Software Engineering & Cloud Computing

Assignment 2

Erik Börve
borerik@chalmers.se

August 16, 2024

1 Introduction

The aim of my project is to produce a safe, and economically-efficient autopilot for heavy vehicle combinations (HVCs), essentially trucks of varying shapes and sizes. To this end, we are investigating optimization-based methods for path planning in dynamics environments, where the "dynamic" part of the environment mainly stem from other road users (cars, pedestrians, bicyclists, etc.). To find a suitable path through traffic for ourselves, we importantly need to be aware of the plan and intentions of other road users. To this end, machine learning methods have shown great prowess in recent years. Hence, our aim is to combine stochastic machine learning methods with conventional optimization methods. Leveraging stochastic machine learning, we may predict the behavior of the dynamic environment, and obtain suitable probabilistic safety guarantees for its' performance. In combination with optimization methods, such as *Model Predictive Control*, we may find the best possible way to operate the vehicle for economical efficiency. The project is an industrial collaboration with a large vehicle manufacturer.

Our project does not aim to directly alter any vehicle components on a hardware level. I.e., we aim to utilize the same sensor-base and actuators. The ideal final product of my PhD project would be an over-the-air software update, that runs "of-the-shelf" in all of our existing vehicles. Further, as of today, all of my research is also being done in simulation environments that I have implemented myself. Currently, I mainly work in python, C++ and Matlab. Hence, high-quality software-development is already of high importance in my day-to-day work, and will be crucial in the success of a potential future implementation of my research.

2 Roberts Lectures

I started my PhD right after my masters so I have only worked in a "real" software development teams during internships and summer works. Hence, I would say that I have very limited knowledge of how software engineering actually works in practice. All though I would like to think that I have quite some experience on the computer science side, I feel that I lack the "reality" part of SE as Robert put it. Hence, I found this part quite interesting, both in Roberts lectures and in the preparatory readings. More particularly, I found it interesting how small part the ML code was in real ML systems. My work naturally also includes, Data collection, Data verification, feature extraction, and analysis tools, but their relative size to the ML code is much more comparable. For example, many open-source data sets that I utilize have already been thoroughly prepared and require little effort in therms of collection and verification. Other of the real-world ML-system is not even included, e.g., monitoring or serving infrastructure. If I where to make a practical application of my work, I would definitely have to spend significant effort in familiarizing myself with these processes.

Another part of the lectures I found interesting was the verification and validation. Naturally, I also do verification and validation in my own research, e.g., though generating test cases that I know should yield a certain outcome. However, I do not think that I have a systematic and robust method for performing this process. Usually, I come up with test cases on-the-fly, which might not be robust enough, or even miss the main hazards. One challenge in this regard is that I do not have enough resources to get a second opinion on the code that I write, i.e., I am not able to perform the code review process. However, perhaps this is a process that I need to implement to some extent in my work. When it code to testing, I think I could benefit from being more systematic with unit, integration, system and acceptance testing. For starters, I usually do not provide clear document of my test cases, this small change would clearly allow me to be more systematic about my verification and validation testing. Every time I define a new function or update an existing one, I could force myself to think of at least 1 test from the four categories and clearly document how the test is done, what it checks, and what the result should be. In conclusion, I think that many of the principles that Robert presented could be useful already now in my research project and I hope to try out some in the near future.

3 Guest Lectures

Per Lenbergs's lecture focused a bunch on the behavioral aspects of software engineering, and operational management in general. I found this really interesting as it was not a topic that I had heard about or thought much about before hand! In particular I found it interesting that such aspect could have such a large impact on the success, or failure of software development in teams. After his lecture, I definitely think that I should have bigger consideration for the group dynamics, already right now, when supervising students in M.Sc. theses, but also in some potential future where I end up managing a teams. Not only do I feel that his research could be useful when building/managing teams, but it also feels useful to consider as an employee in a team. To take some (extreme) examples from his slides, I can easily see how my role as a software developer in a project could change if all other members of the team are having a threesome with each other.

