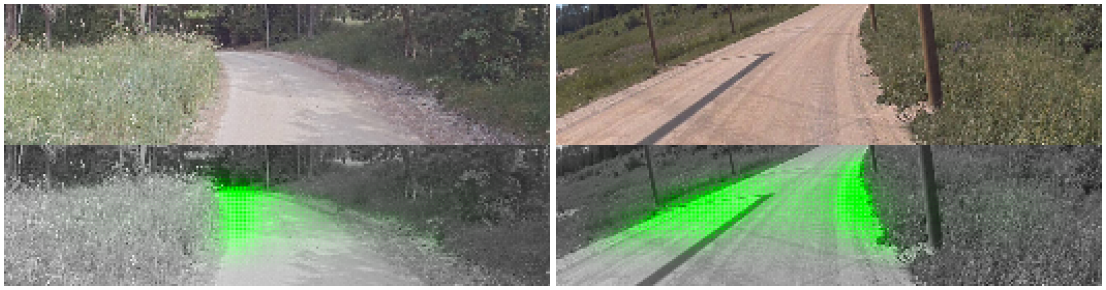


Figure 17. The most common reason causing interventions was cutting the curb of the road. **Left**: Steering angle overfit model drove on the left curb of the road. **Right**: Waypoints model v2 drifted towards the right side of the road. In this section road is tilted towards the right and models had problems countering this inclination.

4.4.2 Crossroads

Models often failed to execute a turn on the crossroad when the direction was given using the turn indicator. The common problem was starting the turn too early and cutting the curb. It is understandable as only the centrally positioned camera is used, and the field of view is not great. A model trained with a wide crop was also tested, but the results were similar. One option would be to use side cameras positioned towards the side of the road, which has been shown to work in urban settings [HSG+20]. It would have needed a sensor fusing of three camera images, which was out of the scope of this work.

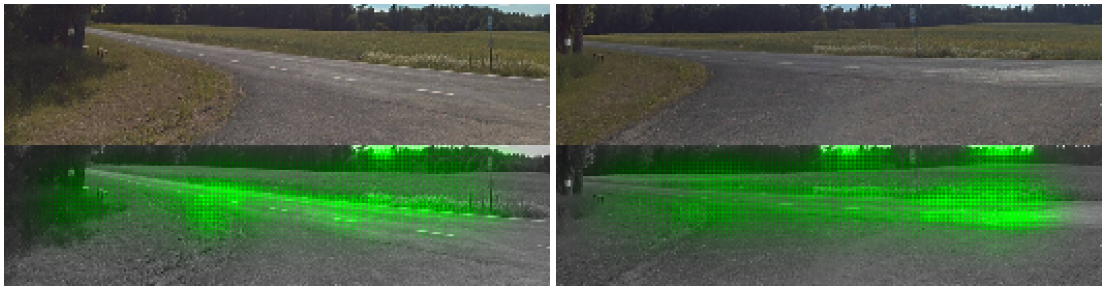


Figure 18. A difficult crossroad, where several models failed to make the turn. **Left**: Steering angle v1 model struggled with crossroads. Here it turned too aggressively and cut the left curb of the road. **Right**: Waypoints model v1 managed to complete the crossroad, although there does not seem to be much difference in the saliency map. Waypoints model seems to have better road positioning and avoids cutting the curb.

4.4.3 Side Roads

One of the biggest problems for all models, especially steering angle models, was taking a side road instead of continuing straight on the main road. It can be seen on the left picture in Figure 19, where the model took the left turn instead of driving straight, although no turn navigational command was given. For a human, it is easy to understand that we should continue straight, but the model seems to prefer going in the direction of the most road surface area. In this case, the shadow on the road also makes the situation harder for the model.

