University of Tartu
Institute of Computer Science
Computer Science Curriculum

Illimar Laanisto

Testing Nvidia Drive for small cars in toy town

Bachelor's Thesis (9 ECTS)

Supervisor
Ardi Tampuu

Tartu 2022

Testing Nvidia Drive for small cars in toy town

Abstract: The aim of this bachelor's thesis is to analyze and evaluate the capabilities of neural network based image processing in Nvidia Drive in conditions far from its original operating domain. The data for the models is gathered from a toy town made of wood and populated with toy cars and toy pedestrians. This town is used for self-driving student competitions, teaching and public demonstrations. If applicable to this data, tools from NVIDIA Drive could be used for achieving self-driving with toy cars in this toy town, similar to how NVIDIA drive can be used on real cars in the real world. Three different Nvidia Drive models are tested in this thesis, which are DriveNet, PathNet, and OpenRoadNet. The performance of these models is evaluated and a conclusion is made.

Keywords: Nvidia Drive, Neural Networks, Donkey Car, Object recognition
CERCS: P170 - Computer Science, numerical analysis, systems, control

Nvidia Drive-i testimine mängulinnas väikestel autodel

Lühikokkuvõte: Antud bakalaureusetöö eesmärgiks on analüüsida ja hinnata närvivõrgupõhise pilditöötluse võimalusi Nvidia Drive'is mitte-ideaalsetes tingimustes. Andmed mudelite jaoks on kogutud puidust mänguasjalinnakust, mis on täidetud mänguasjadest valmistatud autode ja jalakäijatega. Seda linna kasutatakse isejuhtivate autode tundengivõistlusteks, õpetamiseks ja demonstreerimiseks. Juhul kui Nvidia Drive on nendel andmetel kasutatav, saab seda kasutada mänguandmetel isesõitmissüsteemi loomiseks, sarnaselt sellele, kuidas kasutatakse Nvidia Drive'i päris-elu situatsioonides. Selles lõputöös testitakse kolme erinevat Nvidia Drive'i mudelit, milleks on DriveNet, PathNet ja OpenRoadNet. Hinnatakse nende mudelite toimivust ja tehakse järeldus.

Võtmesõnad: Nvidia Drive, Tehisnärvivõrgud, Donkey Car, Objektituvastus

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Introduction

Self-driving cars have been a field in recent years that many are following. This technology has the great potential to revolutionize transport in the near future, both for the benefit of the average user and in the transport industry, such as freight. [1]

All major car manufacturers already use different systems in their cars which are often referred to as self-driving systems. The best known of these systems are automatic cruise control and lane-keeping, but there are also more sophisticated systems, such as the Tesla Autopilot, which in ideal weather conditions can already autonomously navigate on some types of roads. Tesla also claims that their cars have all the hardware to drive autonomously between any two points, but the software is not there yet. [2]

There are problems and challenges with every new technology, and self-driving cars are no exception. The biggest challenge is software, which is the main reason why we don't already see self-driving cars on the road. Self-driving cars use artificial intelligence, machine learning, and artificial neural networks to navigate the roads. The aim of this research is to deploy the artificial neural networks created by Nvidia as part of their Drive project in a slightly different domain, a toy town. Firstly, we wish to validate the applicability of these networks in this domain, with the hope of building self-driving toy cars based on them in the future. Secondly, in the course of this testing, we also get a better overview of how robustly these artificial neural networks work and we can draw conclusions about the strengths and weaknesses of the system.

The research method is software testing conducted in a toy town, using the Donkey Car platform to collect data. The resulting data is processed into a form that can be used by the artificial neural networks from NVIDIA Drive. In particular, the ability of neural networks to detect various toy objects from images and videos that are important in normal traffic, such as other cars, road borders, road signs, people, and traffic lights, will be studied and evaluated.

The first chapter provides an overview of the background related to self-driving cars, artificial intelligence, and the Nvidia Drive platform. The second chapter contains the methodology, including descriptions of the testing area and an introduction to the DonkeyCar robot. In addition, data collection, processing, and various experiments on an artificial neural network are

described. The third chapter presents and analyzes the test results and visualizes the results of the study.

