WASP Software Engineering module assignment

Tiago Rodrigues de Almeida

August 6, 2024

Abstract

Robots and other intelligent systems navigating in human-centered and complex dynamic environments must be able to predict the future actions and intentions of the agents around them. These agents are heterogeneous, often influenced by factors such as their type, speed, and intentions. For example, pedestrians and cars in a shared road scenario typically show different speed patterns, with cars generally moving faster than pedestrians. The goal of my thesis is to investigate the potential for incorporating implicit heterogeneity into motion prediction systems for mobile robots. Incorporating agent heterogeneity into the motion prediction task offers a number of advantages. First, it improves the accuracy of predictions by bringing them closer to the ground truth predictions, as the model can more effectively capture the subtleties of each agent group. Second, it reduces the overall uncertainty in motion prediction. That is, by stratifying behaviors into more homogeneous groups, the model can focus on a narrower range of actions, thereby refining its predictive capabilities within each group. This stratification not only allows for more accurate trajectory prediction, but also facilitates safer and more efficient navigation for mobile robots in diverse and dynamic environments. Technically, I use machine learning techniques because of their proven effectiveness in handling complex and large data sets.

1 Two principles from the lectures

1.1 Software testing

Software testing is a systematic process of evaluating the quality of a piece of software through experimentation to ensure that it performs as expected under various conditions. This principle is related to my thesis, not because of a direct focus on software testing itself, but because it serves as a fundamental practice in any work, task, or project that involves software components. In the context of machine learning, open source code promotes validation, transparency, and accelerates progress in the field.

Researchers must thoroughly test their code before releasing it, ensuring that their results are transparent and replicable. However, with this transparency comes the responsibility to ensure that all code components work as expected and have been rigorously evaluated. In machine learning related products, one can use different datasets to ensure that the models, loss functions, and evaluation pipelines are robust and behave as expected with respect to different inputs. In particular, before starting a machine learning project, and whenever possible, I build my own "artificial" dataset that allows me to draw simple and intuitive inferences from its outputs. For example, while developing a new mechanism for incorporating heterogeneity in motion prediction, I create a dataset with predefined groups of trajectories with different characteristics, such as different velocity, acceleration, and orientation profiles. This controlled setup, similar to the "test oracle," allows me to draw clear and intuitive inferences from the model's outputs, ensuring that the system accurately identifies and models the different groups of trajectories as intended. This is a preemptive testing step for machine learning practitioners that can avoid the ultimate compounding complexity in machine learning projects with real-world data.

Furthermore, one of the most challenging aspects of machine learning is debugging the training and inference of the model, as there are numerous potential sources of error, including corrupted data [1], bugs in the code, poorly defined hyperparameters, loss functions, and evaluation metrics. Therefore, another action I take when developing a machine learning project is to encapsulate and test each step in the pipeline separately. These steps include: (1) data preprocessing, loading, visualization, and

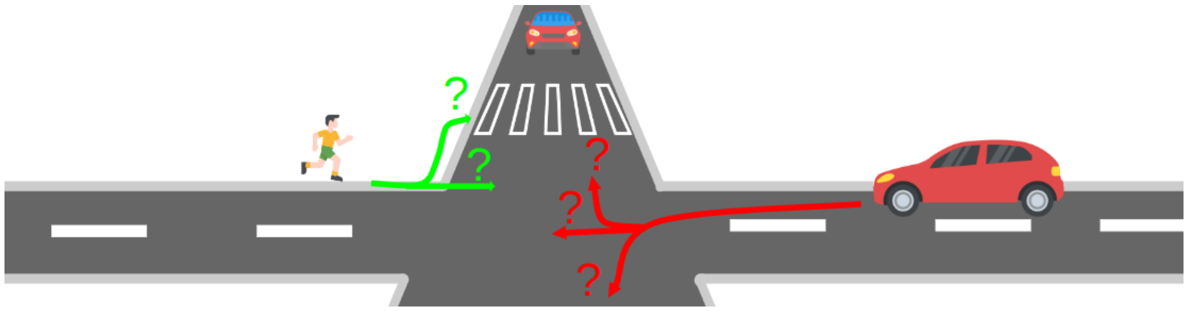

Figure 1: Multiple plausible futures in a motion prediction scenario showing the uncertainty and complexity of the task.

validation (e.g., using [1]The Data Linter), (2) data modeling (including hyperparameter tuning), and (3) model evaluation/inference. I prefer to use "artificial" simple data so that I have full control over all components and outputs.

