Models for Autonomous Vehicle Software Engineering Systems: A Literature Review

Connor Raymond Stewart
David R. Cheriton School of Computer Science
Univeristy of Waterloo
Waterloo, Canada
crstewart@uwaterloo.ca

***Abstract*—The literature review presented is to highlight and describe essential concepts in the field of Machine Learning as they relate to the continued research and development of self-driving autonomous vehicles. Background concepts, software system integration, and software system implementation are included in the report herein. General examples of self-driving autonomous systems are highlighted throughout the paper.**

Keywords—Big Data, Machine Learning, Artificial Intelligence, Robustness, Neural Networks, Validation, Ethics, Autonomous Systems, Autonomous Vehicles, Self-Driving Vehicles, Knowledge Representation, Verification.

I. Introduction

Significant highlights include learning the background concepts behind machine learning methods, problems related to ethics, and issues with machine learning adversarial input and robustness. Systems-level details include machine learning models, machine learning requirements engineering, and machine learning software engineering models. Implementing machine learning systems focuses on elaborating some new foundational concepts, leading to topics related to the validation of machine learning systems, and finishing with the ethical implementation of self-driving autonomous machine learning systems.

The report is structured as follows: three sections describe concepts related to the literature review topic. The first section introduces fundamental concepts for self-driving autonomous systems, AI methods, ML methods, AI ethics, and ML robustness introduced in the second section of the paper. Background information, knowledge, foundations, and practices related to autonomous vehicle systems are discussed. The third section discusses the construction of ML systems through software systems, requirements engineering approaches, and software engineering models. To determine how systems operate and the methods used to build them, the third section lets readers learn how software systems are integrated. Topics in the third section include methodology, author data, approaches, and frameworks that help integrate ML, AI, and big data concepts for self-driving agents. The description of software systems used for autonomous systems summarizes successful ML models, requirements engineering approaches, and software engineering models used in previous research. The fourth section introduces the topic of AI-based software implementations for self-driving autonomous vehicles. Implementation theories related to the legal and functional practice of autonomous systems are introduced to showcase fundamental concepts, validation systems, and ethical practices in the field. Various AI paradigms – such as logical systems and classification systems – related to ML knowledge representation are introduced under the context of autonomous systems. Implementation theories related to communication, irrationality, and cooperation of drivers along the road are also discussed to present the decision-making of autonomous driving agents on roadways. Validation systems are discussed in the context of reducing undesirable states for ML models. Machine learning ethics are introduced to ensure autonomous vehicles' desired legal, ethical, and moral behaviours are ensured on roadways. The implementation of autonomous vehicle movement systems is discussed under the context of motion planning systems. Finally, data validation systems to process invalid data from big data systems or sensory systems were included, along with the topic of testing ML models for errors. The fourth section aims to learn the implementation details of systems and what was used to build concepts related to logical, environmental, and big data-based systems related to self-driving vehicles.

II. Background

Background concepts related to self-driving autonomous systems and machine learning methods are included in this section. Topics of relevance are extracted from related works and showcased below.

A. Machine Learning Methods

Big data technologies and various implementations are discussed in an ACM panel describing the impacts and problems associated with newly assembled big data systems [4]. According to Vipin Kumar, big data from sensors and devices is very different from other datasets that the big data community has previously dealt with [4]. Interconnected devices produce disorganized data leading to challenging data analysis problems that previous large-scale datasets did not have [4]. Another pundit – David Blei – commented that it is not just individual data that helps with recommendations but the aggregated data from mass devices which helps [4]. As more data is generated from devices, the concern of individual rights, access to information, and individual privacy is raised. Michael Stone-Breaker questions the feasibility of mass data integration by

lending an example of a doctor's x-rays for medical diagnosis. However, mass hospital data integration would allow social gains by comparing diagnoses with morbidity rates from patient x-rays. Problems with data formatting, encoding, terms, privacy, and legalities are raised [4]. Daphne Koller claims biases with data interpretation can be resolved with randomized case controls. Michael Stone-Breaker points out that trouble with predictive models arises from the biased humans who build them [4]. As David Blei points out, we are experiencing a transformative time for machine learning and artificial intelligence as new ideas lead to breakthroughs in reinforcement learning, deep learning, and optimization [4]. Vipin Kumar states that new types of sensors and communication technologies – such as small and inexpensive satellites that can monitor the earth at new spatial and temporal resolutions – are truly transformational [4].