1. Theoretical background

1.1 Neural Networks

1.1.1 Definition

Robert Hecht-Nielsen, author of the first textbook on artificial neural networks, defines artificial neural networks as follows: "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.”.[3] However, the definition from Merriam-Webster is clearer: “A computer architecture in which a number of processors are interconnected in a manner suggestive of the connections between neurons in a human brain and which is able to learn by a process of trial and error“. [4]

1.1.2 Description

As the name suggests, neural networks are inspired by the human brain and the networks of brain neurons in it. Artificial neural networks are made up of different layers, which in turn made of cells or points called artificial neurons. There are three types of layers: input layer, output layer, and hidden layer. There is only 1 input layer and 1 output layer, but there can be more hidden layers. Artificial neurons sum inputs an then apply activation functions. The role of the activation function is to introduce non-linearity and keep the values restricted to a limit, as without them the values can climb very high and cause problems. The input gets directed to the artificial neuron which has a weight that gets multiplied with the input, then converted by the activation funtion into an output value that is transmitted to the next layer or yielded as an output.[3][5]

1.1.3 Training

The learning (training) of artificial neural networks works by trial and error. This generally requires a large amount of training data, where the expected output is also known. Initially, the weights of each neuron are random. An epoch is a cycle that goes through all the training data and batch size determines how many samples are processed per batch. If the batch size is larger,

then there are less batches in an epoch, as all the data needs to be processed. During each batch, neurons change their weight values in a direction that brings the network's final output closer to the expected output. If the number of epochs is too large, the neural network may become overfitted. When the neural network is overfitted, it is only able to find the correct answers in the cases in the training data and may not work on similar but new data for the network. [3][5]

1.2 Object recognition

Object recognition is a technology in the field of computer vision. Identifying an object is finding the location of an object in an image and identifying its category or nature. Machine learning has been used for object recognition since the 1990s, but the use of deep neural networks has recently gained popularity [6]. A major problem with a conventional pre-deep learning machine learning based object recognition is the dependence on the human element. The features of an object are found in the image through an algorithm designed by the developers and largely depends on the developer's understanding of the object. Only positive and negative examples, as in both images with and without the objects in view, are needed to train deep learning methods, on the basis of which the neural network learns the necessary features. [7] Object recognition is one of the key technologies for self-driving cars, enabling them to recognize road signs and pedestrians or identify other moving cars. In this thesis, the identification of objects is used to identify toy versions of different objects in the toy town. These objects are other cars, people, road signs, and the free space for the car to drive.

1.3 Nvidia Drive

Nvidia Drive is a project created by Nvidia that includes many different self-driving related systems. Nvidia Drive was launched in 2015 [8] with its first software and hardware version. Since then, Nvidia has released 9 different versions of its hardware. Nvidia promotes the Drive platform as an end-to-end solution for self-driving cars. They provide hardware and software for the car, alongside data center and workstation solutions. In-car, the platform includes Drive OS software, Drive AGX generation internal computers, and the Drive Hyperion suite, which includes a complete set of sensors, an AI data processing platform (the aforementioned

computers) with a complete software package for autonomous control, driver monitoring, and visualization.[10]

The Nidia Driveworks SDK is used in this thesis. The Nvidia Driveworks SDK is part of the Nvidia Drive Software package. DriveWorks is able to run on any x86 Linux-based system with a Pascal GPU or newer.[9] The Driveworks version 2.2 used in this thesis will be run on Ubuntu 18.

1.4 Related work

Self-driving cars as a research topic have gained a lot of attention in recent years. A lot of research papers have been released that test different self-driving systems and their capabilities.[11] There are a lot fewer research papers that are directly related to Nvidia Drive. Most of the related work is based around Nvidia Drive hardware solutions, like the Nvidia Jetson, AGX, or PX.

One example of a related paper is a research paper about traffic sign identification.[12] The paper, written by R.Ravindran, M.J.Santora, M.Faied, and M.Fanaei focuses on using deep neural networks to identify traffic signs in real-time. To run real-time trials the authors used a car equipped with the Nvidia Drive PX2 system. The authors go into detail on different types of deep neural networks used and how those were evaluated. Despite NVIDIA drive networks being freely available and the results published by NVIDIA being very promising, there are not many independent works evaluating the usability of these networks in different use cases. In this work, we deploy the networks to a new domain of toy cars to test their usefulness and explore their ability to generalize.

2. Methodology

This chapter gives an overview of the research methodology. It describes the hardware and software needed for this thesis as well as how it was installed and set up. Problems that occurred during the process are also described.

2.1 Data collection

To carry out the experimental part of this thesis, the first step was to gather data for the models to analyze. This was done by using a toy town model and a Donkey Car. The data collection was carried out in steps, by gathering the data, inspecting it for defects, and discarding broken or corrupted data. In this step, no models or software were used, and the data was inspected by eye.