1.2 Testing invariants and metamorphic testing

Motion prediction is an inherently uncertain [2] and multimodal process [3], where the same observation can lead to a range of plausible future scenarios [4] (see Fig. 1). This variability can make it difficult to establish unambiguous test oracles. Given the inherent complexity of motion prediction, it is reasonable to conclude that finding ultimate motion prediction test oracles can be a tedious task. Current human motion prediction benchmarks typically use a best-of-many evaluation approach [5]. A single observed trajectory can lead to multiple future outcomes. These pipelines generate K future trajectories and select the closest one for evaluation. Because K is often set to a large number, it is difficult to know whether the predictions are accurate or simply produce a wide range of different predictions, some of which may contain significant outliers [4].

Metamorphic testing provides a viable alternative to address these challenges. Metamorphic testing is a technique that addresses untestable software components by exploiting user-defined properties of the system [6]. These properties, known as metamorphic relations, are conditions or transformations that should hold between pairs of related inputs and their corresponding outputs. For example, one might apply certain trajectory augmentations as metamorphic relations that should not lead to different predictions, such as small rotations or translations of the observed motion. Another metamorphic relation could be the introduction of minimal noise into the input observation, which should not affect the prediction. A potential issue when using this technique is to ensure the alignment of additional input modalities, such as the semantic map of the environment. This avoids any impact of the augmentations applied to the observed trajectories on the spatial layout introduced in the dataset. For example, if a pedestrian is walking on the sidewalk in the original dataset, after applying the test transformation, the transformed trajectory must remain within the sidewalk area.

2 Two ideas from the guest lectures

2.1 VOLVO - Unit testing with generative AI: trust/no trust?

Nowadays, Large Language Models (LLMs) have become the subject of numerous debates about the potential displacement of human work. In this talk, the guest from VOLVO presented an analysis of the use of generative AI for unit testing, which prompted a discussion about the trustworthiness of this technology in relation to this specific task. In my opinion, these technologies have the potential to reduce the burden on software engineers in the unit testing process by allowing them to focus on the less critical aspects of the task. For example, by generating initial test cases and templates, LLMs can significantly reduce the time spent on boilerplate code. However, I also argue that the ultimate responsibility lies with the human engineer, who must ensure that the code is robust, accurate, and secure. Moreover, because these models are trained on large amounts of data, they can help identify

[1]https://github.com/brain-research/data-linter

potential edge cases that may not be immediately obvious to humans, improving the overall quality and reliability of software. However, it is important to maintain a critical perspective on the output of LLMs, as they may still propagate biases or inaccuracies present in their training data.

Ultimately, the decision is up to the software engineer. If someone prefers to code the unit tests, an LLM can be used for soft validation and documentation generation (as an assistant). I do not have a software engineering background, so I rely a lot on these types of technologies to generate and test my code. At the end of the day, I am the one who validates the code generated by the LLM, as it should be. This allows me to spend more time on the most important tasks, such as ideation and research.

2.2 SAAB - Behavioral Software Engineering

The SAAB talk discussed the use of machine learning in the daily tasks of a software engineer in a more holistic way. Specifically, Behavioral Software Engineering (BSE) was introduced as the intersection of software engineering and psychology, where human and behavioral aspects are integrated into software development processes [7]. The guest gave a questionnaire to the software engineers working at SAAB about their feelings about the use of AI/ML in their daily work. The answers seemed rather vague, but it is worth noting that the software engineers seemed uncertain about their future.

As a bridge to my topic, my research on trajectory modeling can be used to model and predict software engineer behavior, such as task switching behavior. These predictions can then feed downstream mechanisms to, for example, improve engineer efficiency. Furthermore, my interest in heterogeneous motion patterns can also have some impact on BSE, since there should be some heterogeneity in the behavior of software engineers. This can be used in a twofold framework: (1) to help understand different but common behaviors of software engineers; (2) similar to motion prediction, it can help predict the behavior of software engineers.