Deep neural reasoning is discussed in detail by H. Jaeger (2016) in a brief publication on the topic of new neural reasoning and neural network systems for computing and AI [8]. Reasoning can be explained as rule-based mental manipulation of symbols according to ancient and modern viewpoints [8]. Human brains consist of neurons that operate through the exchange of electrical impulses rather than symbols, leading to the neural-symbolic integration problem, which remains unsolved [8]. *Graves et al.* have been working on potential solutions to neural-symbolic integration problems using machine learning systems to impart symbolic-reasoning mechanisms to an artificial neural system such that the system can learn through symbolic-reasoning rules from examples [8]. Another requirement for reasoning is working memory which digital computers fulfill with RAM since computers execute programs and functions by building information in working memory with ever-changing combinations [8]. Aristotle's definition of syllogisms transitions to modern models of programmable computers with the Turing Machine, as it formulates that the behaviour of computers at any moment is determined by symbols, which is observing and its state of mind at that moment [8]. A working understanding of working memory works in human biological systems or artificial neural networks still needs to be improved [8]. Developments show similarities to digital computers, including neuro-computational systems that contain non-symbolic modules that operate by exchanging streams of purely analog activation patterns like those from biological brains.

In contrast to digital computers, which need to be programmed, neural systems must be trained [8]. Training involves presenting solved examples to the system and letting it adapt its internal neural wiring to get closer to the correct solution [8]. The neural system's trainability is a product of its smoothly adaptive internal neural wiring [8]. The wiring is due in part to a differentiable function with tens of thousands of adjustable parameters [8]. Deep learning systems have been designed to resolve computational problems associated with many adjustable parameters in differentiable neural computers [8]. Practical applications of deep-learning systems have excelled in tasks requiring little to no working memory, like image recognition [8]. Future applications of differentiable neural computers include big-data applications with rational reasoning components like semantic text analysis [8].

B. Artificial Intelligence Ethics

The topic of AI technology regulation and liabilities is the focus of a publication put forward by Oren Etzioni (2018). Novel disciplines like AI are subject to political interference and distortion, and government regulations are needed to prevent harm [10]. To prevent regulating AI research and stifling innovation *Oren Etzioni (2018)* suggests regulating AI applications rather than AI research. AI leaders are expected to gain significant economic and geopolitical advantages but defining and categorizing AI is difficult due to its rapid evolution over the years [10]. Harm is challenging to characterize in algorithms, and AI systems do what is needed to achieve goals regardless of their implications [10]. AI cannot have common sense under vague concepts like harm [10].

Five guidelines for regulating AI applications so regulatory bodies can ensure AI applications do not harm people were suggested by *Oren Etzioni (2018)*. Firstly, not weaponizing AI technology as autonomous weapons would permit armed conflicts on greater scales and timescales than ever [10]. Not weaponizing AI would help prevent the creation of weapons of terror that could cause harm to despots or terrorists [10]. Secondly, AI is subject to the same laws as human operators – there is already a well-developed body of laws – such that human operators or manufacturers are liable to ensure that an AI does not hurt anyone [10]. Thirdly, AIs should not disclose that it is not human since bots can mislead people, and political bots can influence people by commenting on new articles to generate propaganda [10]. Fourthly, AI systems cannot retain or disclose confidential information without approval from the source of information – to prevent bad actors from misusing private data from smart devices [10]. New legislation must address privacy issues associated with smart device data [10]. Fifthly, AI must not increase bias in systems. Predictive algorithms generalize to create predictions, and mathematical methods that ensure predictive algorithms do not induce bias are required [10]. Topics related to racial bias include racial patterns where introducing extra bias can institutionalize bias and produce morally reprehensible results [10].

AI must be accountable, interpretable, and transparent so that people can understand their decisions [10]. AI algorithms can determine who is scrutinized, and predictive algorithms can calculate future risks for inmates [10]. Historical data has the marks of history, which include past trends, and we must consider historical data's origins and patterns to determine whether we want the trend to continue [10].