2.1.1 Toy town model

To gather data for the models used in this research a toy town was used (Figure 1). The toy town was used as a rough approximation of real streets to test the models' performance in non-ideal conditions. The toy town was made out of wood and was approximately 3 by 3 meters in size. It is a simulated town road with buildings on either side. The track had a total of 6 turns in it with different angles and sizes. There was a 180-degree turn, four 90-degree turns, and an S-bend. Children’s toys, specifically Lego Duplo, were used to simulate objects in a typical town. There were toy cars placed on the track in various different positions and orientations. On the sides of the road were also toy humans and toy traffic signs.

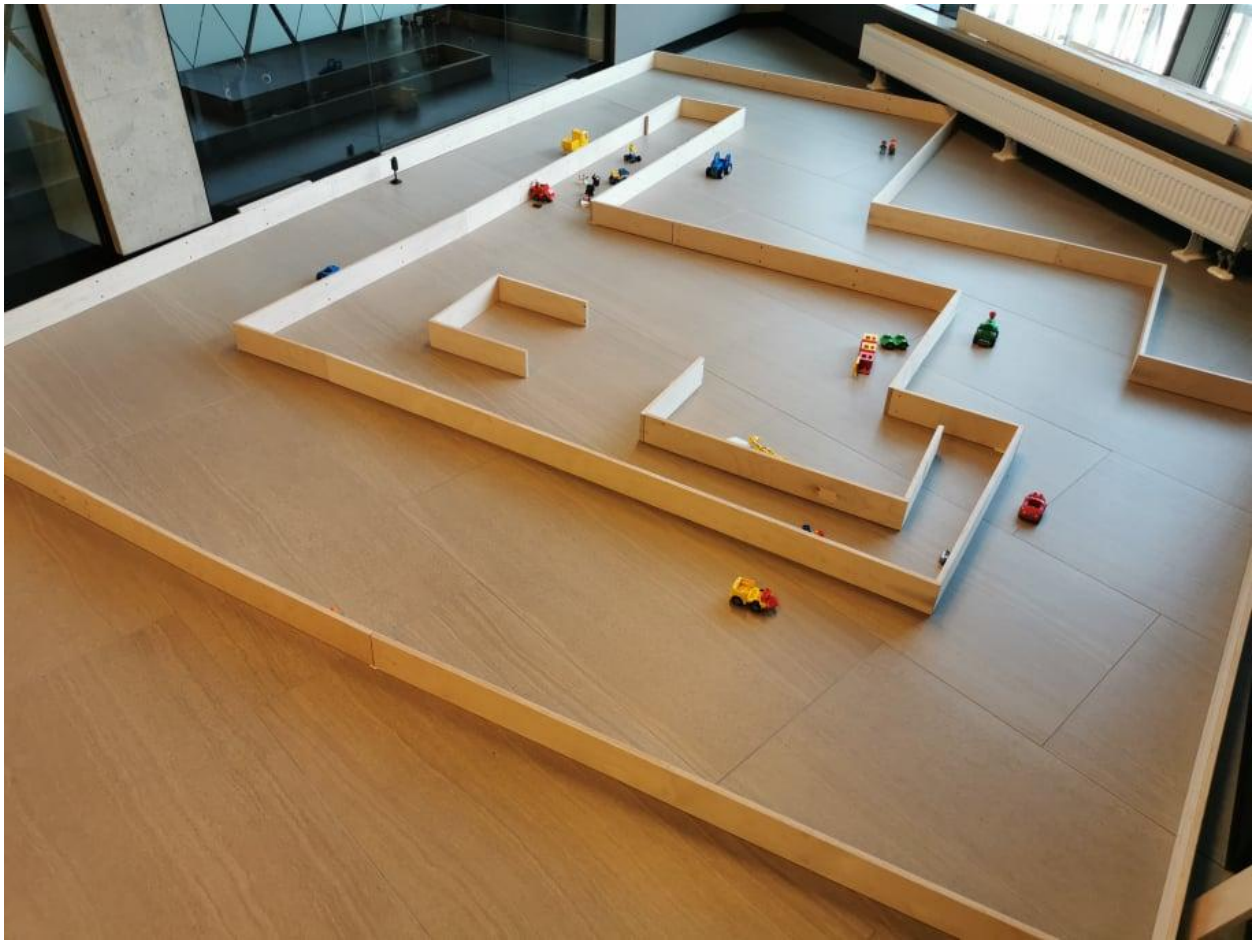

Figure 1. Toy town

2.1.2 Donkey Car

Donkey Car (Figure 2) was used to drive around in the toy town to gather data. Donkey Car is an open-source self-driving platform for small-scale cars. It is managed by an onboard Raspberry Pi 4. Donkey Car was chosen for data collection because it is easy to use and control. It was controlled directly by a human with a Logitech F710 gamepad and driven around at a constant speed as smoothly as possible. [13]