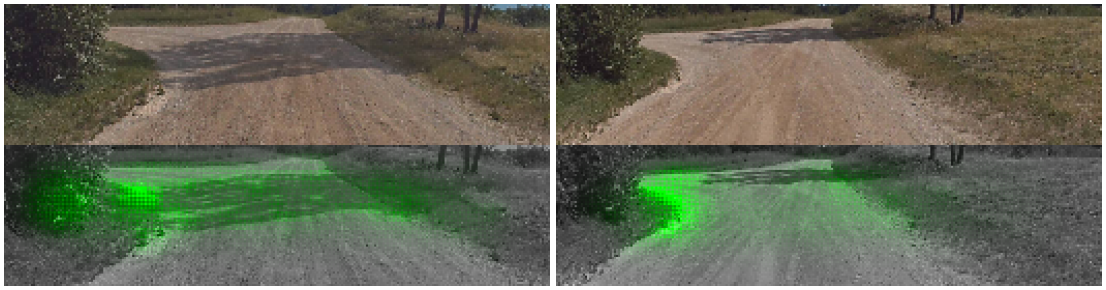


Figure 19. The model considers the side road as the main road and makes the mistake of turning left. The green color shows what pixels were considered most important by the model choosing the route. **Left**: The steering angle v2 model took the left turn, although no left navigational command was given. **Right**: Waypoints v2 model could avoid the left turn. It could be attributed to better road positioning as the vehicle is more on the right side of the road. The tree's shadow has different positions, which could affect the model's decision.

The problem with side roads was especially evident in one section, where 5 out of 6 models took the side road towards the excavation site. Although in this case, the models behaved very human-like as the same mistake was made by two human drivers when collecting data as this side road is more significant than the main road as seen in Figure 20). Note that human mistakes made during data collection are not in the training data, so it is not learned by the model and is a general emerging behavior. Here a better representation of navigational command would help as the turn signal does not have rich information to handle these complex situations.

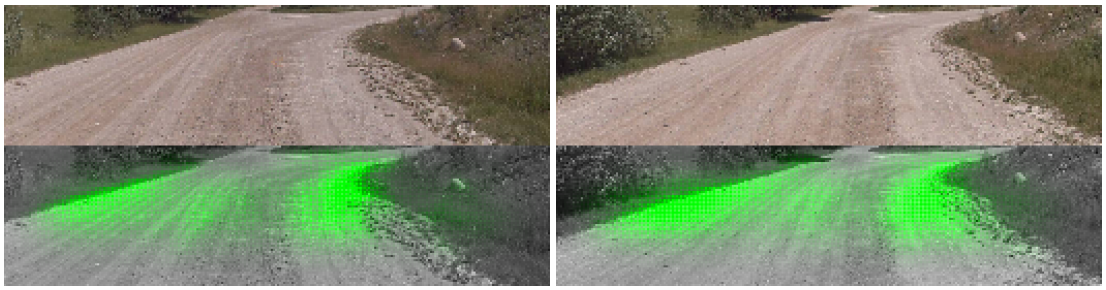


Figure 20. The model considers the side road as the main road and turns towards the entrance of an excavation site. **Left**: Steering angle v2 model turned towards the excavation site. **Right**: Waypoints v2 model has a very similar saliency map and makes the same mistake turning towards the excavation site.

It is difficult to diagnose, but it seems like long-term planning and better road

positioning could be the reasons that help waypoints models to avoid some of the side roads.

4.4.4 Turning Off The Road

In rare cases, models turned off the road for some unknown reason, for example, toward a rock wall or tree, as seen in Figure 21. Another cause was losing sight of the road on a deep ascent. These incidents would be challenging to resolve as there is a fat-tail of random incidents, especially when driving in more dynamic environments like city streets.