Creating new bodies of regulations for AI is likely impractical, so using existing laws and regulations is the best approach for AI technologies [10]. Philosophical problems such as the trolly problem can be addressed because driverless cars reduce the number of people hurt during accidents [10]. There is an ongoing dialogue between various institutes to formulate AI best practices and ethical guidelines [10].

C. Machine Learning Robustness

Machine learning robustness and concerns related to deceiving AI systems is the topic of concern for Don Monroe (2021). Deep learning systems can classify images, translate text, and learn to perform tasks but still make trivial mistakes that threaten security [5]. Vulnerabilities present problems for

applications to medical, legal, financial, and military systems [5]. A tank owned by the US was pained with pictures of cows which caused AI vision systems to misclassify a tank as a herd of cows [5]. Researchers are exploring ways to increase machine learning robustness to adversarial attacks and to understand the principles and limitations of machine learning approaches; so that they can be implemented into multilayered security strategies to slow attackers [5]. Image classification is intuitive for understanding the deception associated with machine learning; adding a small number of pixels to an input image can cause unnoticeable changes to an image - resulting in unnoticeable changes to an image that can fool classifiers into declaring stop signs as speed limits [5]. Ultimately, deep learning systems base decisions on features different from what people consider essential for image classification [5]. Vulnerabilities in machine learning extend further than the domain of image classification, and some areas do not possess visualizable results [5]. Image manipulation is difficult to defend against since pixels have enough variability that attackers can manipulate whatever they want [5]. One can exploit the external gradients used by machine learning systems when training to generate adversarial images, so attackers may bypass classifiers during attacks using the learning algorithm [5]. Imperceptible images added to a school bus caused an image classifier to declare the bus an ostrich – which highlights problems associated with safety when using machine learning systems to classify vehicles [5].

Deception is a general problem for AI systems that use sensory inputs since evasion attacks – altering input to a classifier – can occur [5]. Poisoning occurs when attackers insert doctored data into training sets – but can only occur if attackers have access to the data [5]. Mislabeled data can move decision boundaries that separate different classifications [5]. Three attack scenarios – white-box, black-box, and gray-box scenarios – can occur with the number of data attackers who know about the internal design details of a machine learning system [5]. With white-box scenarios, attackers know the internal design details of machine learning systems and are used as a worst-case scenario analysis [5]. With black-box scenarios, security protocols obscure a machine-learning system's design and data to protect it from attacks [5]. Finally, a grey-box scenario occurs when partial information is known or inferred and exists between the white- and black-box scenarios [5].

Attacks against one system will commonly work against other systems with different internal structures since classifiers tend to learn the same correlations from datasets [5]. The transferability of attacks highlights problems associated with using large open datasets, as they are all potentially vulnerable to the same biases [5]. Adversarial training is a process where systems are trained using adversarial examples to increase the system's overall robustness [5]. Examples have been noted where spoofing of sensory input is more complicated by cross-referencing with multiple sensors [5]. Robust machine learning is a tool for computer security, and current research is focusing on producing theoretical guarantees of how long systems can hold out to inform the design of secure layered defences [5].

III. Integration of Software Systems

Building machine learning systems has focused much on technical-level implementations in recent years. A broad understanding of software engineering integration for machine learning systems is still a topic of active research. Below highlights some important topics related to the integration of software systems related to machine learning systems with specific focuses on self-driving autonomous vehicle systems. Highlights and descriptions of software systems used for self-driving autonomous systems are used to summarize successful ML models, requirements engineering approaches, and software engineering models. Learning how software systems are integrated helps determine how authors got systems to operate and what methods they employed. Highlighting methodology, author data, approaches, and frameworks can help with integrating fundamental concepts related to machine learning, artificial intelligence, and big data into a more comprehensive model for self-driving autonomous agents.