The car has an onboard wide-angle, 160-degree Raspberry Pi camera. The sensor used by the Raspberry Pi camera is the OV5647 developed by OmniVision. It has a native resolution of 5 megapixels, which equates to a 2592x1944 pixel resolution. For this thesis the following resolutions were used:

- 640x480
- 960x720

Figure 2. DonkeyCar

2.1.3 Processing

A total of 4 datasets were collected with a total of 1426 frames. The datasets were for different resolutions and directions of travel (clockwise and counter-clockwise around the town). The data recorded with the Donkey Car comes in folders named tubs. These tubs contain each frame of the video recording in a JPG file. This was already a suitable format for Nvidia Driveworks models.

2.2 Testing

Testing of the models was performed by running the collected data on the respective models and analyzing the output.

2.2.1 Host machine

In order to carry out the tests on the Driveworks model a Linux host machine needed to be used, as the implementations were made for use on Linux. Due to Windows 10 limitations, a virtual machine was not an option. Windows 10 does not allow pass-through of graphics cards to virtual environments. Nvidia Driveworks requires an Nvidia-made graphics card with Pascal architecture or newer. For this reason, dual-booting was used to run both Windows 10 and Ubuntu 18.04 on the same system, which allowed Ubuntu to use the installed GTX 1070TI graphics card. Dual-booting was chosen over a typical Ubuntu installation for usability, as it allowed to quickly switch between operating systems. Technopedia describes dual booting as follows: “A dual boot is a technique through which multiple operating systems can be kept within the boot sequence on the same computer. It enables a user to select from more than one operating system at the initial boot sequence or system startup”.[15] Ubuntu 18.04 was chosen because the Nvidia Driveworks version accessible with the Nvidia Drive AGX SDK developer program was only compatible with Ubuntu 18.04. This version was also used in the Autonomous Driving Lab at the University of Tartu and their labeler application, which is used in this thesis.

2.2.2 Nvidia Driveworks

In order to install Nvidia Driveworks, the Nvidia SDK Manager was used. The Nvidia SDK Manager is an application made for developers to install required software development kits. Nvidia Driveworks also required the CUDA toolkit, which had to be installed separately. The neural network models included in Nvidia Driveworks are bare-bones and need an application to run them. To run the models, the nn_labeler application from the Autonomous Driving Lab GitLab was used. [16]

The following models were chosen for testing: driveNet, openroadNet, and PathNet. These models were chosen as these have the most general features, which are easier to detect from a toy. It is easier to detect if an object is a car or a road sign than to detect which road sign it is.

2.2.3 DriveNet

DriveNet can detect different objects in everyday traffic and label them. Currently, the following objects can be detected:

- Car
- Bicycle
- Pedestrian
- Traffic sign
- Traffic light

Trucks can also be detected but will be labeled as cars by driveNet. Motorcycles are also detected and labeled as bicycles. DriveNet also calculates and returns some extra data that would be useful for self-driving applications: [17]

- A confidence value, which shows how confident DriveNet is, that the object was labeled correctly. This value is between 0 and 1.
- A depth value, which shows the distance to the object in meters.
- An urgency value, which is calculated from framerate and depth. The value is the inverse of time to collision, meaning the larger the value, the less time there is until the collision.

Figure 3. DriveNet example from Nvidia

2.2.4 OpenRoadNet

OpenRoadNet detects the open and free parts of the area that the car can reach without a collision. It can also detect the type of object or boundary that makes it impossible to reach an area. These obstacles are as follows: [18]

- Vehicles
- Pedestrians
- Curbs
- Undefined boundaries
- Other objects

Figure 4. OpenRoadNet example from Nvidia

2.2.5 PathNet

PathNet detects between 0 and 3 possible collision-free paths. There are 3 classes of paths to detect: [19]

- Path of the main vehicle
- Path adjacent to the main path on the right
- Path adjacent to the main path on the left

PathNet also outputs a confidence that shows if a given path contains traffic traveling in the opposite direction of the main vehicle. Sadly there is no offical Nvidia example photo for the output of this network.

3. Results

This chapter gives an overview of the results and metrics used. It describes evaluating the performance of each model and makes conclusions based on that. Due to an issue with CUDA toolkit compatibility and the lack of documentation on the issue, the rendering did not work on any Driveworks model. This affected the final results and evaluation phase greatly, as it isn't feasible to evaluate all the data from JSON files describing each frame. This could have been solved by writing a script to parse the JSON files into a collection of points that could be added to the original image, but this solution was out of the scope for this thesis, as the author is not well versed in the subject.