3 Two papers

The first paper I analyzed is entitled *"An Exploratory Study of Dataset and Model Management in Open Source Machine Learning Applications"* [8].

Core idea(s) and importance to the engineering of AI systems This paper focuses on dataset storage, versioning, and integration issues in open source ML applications. The authors identified common storage locations for datasets and models and highlighted the lack of version control and proper update actions. I find the data collection process a bit ambiguous, as the authors took the repositories with 10 or more stars on Github "to ensure their quality," which can be considered a low score in terms of Github metrics, especially for machine learning projects. First, the authors found that the file system is the most popular choice among application developers for storing ML artifacts (i.e., datasets and trained models), followed by the proper repository, and remote storage. This is quite relevant to the development of AI systems, as the sample studied shows that the most common storage location (file system) can lead to availability issues. Ultimately, this can lead to lack of visibility and slowness in the research process. Moreover, for URL-based data access, the authors found several problems such as: not guaranteeing continuity (e.g., university domain), errors, and unspecified data file. All this contributes to a rather inefficient AI environment, where the most important artifacts are not made publicly available or carefully stored. In addition, most of the datasets and models stored in the repositories are rarely updated during the development of the application. This demonstrates the lack of integration of version control systems in open source machine learning applications

Relation to my research The reason I focused on this work is because it is directly related to my research, as I have also published a dataset [9]. This dataset consists of different data collections for human and robot motion in the same scene. We have several data modalities: human trajectories, eye tracking data, and robot sensor data such as video streaming and lidar data. We also have a [2]github repository where we show users how to manipulate the data: preprocessing, loading, and visualization. The github repository contains the URL to the [3]Zenodo

[2]https://github.com/tmralmeida/thor-magni-tools
[3]https://zenodo.org/records/10554472

where practitioners can download the dataset. Zenodo is an open data repository that allows for versioning. So every time we upload a new version of the dataset, the versions are automatically updated and the corresponding link is created (currently our dataset is on version 1.1).

Furthermore, this study can provide some pointers or negative actions (that I should not do) when making available models checkpoints. Since I work directly with machine learning, storing artifacts properly is quite important to ensure that other researchers have access to the latest versions.

Improve a larger AI-intensive software project My research can be integrated into a broader project that aims to analyze, model, and understand human motion in different contexts, such as shopping malls, airports, home, and industrial environments. These contexts affect human navigation in different ways because they have different characteristics. For example, an airport can be considered a more "unfamiliar" environment for the human, and therefore his motion can be considered more stochastic than in his home, where he is completely familiar with the spatial layout. As the paper addresses current practices for storing datasets and points out several issues, and this hypothetical broader project implies a huge data collection, storage locations, version control systems, and proper data management would be paramount in such a project. Perhaps a dedicated set of tools for integrating the related artifacts would be one of the outcomes of the project.

My research adapted/changed based on the paper As I mentioned in the last question, perhaps one of the outcomes of the project would be a dedicated set of tools for integrating large amounts of data, as would be the case in the broader project. To do this, we would need software/data engineers to help us build the infrastructure to make such large amounts of data available to other researchers. In this direction, my first thought is to build an infrastructure around cloud services (e.g. AWS or Azure), which already contain some interesting features for data storage and manipulation.

The second paper I analyzed is entitled *"Uncovering Energy-Efficient Practices in Deep Learning Training: Preliminary Steps Towards Green AI"* [10].

Core idea(s) and importance to the engineering of AI systems This paper proposes the use of an energy efficiency metric to evaluate deep learning models, in addition to the common "accuracy" metric, provoking a shift towards "Green AI". To this end, the authors compare strategies used at different stages of training a deep neural network: hyperparameter tuning and model layers. Specifically, the authors compare Bayesian optimization, random search, and grid search for hyperparameter tuning. With respect to model layers and the corresponding complexity, the authors find strategies to reduce the complexity of the model and thus be more efficient while maintaining the accuracy of the model. The authors conclude that the most efficient hyperparameter tuning method is Bayesian optimization, providing guidance for AI system engineers. More importantly, it is crucial to think about these issues today because training large language models is so expensive, both in terms of money and energy (see this [4]blog post). Finally, the authors have shown empirically that increasing the number of convolutional layers (the most computationally expensive layers) does not always mean increasing accuracy. Therefore, they suggest reducing the number of convolutional layers to a point where the accuracy is still within a reasonable range.