A. Machine Learning Development Models

Machine learning processing relating to the development of a maturity framework for use cases and implementations was put forward by *Akkiraju et al. (2020)*. It is known that existing machine learning models fail to convert business requirements to technical requirements, which presents issues related to the practical adoption of self-driving autonomous vehicles [11]. Reinterpretations of the software capability maturity model (CMM) for ML model development are the focus of the presentation by *Akkiraju et al. (2020)*. The software community has generated lifecycle management theories and practices over the years, but two outstanding issues have resulted in a new ML maturity framework [11]. Issues include that ML systems differ from traditional software and that ML solutions that work for software enterprises require different solutions from the literature [11]. ML system management developments are needed since they are not deterministic like classical systems – they are probabilistic [11]. A modern version of the CMM maturity model can define business requirements as data requirements work using five process maturity levels for software [11]. In general, we notice a specific machine learning model lifecycle that can be mapped onto the production of self-driving autonomous vehicles [11]. Note that some topics are obtained from the article *The integration of machine learning models in complex autonomous driving systems (ADS)* – a research topic presented by *Peng et al. (2020)*:

1. Model goal setting is initiated by offering managers. Autonomous vehicle production managers can initiate new processes to produce vehicles [11]. Monitoring the quality and performance of models across versions to maintain alignment with predefined goals and targets is also the focus of the offering managers and goal-setting team [11].

2. Content management strategies are employed to identify suitable training sources. Content managers can establish governance around data legalities and lineage management [11]. Detecting sources of bias in data samples and attempting to incorporate established ethical norms and legal

practices into data collection for autonomous vehicles would be included in this step. Manual code inspection can also lead to biased results and is difficult to resolve with systemic approaches [15].

3. Data collection and preparation are essential in training AI models. Data leads allow for AI data collection and labelled data preparation [11]. Checking data quality, labelling data, and establishing metrics for data are included in this step. Previous systems have relied on multiple data sources to generate final predictions [15].

4. Feature preparation involves preparing features from collected data [11]. Preparing features from collected data to initiate the training of models is determined in this step [11]. The types of features generated depend on the ML paradigm and implementation. For example, developing vectors associated with Bellmen Coefficients – as was the case with the model generated by *Chen et al. (2021)* – would be included here.

5. Model training is engaged by a training lead that decides algorithms for experimentation. Determining what framework to use – like Pytorch or TensorFlow – is included in this step, along with parameters related to neural networks [11]. Fine-tuning a system for autonomous vehicles would occur during this step. Twenty-eight ML models are widely used in Apollo and can be categorized as either traffic light perception, lane perception, obstacle detection, or trajectory prediction systems [15]. Determining the type of model to train for specific cases can become very long with complex applications such as autonomous vehicles.

6. Test leads use testing and benchmarking. Testing the effectiveness, safety, robustness, adversarial defences, ethical requirements, and legal requirements of the system would occur during this step [11]. Audits conducted by agencies providing oversight or competitor agencies would also occur during this step [11]. Passing tests related to vehicle safety would need to occur during this step. Empirical studies on ML engineering systems reveal that bugs in ADS systems scatter across systems and show the necessity of providing quality assurance for practical ML systems [15].

7. Model deployment is where a deployment lead makes vital decisions related to a system's hardware and engineering constraints [11]. Considering the engineering constraints required to keep a self-driving system operational, up-to-date, safe, and upgradable may be difficult for future manufacturers. Using real-time training on self-driving vehicles would be resource intensive, meaning active tasks occurring in a self-driving vehicle would need to be considered in terms of hardware constraints. For example, the empirical analysis of Apollo studied machine learning roles, models, and complexity to provide analysis [15].

8. AI operations management ensures AI models improve over time by learning from mistakes [11]. Managing iterations, versions (with possible regional preferences), and updates for self-driving autonomous vehicle software models present a final task for future developers [11]. Different iterations and versions of software related to a self-driving system would require maintenance and organization. ADS's aim to produce autonomous vehicle driving processes, but so far, ML model testing has only occurred at the unit level rather than the collaborative level [15].

B. Machine Learning Requirements Engineering

Machine learning requirements engineering has been chiefly focused on the intersection between requirements engineering and machine learning leading to a scarcity of research on the topic of requirements engineering related to the development of machine learning systems [9]. Requirements engineering for machine learning-based systems is presented through a systemic mapping study by *Villamizar et al. (2021)*. It aims to outline research contributions and contemporary gaps for future research to fill [9]. Systemic mapping (SM) studies are designed to provide comprehensive overviews of research areas to establish if research evidence exists on topics to provide indications of the quantity of evidence [9]. Several research questions related to the objective should be asked when implementing an ML system, and we will borrow these to assess a research paper by *Censi et al. (2019)*:

1. What requirements for engineering contributions have emerged to support the software development of ML-based systems [9]? We could begin by listing requirements for engineering contributions to self-driving autonomous vehicles such as those listed by *Censi et al. (2019)* – who defined a rulebook as a pre-ordered set of rules.