3.1 Metrics

In this thesis there are tests for 4 different models. Wherever possible, quantitative metrics were used. Unfortunately most of the metrics are qualitative due to the lack of precisely labeled test data. All of these models return different outputs and thus must be evaluated with metrics corresponding to these specific models. One metric all models share is the FPS or Frames Per Second, which was included to show the difference in computational time between different networks. Average FPS did not differ between resolutions.

3.1.1 DriveNet

DriveNet is the only model that could be evaluated with quantitative metrics. A total of 3 metrics were used in evaluating the performance of this model:

- FPS, which shows how fast the model could process a frame
- Total detection rate, which shows how many of the objects on the total area of the track were detected at least once
- Total correct detection rate, which shows how many of the objects on the total area of the track were detected correctly at least once
- Average detection rate, which shows the percentage of frames an object is detected correctly when it can be seen

3.1.2 OpenRoadNet and PathNet

OpenRoadNet and PathNet were evaluated based on a sample of frames due to the aforementioned rendering issue, specifically 14 frames per resolution. There was no difference between resolutions, so in the following results, different resolutions will not be written about specifically. These frames were chosen in front of turns and then evaluated qualitatively. There were also 20 random points chosen out of the outputs (frames) of these models and evaluated for quantitative insight, but the sample is not large enough to make meaningful statements solely based on those insights, as it is a random sample of about 0.2% of the data.

3.2 Model Specific results

3.2.1 DriveNet

The DriveNet model performed the worst out of all of the models, which was expected. The toy objects did not resemble real-world objects well enough for the model to detect them with meaningful consistency. In total over 4 different datasets with 1426 total frames, objects were detected a total of 6 times and only 2 of these detections were unique, as in 2 different objects were detected. In all six cases, DriveNet labeled these objects as traffic signs. The first detected object was a small construction cone, which was detected 4 times, while it was visible on 104 frames. The second detected object was a traffic light, which was detected 2 times, while it was visible on 153 frames. Only 2/11 (18.18%) object types were ever detected and only 1/11 were detected with the correct class label. The average detection rate was 0.7%, as 842 frames had objects fully visible and only 6 of those objects were labeled. This calculation does not consider frames with multiple objects in view as multiple failures. In that case, the average detection rate would be lower. The average frames per second the DriveNet model ran at was 266 FPS, completing each dataset on average in 1.5 seconds. Different resolutions did not affect the framerate. In conclusion, the DriveNet model did not work at all on images from a small-scale toy town, as the objects differed too greatly from the training data.

3.2.2 OpenRoadNet

OpenRoadNet was evaluated based on 14 samples of frames. The labeler running the model outputs a JSON file for each frame which was analyzed to match the described boundary points to the image. Out of the 14 frames analyzed 9 were before corners, 2 on a straight line and 3 during corners. Each sample provided a usable output. Out of the total of 280 points chosen from the 14 files, 243 were labeled correctly as road borders to a reasonable degree of accuracy, meaning the point was less than 5 pixels away from the exact pixel the wall started. About 63 of those pixels were exactly on the start of the wall. All of the 37 points chosen that were not identified correctly were labeled as part of a boundary while the road was open. This was most likely due to the perspective of the car, as the camera could not see around the upcoming corner while on a straight. The average FPS of this model was still calculated over all datasets and was 225 FPS. Generally, the OpenRoadNet model was accurate and produced good enough results to consider it successful.(Figure 5.)