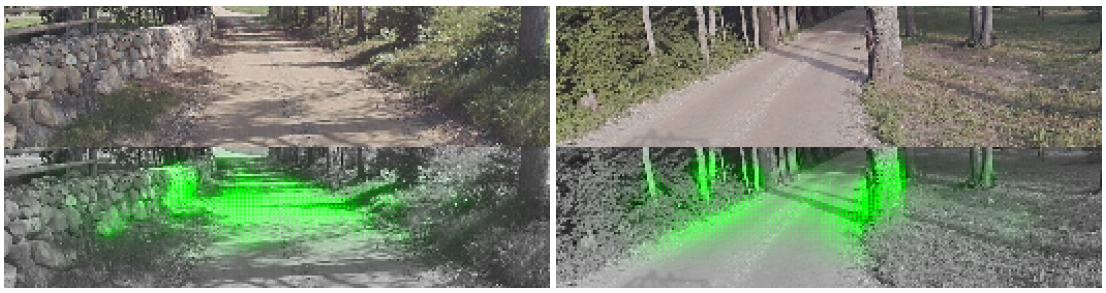


Figure 21. The model can get confused about the environment and turn off the road. **Left**: Steering angle v1 model turned towards the stone wall. The shadow on the road and the texture of the stone wall look like a left-turning road. The model gets confused and turns left. **Right**: Waypoints overfit model considers the tree as an extension of the road and steers towards it.

5 Conclusion

This work encompassed the machine learning pipeline from data collection to deploying models on the autonomous physical vehicle. Expert driving data recorded on WRC Rally Estonia 2020 and 2021 tracks was cleaned and prepared for training. Camera sensory input was used to train models to predict steering angle and trajectory. Models were evaluated using both off-policy and on-policy using 50% of human speed.

Models predicting steering angle could drive 1488 meters per intervention on average. As the model predicts vehicle control, which can be fed directly into the vehicle's control system without an additional controller, it is a straightforward solution to implement. The downside is that as the model predicts just one scalar value, it is harder to interpret as one single value does not give much information on the model's longer plans or strategy.

Models predicting waypoints were able to drive with fewer interventions than steering angle models and averaged 2506 meters per every intervention. The downside was

that these models drove less smoothly, which can be caused by the simple controller developed just for this use case, but more research must be done to understand the cause. The trajectory of waypoints is more straightforward to comprehend for humans than a single steering angle as it conveys the long-term plan. In addition to being humanly understandable, a trajectory is easier to integrate with other systems like object detection or safety fallback [VCY+21]. Predicting trajectory also generalizes across vehicles as only the controller converting trajectory into vehicle-specific control command has to be each vehicle specific.

All models made similar mistakes and it is hard to tell how much long-term planning helps waypoint models. It seems it is improving the road positioning and recovering after drifting off the ideal trajectory, but this could also be improved for steering angle models with a better training process or using a sequence of steering angles.

6 Future Work

Doing real-world autonomous vehicle research is a complex and time-consuming process. The other goal of this thesis is to document the machine learning process needed for developing an end-to-end autonomous vehicle platform. Hopefully, learnings from this work and open-sourced code [5] will help future students have a good baseline for future research. The main limitation of current work is that the controller used for converting trajectory into steering angles was basic, which set the ceiling for the speed used. With higher speeds, the oscillation was a problem and caused unwanted interventions, which would make the comparison unfair as this was not a problem for the model predicting steering angle directly. A case can be made that problem comes from model predictions, but this needs to be verified.

Although the car is fitted with four cameras, only one central camera was used. Using side cameras can potentially improve models' performance at crossroads as the side-wise visibility is relatively limited using only the central camera. It could also help improve road positioning and decrease interventions caused by cutting the curb. Preliminary tests were done to fuse images of one central camera and side cameras, but as the system could not process 30 frames in a second, performance tuning must be done to use this approach. Also, the side cameras are currently too straight looking, and it would be helpful to change the angle towards the side of the road for a more diverse view of the environment.

Side cameras can also augment the training process for better recovery from the non-ideal trajectory [BdTD+16]. Models trained using this technique were tried, but it caused too much swerving and interventions. It could be caused by the problem that the two side cameras did not have an equal angle in relation to the vehicle's viewing

[5] https://github.com/UT-ADL/e2e-rally-estonia

direction, causing the dataset to be unbalanced.

Rally tracks are relatively uneven, with vehicles' pitch and roll changing more than driving on a street or highway. Models showed some problems when the road had an ascent or side-wise slant. It would be possible to use vehicle state information like pitch and roll to give more signals to the model to achieve correct behavior. On the deep ascents, visibility gets restricted and the position of the horizon changes (Figure 22). Dynamically cropping a different input image area using pitch value could improve the results.