2. What requirements of engineering topics do the contributions address [9]? *Censi et al. (2019)* produced a rulebook to minimize collisions and maintain driving performance during split-second dilemmas.

3. What quality characteristics do the requirements engineering contributions consider for ML-based systems [9]? For example, *Censi et al. (2019)* sought to prevent collisions and make the right choices when driving, to follow ethical and legal requirements.

4. What are the reported challenges and research direction on the interplay between RE and ML-based systems [9]? It is noted in an article about liability and ethics for machine learning that defining what the car is supposed to do in specific scenarios is one of the primary challenges for developing self-driving cars [16].

5. What is the research-type facets of the contributions [9]? An example is the fact that there are many different types of facets for research on self-driving

cars, including temporal, deontic logic, LTL, and ML systems [28], [16], [15].

6. What kind of empirical evaluations has been performed to assess the contributions [9]? Examples of experiments include collision testing, simulations, lane changes, and merging [16], [15], [11].

Vogelsang & Borg (2019) conducted interviews with four data scientists to better define characteristics and challenges related to requirements engineering in machine learning to help better understand machine learning elicitation, specification, and assurance of requirements [3]. Quality targets for ML systems are essential since functional requirements are needed to satisfy specific rules and expectations for autonomous vehicles [3]. Explainability is difficult for ML systems since humans cannot interpret their functionality like conventional software [3]. Explaining what a model learns is essential to justify an agent's actions, especially in a legal trial [3]. Considering self-driving agents would need to defend their actions the same way a human would, we need to explain why an agent made a specific choice on the road. Explaining single predictions is also crucial for a model since it must abide by specific rules for specific scenarios [3]. Typically, models that people prefer map relationships between input and output, but the issues are that this lowers the predictive power of models [3]. There is also no specific focus on what situations demand explanations, meaning requirements engineers should specify requirements for specific applications [3]. Legal requirements for self-driving vehicles need to be specified for specific scenarios before implementation, and there is no general model for the process. Ensuring freedom from illegal discrimination can be implemented by requiring systems to use training data prepared so that it does not contain any protected characteristics (like race or gender) or so that they can analyze trained models to find essential features in training data [3].

We must know what an ML model will deliver before developing it [3]. Training data needs testing like code since it affects the outcome of developed models, unlike traditional software engineering [3]. Data quantity and quality should be considered using evaluation metrics to assess datasets before training [3]. Requirements engineering entails multiple steps, including elicitation, analysis, specification, validation, and verification [3]. Elicitation involves the process of a requirements engineer identifying all possible data sources that can help ML systems provide excellent and robust results [3]. Data scientists should be consulted to elicit requirements for specific data and its processing, and legal experts should be consulted to determine constraints concerning how data is allowed to be used [3]. Analysis of ML system requirements is needed to define and discuss the performance measures and expectations by which ML systems are assessed [3]. Performing exploratory data analysis where the focus is on data understanding and selection is required by requirements engineers to facilitate the discussion between customers and data scientists [3]. Requirements specification should contain statements about expected predictive power expressed by performance measures discussed during the elicitation and analysis stages [3]. Expected performance should be capable of being immediately checked after the training process, whereas performance at runtime can only be expressed as desired performance that can be assessed during operations [3]. Verification and validation of data are crucial to define actions that ensure training data corresponds to actual data [3]. A checklist of measures to consider during ML system operations is provided by *Breck et al. (2019)*. Requirements engineers also need to identify conditions for data anomalies that can lead to unreasonable behaviour of ML systems during runtime [3].