Dhas's lecture focused mainly on the role of large language models (LLMs) in software engineering. All thought the slides are not available on canvas, I happen to be a colleague of him, and have some of his slides available in my brain. I find this topic quite interesting, and it's clear that there is a large potential for integrating these ML-methods in many areas in DevOps. For starters, the Github co-pilot is already a common occurrence in the daily work of many software engineers. However, one of the interesting questions is whether or not we really can trust an AI to generate code for safety critical systems, such as in Dhas's case of heavy vehicles. Clearly, we do not yet have fully automated software engineers and there is still a large need for human oversight. Dhas's work with generating automated test cases is another, perhaps even more safety-critical, example where this applies. SafetyOps is a time intensive process that many managers would wish to make more productive. However, such a process may be pointless if it is not done rigorously. One can for example think of the common case where a software engineer chooses to get help from some LLM to generate the function code. If then the function testing is also done with an LLM, how do we guarantee that the (potentially safety-critical) flaws in the first LLM, are not similarly overlooked in the second? So far the answer seems to be continuous and rigorous human-oversight, but perhaps someone like Dhas will find a solution in the future.

4 Welcome Your New AI Teammate: On Safety Analysis by Leashing Large Language Models

The paper [1] investigates how large language models can be applied to automate function testing. The part that peaked my interest about this paper in particular is that the function in question is related to Advanced Driver Assistance Systems (ADAS), which currently is the highest level of automated driving that is encounter in most modern vehicles. More specifically, the paper investigates an ADAS function called "Collision Avoidance by Evasive Maneuver" (CAEM). The aim of such a function is to intervene when it notices that a human driver is making dangerous decisions, for example, drifting out of the lane margins into oncoming traffic. In such a scenario the goal of the CAEM is to bring the vehicle back into the current lane safely.

Clearly, such a function has the potential to cause serious negative consequences if it does not work as intended. Hence, it is essential for software developers to create detailed safety requirement specifications. An initial part of this process is often to perform a "Hazard Analysis & Risk Assessment" (HARA). In short, the goal of a HARA is to identify potentially hazardous events, assign them to a certain category, and finally specify safety goals, such that the consequences of the events can be prevented entirely, or at least mitigated. However, ensuring that the safety analysis meets the required standards requires a lot of man hours, which poses challenges for achieving rapid safety operations ("SaftyOps").

The paper investigates this problem by proposing a pipeline that sequential prompts an LLM to perform different tasks that typical are done by a human in a HARA. To this end, the paper engineers 6 different prompt-types to achieve the following goals; 1.) Identify hazards in the functions. 2.) Identify critical driving scenarios. 3.) Identify hazardous combinations of function hazards and critical driving scenarios. 4.) Classify their severity. 5.) If sever, identify an appropriate safety goal. 6.) Cluster the results into the appropriate categories. The results are then presented in a CSV file, such that it is suitable for human evaluation. The researchers conclude that this process could reach the requirements of industry standards and experts, however also remarked that it was only an initial study, and more work was necessary. They also emphasize that the use of current LLMs for safety critical applications, always require cautious human oversight.

This paper is directly relevant for my work, as any practical application of it, would need to go through the same, or even a more involved safety analysis process. I would personally say that it is not as relevant from a pure research perspective, as it does not contribute to pushing the boundaries of autonomous vehicles (the best way to get published in my research area). However, I believe that it could be very beneficial for me to be aware of where these boundaries are from a safety, and even legal standpoint. This could help guide my research in a direction that eventually yields a product that is possible to realize. Further, being a busy PhD student with limited experience of industry standards, it would be incredibly convenient for me to directly get initial suggestions for specifications from an LLM. For example, when generating and evaluating scenarios in my simulation environment, I can simultaneously investigate if my driving strategy is able to live up to the industry requirement standards. I could then adapt our use of ML and optimization such that these standards are fulfilled. I find this idea quite interesting, and I believe that it could spawn a bunch of interesting innovations.

Lastly, considering a larger AI-intensive software project, such as actually integrating my work in a real autonomous vehicle, I believe that the results of this paper could be very beneficial. Competition in the autonomous vehicle space is fierce and many companies are pushing the boundaries of what is reasonable to be the first to achieve a fully autonomous vehicle. Hence, I believe that rapid SafetyOps will be a crucial competitive advantage, and perhaps even a requirement for a successful autonomous vehicle company. If LLMs can be utilized for this purpose, in any way, I am confident that every single successful company will be utilizing it to some extent. If I wish to develop successful products in this space from on my own research, I would likely want to follow the lead of such companies.