Relation to my research The paper's focus on making deep learning models more energy efficient is very relevant to my research, especially since I use deep learning methods. For example, this is critical when running these models on mobile robots with limited power [11]. By using the techniques from the paper, I can make my prediction models less energy hungry, which is a big deal for extending the battery life and efficiency of these robots. Basically, it's about making smart choices about model complexity to keep energy consumption low while still getting good performance.

Improve a larger AI-intensive software project My research can also be integrated into a larger AI-intensive software project, such as a mobile robot in human-populated environments, as in

[4]https://lambdalabs.com/blog/demystifying-gpt-3

the Spencer project [12], where the robot should be able to predict the trajectories and poses of people in an airport. In these cases, where the deep learning models are to be used in autonomous mobile robots, the trajectory predictors need to be as efficient as possible to save power for other sensors. In my research, I have applied Bayesian optimization to the hyperparameter process of Ray [13]. However, I have to admit that model complexity was never the goal of my research, my concern is to create a set of baseline methods that have similar amount of parameters whenever possible and can run in CPU so that everyone can use them.

My research adapted/changed based on the paper If the goal of my research would be to eventually deploy it in a mobile platform where power is limited, then it would make sense to use a model pruning/quantization approach that reduces the number of parameters and thus the power. This alignment with "Green AI" principles can facilitate the deployment of theoretical deep learning methods in real-world platforms.

References

[1] Elizamary de Souza Nascimento, Iftekhar Ahmed, Edson Oliveira, Márcio Piedade Palheta, Igor Steinmacher, and Tayana Conte. Understanding development process of machine learning systems: Challenges and solutions. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–6, 2019.

[2] Guopeng Li, Zirui Li, Victor L. Knoop, and Hans van Lint. Unravelling uncertainty in trajectory prediction using a non-parametric approach. *Transportation Research Part C: Emerging Technologies*, 163:104659, 2024.

[3] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 0–0, 2019.

[4] Tiago Rodrigues de Almeida and Oscar Martinez Mozos. Likely, light, and accurate context-free clusters-based trajectory prediction. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1269–1276, 2023.

[5] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2021.

[6] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortés. A survey on metamorphic testing. *IEEE Transactions on Software Engineering*, 42(9):805–824, 2016.

[7] Per Lenberg, Robert Feldt, and Lars-Göran Wallgren. Towards a behavioral software engineering. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE 2014, page 48–55, New York, NY, USA, 2014. Association for Computing Machinery.

[8] Tajkia Rahman Toma and Cor-Paul Bezemer. An exploratory study of dataset and model management in open source machine learning applications. In *2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 64–74, 2024.

[9] Tim Schreiter, Tiago Rodrigues de Almeida, Yufei Zhu, Eduardo Gutierrez Maestro, Lucas Morillo-Mendez, Andrey Rudenko, Luigi Palmieri, Tomasz P. Kucner, Martin Magnusson, and Achim J. Lilienthal. Thör-magni: A large-scale indoor motion capture recording of human movement and robot interaction, 2024.

[10] Tim Yarally, Luis Cruz, Daniel Feitosa, June Sallou, and Arie van Deursen. Uncovering energy-efficient practices in deep learning training: Preliminary steps towards green ai. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 25–36, 2023.

[11] Yongguo Mei, Yung-Hsiang Lu, C.S.G. Lee, and Y.C. Hu. Energy-efficient mobile robot exploration. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 505–511, 2006.

[12] Juergen Gall Umer Rafi, Bastian Leibe and Ilya Kostrikov. An efficient convolutional network for human pose estimation. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 109.1–109.11. BMVA Press, September 2016.

[13] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.