Figure 22. During steeper ascents, the horizon is lower in the cropped image causing models to concentrate on the sky and lowering predictive performance.

Collecting more data is an easy way to improve model performance, as preliminary tests showed that the model's performance had not saturated yet. Adding more data can improve both interventions and whiteness metrics. Data and its quality are usually more important than model architectures. While much research is focused on neural network architectures, with newer architectures showing better performance in benchmarks, the road following is quite a simple and non-dynamic task. It is doubtful that a change in network architecture would improve the performance significantly.

References

[ARKR19] Alexander Amini, Guy Rosman, Sertaç Karaman, and Daniela Rus. Variational end-to-end navigation and localization. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8958–8964, 2019.

[AWG+21] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertaç Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. *ArXiv*, abs/2111.12083, 2021.

[BCC+16] Mariusz Bojarski, Anna Choromańska, Krzysztof Choromanski, Bernhard Firner, Lawrence D. Jackel, Urs Muller, and Karol Zieba. Visualbackprop: visualizing cnns for autonomous driving. *ArXiv*, abs/1611.05418, 2016.

[BCD+20] Mariusz Bojarski, Chenyi Chen, Joyjit Daw, Alperen Deugirmenci, Joya A. Deri, Bernhard Firner, Beat Flepp, Sachin Gogri, Jesse Hong, Lawrence D. Jackel, Zhe Jia, BJ Lee, Bo Liu, Fei Liu, Urs Muller, Samuel Payne, N. N. S. S. R. K. Prasad, Artem Provodin, John Roach, Timur Rvachov, Neha Tadimeti, Jesper E. van Engelen, Haiguang Wen, Eric Yang, and Zongyi Yang. The nvidia pilotnet experiments. *ArXiv*, abs/2010.08776, 2020.

[BCP+20] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. Label efficient visual abstractions for autonomous driving. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2338–2345, 2020.

[BdTD+16] Mariusz Bojarski, David W. del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *ArXiv*, abs/1604.07316, 2016.

[BYC+17] Mariusz Bojarski, Philip Yeres, Anna Choromańska, Krzysztof Choromanski, Bernhard Firner, Lawrence D. Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *ArXiv*, abs/1704.07911, 2017.

[CLKD18] Felipe Codevilla, Antonio M. López, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. In *ECCV*, 2018.

[CMD+18] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio M. López, and Vladlen Koltun. End-to-end driving via conditional imitation learning.

2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–9, 2018.

[Cou92] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. 1992.

[CSKX15a] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[CSKX15b] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2722–2730, 2015.

[CSLG19] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9328–9337, 2019.

[CSU21] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14398–14407, 2021.

[CZKK19] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. *ArXiv*, abs/1912.12294, 2019.

[dHJL19] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *NeurIPS*, 2019.

[DRC+17] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. Carla: An open urban driving simulator. *ArXiv*, abs/1711.03938, 2017.

[EMH17] Hesham M. Eraqi, Mohamed N. Moustafa, and Jens Honer. End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. *ArXiv*, abs/1710.03804, 2017.

[Fer18] Nelson Fernández. Two-stream convolutional networks for end-to-end learning of self-driving cars. *ArXiv*, abs/1811.05785, 2018.

[HBWZ17] Christian Hubschneider, Andre Bauer, Michael Weber, and Johann Marius Zöllner. Adding navigation to the equation: Turning decisions for end-to-end vehicle control. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2017.

[HHBK21] Jeffrey Hawke, E Haibo, Vijay Badrinarayanan, and Alex Kendall. Reimagining an autonomous vehicle. *ArXiv*, abs/2108.05805, 2021.

[HSG+20] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemyslaw Mazur, Sean Micklethwaite, N Griffiths, Amar Shah, and Alex Kendall. Urban driving with conditional imitation learning. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 251–257, 2020.

[HTMT07] G.M. Hoffmann, Claire J. Tomlin, Michael Montemerlo, and Sebastian Thrun. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. *2007 American Control Conference*, pages 2296–2301, 2007.