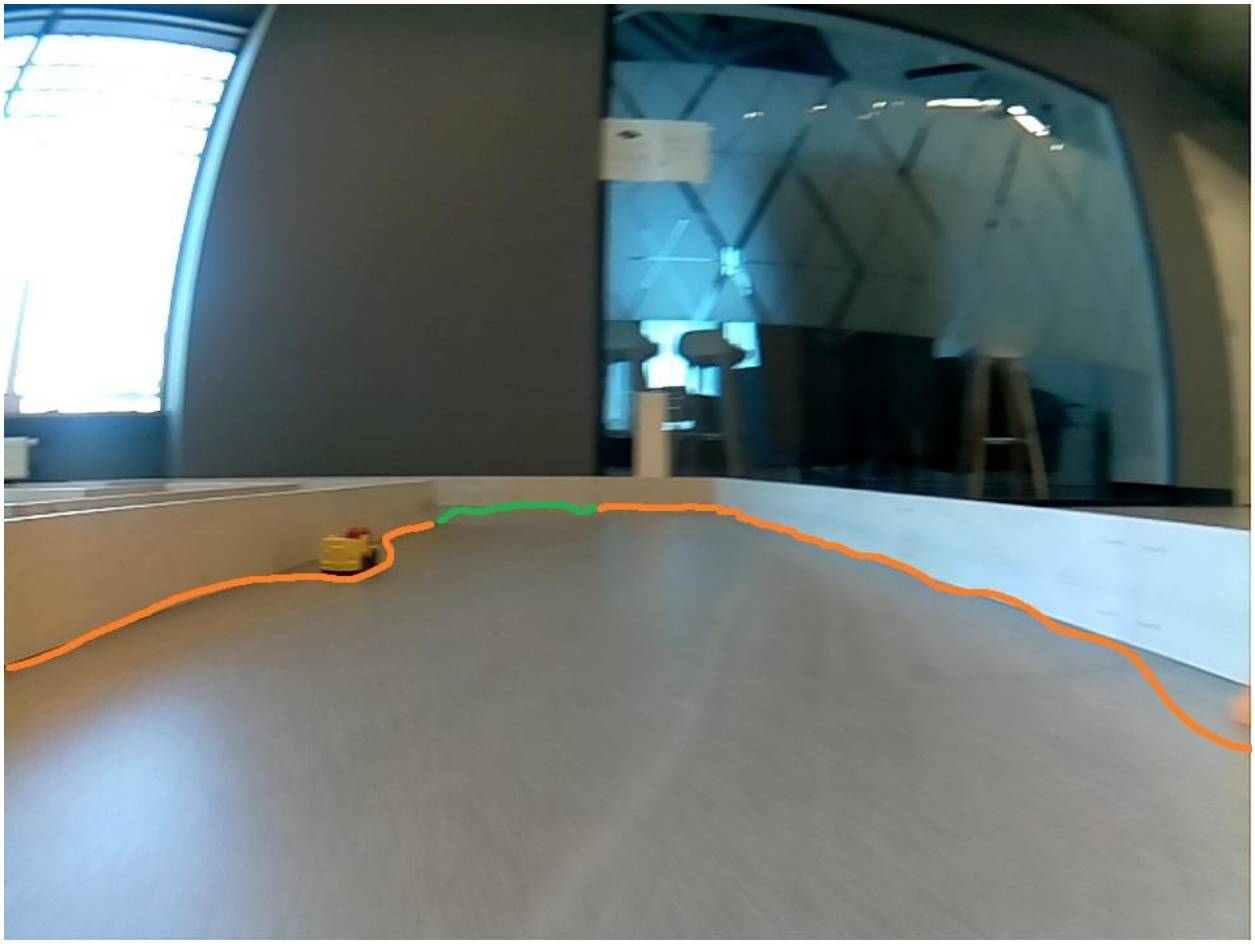

Figure 5. Example OpenRoadNet output

3.2.3 PathNet

We have no ground truth “correct paths” associated with the input frames fed into the PathNet model. Therfore, PathNet was evaluated entirely qualitatively based on the general coordinates of paths in the JSON file, as there was no reasonable way to gather any meaningful data due to the rendering error. Random samples were chosen from different datasets and the coordinates were used to plot out a general direction for the path. In about 75% of the samples, this path followed a reasonable route for the car to take, as the path did not intersect with walls or objects (Figure 6). This could be considered a good result based on those samples and do instill confidence that the model can generate paths for the car to follow. This is not a conclusive result and more testing and evaluation should be done, but it does show the capabilities of the model. The average FPS was 199.