5 Worst-Case Convergence Time of ML Algorithms via Extreme Value Theory

The second paper [2] investigates methods for predicting the worst-case convergence time (WCCT) of different ML-algorithms. Knowing the worst case training time can be useful for many different reasons. For example, ML training (especially large models like language models) requires large computational resources, which may pose environmental risks due to the large energy consumption of deploying code on many GPUs. The primary reason I choose this paper is because they investigate some algorithms that I myself use in my research, such as standard logistic regression and deep neural networks (DNNs). More specifically, they investigate the convergence time of ML training and ML "inference" using extreme value theory (EVT). To this end, the aimed to utilize maximum likelihood methods to estimate the parameters of a generalized extreme value distributions (GEVs). This was done by first classifying a threshold value for extreme values, like the 95%-percentile, and then fitting the three parameters of the GEV-based distribution to the extreme values (location μ, scale σ and shape ξ). The paper then investigated three different questions. 1.) Are GEV-based WCCT predictions better than other existing methods? 2.) How good are they at estimating training convergence WCCT? 3.) How good are they at estimating "inference convergence" WCCT ?

The first question was investigated by looking at a linear support vector machine example. In this case, they got quite nice results. The GEV-based prediction gave better predictions in 83% of the test cases and the predictions where often significantly closer than other estimates. However, for the second question, they look at logistic regression, and found that the GEV gave a very poor estimate. The likelihood of observing their predicted WCCT in practice was estimated at 0.12%. They also looked at a multiple different methods and overall found that their method was "accurate" (defined as having the true WCCT within a 95% confidence interval of their prediction) in 57% of the cases. Lastly, they investigated "inference" of DNN methods by looking at ML-based controllers, trained to stabilize different physical systems, e.g, balancing a pole on a car. They randomized the initial state of the system and looked at the time it took for the system to stabilize around the reference, e.g., having the pole standing straight up. In this case they got a bit better results and found that their method was accurate in 75% of the investigated cases.

All though I found the topic and method of the paper quite interesting, the results did however not end up being that useful for my research, simply because it showed quite poor performance on the methods that I was interested in. Regardless, I think that WCCT is something that could be useful for me to consider in my research. If I end up working with training large ML-models, I should definitely consider the potential environmental impact it could have. There are not only ethical motivations for these considerations, but also legal considerations for my company. For example, a company has limitations on the amount of energy it is allowed to consume, which is dictated by strict legal requirements on environmental impact. Using the WCCT to over-estimate the predicted run time, and further the predicted energy consumption, could provide a conservative estimate of my resource consumption, which might be useful for decision makers and emission reporters that govern my project, and the company. In this regard, I think I could improve my research by, first and foremost investigating smaller networks, that hence take less time to train, and generate less emissions. Larger networks, should only be utilized when it is necessary.

Another separate concern I have with the paper is that I do not think that the experiment used to answer research question 3 was properly formulated. This since I believe that they did not investigate what they refer to as "inference convergece" of DNN methods. The paper itself states that the WCCT of inference with ML-methods (widely defined as the process of generating a prediction from a given ML-algorithm), does not vary significantly for DNNs. Indeed this is correct as the inference step is simply a series of matrix multiplications. What the paper then decides to investigate is pre-trained DNNs and how they effect the performance in physical systems, from different initial conditions. This has little to do with the inference of the DNNs, but rather the dynamical properties of the physical system, and how they are effect when combined with the control law that is generated by the DNN. As an example, there are no guarantees that the obtained control law would yield a stable system, and hence it might never converge to the reference and no convergence time may be obtained. Including these aspects in the research question would have been a much more accurate description of their experiments and results. Regardless of the accuracy of their description, the results does say some interesting things about the reference-convergence time of the combination of DNN-based controllers with physical systems, and GEV seems to be a good prediction method for the WCCT. However, the results are not useful for my research with DNNs as I initial hoped when reading their problem description.

If I where to eventually deploy a large software-project around my research, I definitely think it would interesting to employ some of the concepts from WCCT estimation. It is first and foremost interesting to estimate the environmental impact, as previously discussed, but it is also relevant for optimizing the training time of any potential ML-algorithms. One may for example use it to over-estimate the time that it takes to complete different tasks involving ML. This could be used to appropriately allocate resources and better manage the company. For example, if many large models need to be trained in a short time period, the GPU resources needs to be allocated efficiently to minimize the elapsed time. If the WCCT is estimated accurately, we could obtain guarantees for the time of the respective training tasks, and determine the best time and location for the training of each network. Overall, it could aid in getting a better estimate of the time-to-market of the ML-based software product that we aim to create.

Bibliography

[1] Ali Nouri et al. “Welcome Your New AI Teammate: On Safety Analysis by Leashing Large Language Models”. In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 2024, pp. 172–177.

[2] Saeid Tizpaz-Niari and Sriram Sankaranarayanan. “Worst-Case Convergence Time of ML Algorithms via Extreme Value Theory”. In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 2024, pp. 211–221.