[JPGO21] Ashesh Jain, Luca Del Pero, Hugo Grimmett, and Peter Ondruska. Autonomy 2.0: Why is self-driving always 5 years away? *ArXiv*, abs/2107.08142, 2021.

[JPKA95] Todd Jochem, Dean A. Pomerleau, B. V. K. Vijaya Kumar, and Justin Armstrong. Pans: a portable navigation platform. *Proceedings of the Intelligent Vehicles '95. Symposium*, pages 107–112, 1995.

[KHJ+19] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254, 2019.

[LMC+19] G. Li, Matthias Müller, Vincent Casser, Neil G. Smith, Dominik Ludewig Michels, and Bernard Ghanem. Oil: Observational imitation learning. *Robotics: Science and Systems XV*, 2019.

[MBB+08] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008.

[MBC+05] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann Cun. Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[MDGK18] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. In *CoRL*, 2018.

[PBOB+20] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[Pom88] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[RGB11] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[SCR+20] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *ECCV*, 2020.

[SDV+16] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *ArXiv*, abs/1611.07450, 2016.

[SFMP21] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, PP:1–1, 11 2021.

[Sie03] M. Siegel. The sense-think-act paradigm revisited. In *1st International Workshop on Robotic Sensing, 2003. ROSE' 03.*, pages 5 pp.–, 2003.

[Sni09] Jarrod M. Snider. Automatic steering methods for autonomous automobile path tracking. 2009.

[SSG18] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *Conference on Robot Learning (CoRL)*, 2018.

[STIM18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *NeurIPS*, 2018.

[TAvGM22] Ardi Tampuu, Romet Aidla, J. van Gent, and Tambet Matiisen. Lidar-as-camera for end-to-end driving. *ArXiv*, abs/2206.15170, 2022.

[TSM+20] Ardi Tampuu, Maksym Semikin, Naveed Muhammad, Dmytro Fishman, and Tambet Matiisen. A survey of end-to-end driving: Architectures and training methods. *IEEE transactions on neural networks and learning systems*, PP, 2020.

[TWB20] Varundev Suresh Babu Trent Weiss and Madhur Behl. Bezier curve based end-to-end trajectory synthesis for agile autonomous driving. In *NeurIPS 2020 Machine Learning for Autonomous Driving Workshop*, 2020.

[UAB+08] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matthew Mcnaughton, Nick Miller, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25:425–466, 01 2008.

[VCY+21] Matt Vitelli, Yan-Xia Chang, Yawei Ye, Maciej Wołczyk, Bla.zej Osi'nski, Moritz Niendorf, Hugo Grimmett, Qiangui Huang, Ashesh Jain, and Peter Ondruska. Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. *ArXiv*, abs/2109.13602, 2021.

[WAS+22] Tsun-Hsuan Wang, Alexander Amini, Wilko Schwarting, Igor Gilitschenski, Sertaç Karaman, and Daniela Rus. Learning interactive driving policies via data-driven simulation. *ArXiv*, abs/2111.12137, 2022.

[WB20] Trent Weiss and Madhur Behl. Deepracing: Parameterized trajectories for autonomous racing. *CoRR*, abs/2005.05178, 2020.

[XWCL15] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *ArXiv*, abs/1505.00853, 2015.

[ZKK19] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. Does computer vision matter for action? *Science Robotics*, 4, 2019.

[ZKL+16] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *2016*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2921–2929, 2016.

[ZWL+21] Jinyun Zhou, Rui Wang, Xu Liu, Yifei Jiang, Shu Jiang, Jiaming Tao, Jinghao Miao, and Shiyu Song. Exploring imitation learning for autonomous driving with feedback synthesizer and differentiable rasterization. *ArXiv*, abs/2103.01882, 2021.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Romet Aidla**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Comparing Output Modalities in End-to-End Driving,
   *(*title of thesis)

   supervised by Tambet Matiisen and Ardi Tampuu.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Romet Aidla
08.08.2022