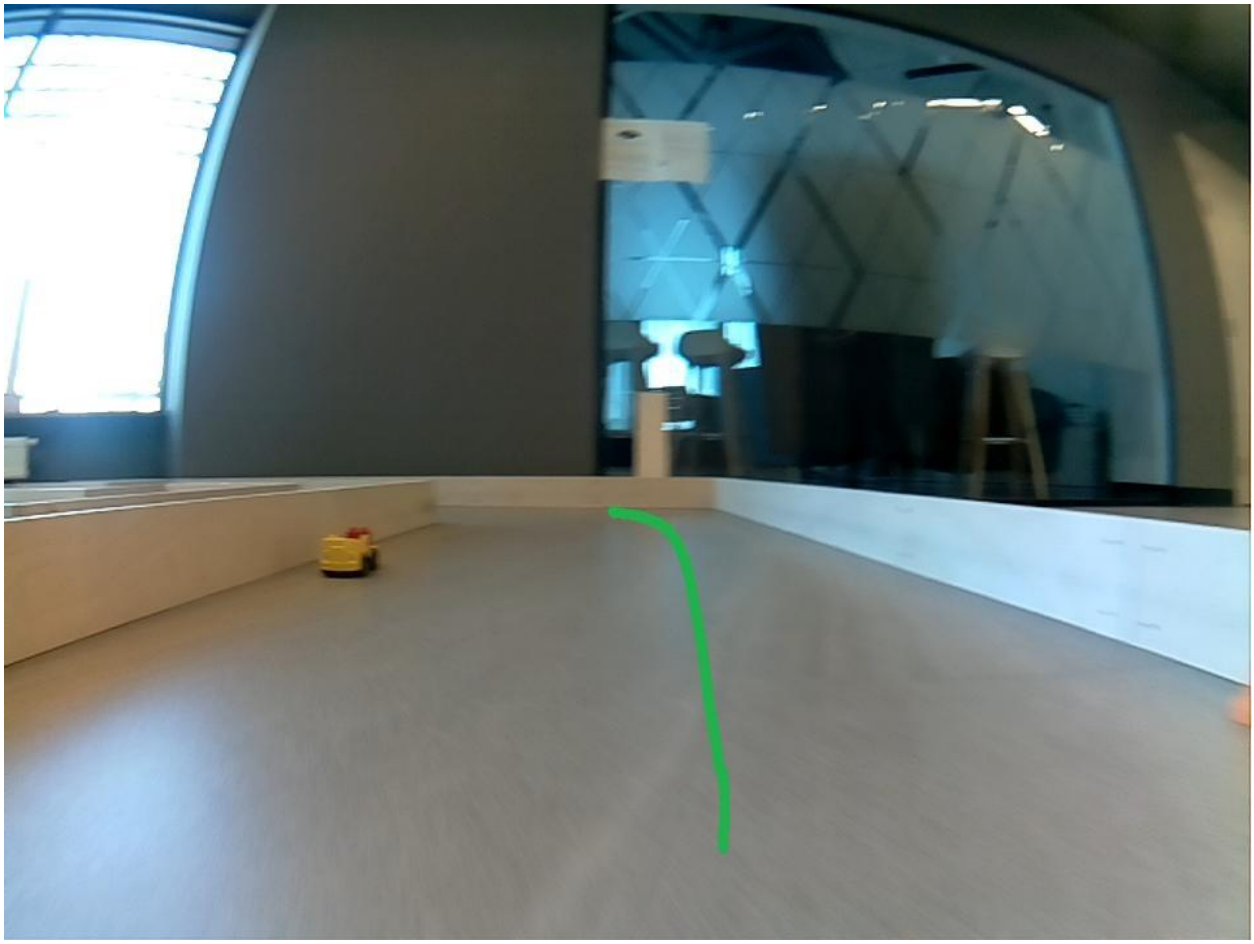

Figure 6. PathNet example output

3.3 Collective results

In general, the models related to path planning and available road area did perform well and got usable results, but the results seem to be highly variable across input images. The output varied greatly between consecutive frames, even if the car was moving so slowly that there was no perceivable change in the frame. In particular, all of the models seem to be very sensitive to lighting conditions and reflections, as in frames where the lighting changed, (i.e the car turned around a corner towards a light source) the models had a very low chance of success.

4. Conclusion

Self-driving cars have been a field that many have been following in recent years. This technology has the potential to revolutionize transport in the near future, both for the benefit of the average user and in the transport industry, such as freight transport. The aim of this thesis was to analyze and evaluate the capabilities of neural network based image processing in Nvidia Drive in unideal conditions. If Nvidia drive works on toy data, we can build on top of its results and make toy cars drive autonomously on the track. Having learned to do that for toy cars, these networks also work for real cars and students are better equipped to try it with a real car.

The main question was whether Nvidia Drive is capable of identifying key elements in an environment that only roughly resembles the data the neural networks were originally trained on. Given a toy town, toy cars, and toy people a human can instantly identify those elements and has all the information required to complete the task. However, neural networks are known to not always generalize well to inputs outside the distribution their training data originated from. Testing showed, that Nvidia Drive is currently not capable of identifying objects that only loosely resemble the real world. This lack of generalization ability to objects that humans would clearly identify and avoid is a serious problem also in the real world. The unpredictability of people makes it entirely possible, that some objects encountered on the road will more closely resemble the toy versions, than real versions. This could be either due to damage or vandalism on traffic signs, modified cars, halloween, or new fashion trends.

Though object detection may not be capable enough to work in unideal conditions, path drawing and drivable area detection produce usable results that could be made use of in a self-driving system, even in a toy town. For example, based on the drivable area prediction, a driving system could be designed that just keeps the car going towards the farthest point of the drivable area while keeping a safe distance from non-drivable parts (obstacles, walls).

In the author’s opinion, the usability of Nvidia drive is bad from a developers' perspective and should be improved to encourage developers to use it to further the research on self-driving vehicles. Nvidia Drive had a lot of features, but the installation and usage of these were unnecessarily complicated and long. Also, the documentation does not cover all versions and is missing for some versions, which is why the rendering issue came to be.

References

1. McKinsey logistics “Distraction or disruption? Autonomous trucks gain ground in US logistics” https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/distraction-or-disruption-autonomous-trucks-gain-ground-in-us-logistics (01.08.2022)
2. Tesla Autopilot. https://www.tesla.com/autopilot (10.05.2022)
3. A Basic Introduction To Neural Networks. The University of Wisconsin-Madison https://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html (10.05.2022)
4. Neural Network. *Merriam-Webster* https://www.merriam-webster.com/dictionary/neural%20network (10.05.2022)
5. Tampuu A. Artificial Intelligence Entry-level Course. University of Tartu https://courses.cs.ut.ee/2020/Tehisintellekti_algkursus/Main/PARTIITehisn%C3%A4rviv%C3%B5rgud (10.05.2022)
6. Krizhevsky A. Sutskever I.; Hinton, Geoffrey E. ImageNet classification with deep convolutional neural networks https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (24.05.2017)
7. Fujiyoshi H., Hirakawa T., Yamashita T., Deep learning-based image recognition for autonomous driving. *IATSS Research,* Volume 43, Issue 4,2019, Pages 244-252, https://www.sciencedirect.com/science/article/pii/S0386111219301566
8. Cunninghawm W., Cars drive autonomously with Nvidia X1-based computer, 2015 https://www.cnet.com/roadshow/news/cars-drive-autonomously-with-nvidia-x1-based-computer/ (10.05.2022)

9. Nvidia driveworks documentation, Getting Started https://docs.nvidia.com/drive/archive/driveworks-3.0/dwx_devguide_getting_started.html (10.05.2022)
10. Nvidia Drive https://developer.nvidia.com/drive (10.05.2022)

11. Badue C., Guidolini R., Carneiro R.V., Azevedo P., Cardoso V.B., Forechi A., Jesus L., Berriel R., Paixão T.M., Mutz F., Veronese L.P., Oliveira-Santos T., De Souza A.F., "Self-driving cars: A survey", *Expert Systems with Applications*, Volume 165, https://www.sciencedirect.com/science/article/pii/S095741742030628X (05.08.2022)

12. R. Ravindran, M. J. Santora, M. Faied and M. Fanaei, "Traffic Sign Identification Using Deep Learning," *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, pp. 318-323. https://ieeexplore.ieee.org/document/9070858
13. Donkey Car store page for raspberry pi camera. https://store.donkeycar.com/products/wide-angle-raspberry-pi-camera-for-donkey (10.05.2022)
14. Tammvee M. Openpilot supercombo model. GitHub https://github.com/MTammvee/openpilot-supercombo-model (10.05.2022)
15. Dual Boot, Technopedia https://www.techopedia.com/definition/3353/dual-boot (10.05.2022)
16. Autonomous driving lab, Nvidia labelers https://gitlab.cs.ut.ee/autonomous-driving-lab/wp4/nvidia_labeler (10.05.2022)
17. Nvidia driveworks documentation, DriveNet https://docs.nvidia.com/drive/archive/driveworks-3.0/drivenet_mainsection.html (10.05.2022)
18. Nvidia driveworks documentation, OpenRoadNet https://docs.nvidia.com/drive/archive/driveworks-3.0/openroadnet_mainsection.html (10.05.2022)
19. Nvidia driveworks documentation, PathNet https://docs.nvidia.com/drive/driveworks-3.5/pathnet_mainsection.html (10.05.2022)

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Illimar Laanisto

1. **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, my thesis

Testing Nvidia Drive for small cars in toy town

supervised by Ardi Tampuu,

2. I grant the University of Tartu the permit to make the thesis specified in point 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 4.0, 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 from *dd/mm/yyyy* until the expiry of the term of copyright,

3. I am aware that the author retains the rights specified in points 1 and 2.

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

Illimar Laanisto
08/08/2022