C. Machine Learning Software Engineering Models

A systemic mapping study by *Martínez-Fernández et al. (2022)* introduced software engineering for AI-based systems, which resulted in the mapping of multiple software engineering approaches for AI systems [13]. Intense focuses on qualities related to safety have been noted for cyber-physical systems like autonomous vehicles – however, gaps remain for less studied properties such as usability, portability, and aspect [13]. Inherent characteristics of AI-based systems, like explainability or transparency, require researchers to ensure the maturity of industry-specific applications [13]. It was noted that software design articles focusing on designing for specific parameters focused on safety-critical domains where unsafe domains had significant impacts, with many lending applications to autonomous vehicles [13]. Authors in a study on autonomous vehicle control proposed design strategies to be used at an architectural level for autonomous vehicles to facilitate the systems' development, analysis, and safety level [21]. The implementation of an independent module known as autonomous vehicle control (AVC) is used to interact with the vehicle's systems and create a protection layer independent of the vehicle's system [21]. The system is designed to be useable by any autonomous system and support individual testing as a result [21]. The result is the proposal of a safety standard for autonomous vehicle software or is used to suit existing safety standards for road vehicles [21]. Many verifications and validation systems focused on verifying cyber-physical systems for autonomous vehicles [13].

Furthermore, incompleteness with training or testing data has significant challenges leading to problems related to executions needing to be fully represented in training sets for autonomous vehicle systems [13]. New design methodologies and algorithms for autonomous vehicles are proposed and discussed by *Lan et al. (2018)*, including verification methods like precise computing and lower bound estimation [22]. Testing benchmarks such as vision benchmark suits, automatic test generation based on convergence criteria, and test suite evaluation are also included [22]. Finally, the use of machine learning in autonomous vehicles and a lack of safety certification are touched on by *Salay et al. (2017)*. An analysis of the impacts of ML and implementation approach on ISO 26262 safety lifecycle and questions of what should be done to address them result in a set of recommendations on how to adapt the standard to accommodate ML systems [23]. A summary of the recommendations includes [23]:

1. Identifying hazards resulting from behavioural interactions between humans and ML systems to determine areas that should be mitigated by system design [23].
2. Determining fault and failure modes where ML components have development lifecycles different

from regular software, using ISO 26262 to address ML lifecycles explicitly, and using fault detection systems customized to the lifecycle [23].

3. Using training sets with different safety requirements, as presented by ISO 26262, is necessary since ML models are trained from inherently incomplete data sets [23].

4. ML usage should be limited to the component level since the end-to-end approach challenges the assumption that complex systems are modelled as stable hierarchical decompositions of components with their functions [23].

5. Finally, required software techniques include techniques for various stages of the software development lifecycle as stipulated in ISO 26262 [23]. Analysis shows that some techniques remain applicable to ML components, and others could be readily adapted [23]. It is recommended that requirements be expressed in terms of intent and maturity of techniques rather than specific details to remove bias [23].

Advances in machine learning have forced organizations to evolve their development processes, resulting in a new nine-stage workflow process informed by prior experiences developing AI applications [12]. A nine-stage workflow process informed by prior AI-applications experience has been produced by *Amershi et al. (2019)*. Three areas of the AI domain make it different from existing software application domains: discovering, managing, and versioning data, model customization and reuse, and management of distinct models [12]. A machine learning model workflow consists of the following stages, with applications applied to autonomous vehicles [12]:

1. During model requirements, developers determine what machine learning models are feasible and what models are helpful for the product [12]. Determining the size and scope of a self-driving vehicle AI would be included in this scenario. Determining the types of roads (i.e., only highways), compatibility features (i.e., does not work in parking lots), and safety features (i.e., do not use near schools) would be included.

2. During data collection, teams integrate datasets into their model [12]. Finding datasets representative of different communities, driving styles, and age groups would be appropriate at this step.

3. Data cleaning involves removing inaccurate, noisy, or unnecessary data from datasets [12]. Removing invalid entries or outlier values from roadway vehicle datasets is included in this category.

4. Data labelling assigns ground truth labels to records [12]. Humans can audit sets of traffic images or sensor data values to apply specific values or attributes for the ML model to integrate with later.

5. Feature engineering refers to activities performed to extract and select informative features from machine learning models [12].

    a. The selected models are trained and tuned during model training on the clean data and their labels [12]. Collecting the data and configuring the machine learning model features would be included in this step.

    b. In model evaluation, output models on tested and safeguard datasets are evaluated with pre-defined metrics [12]. For autonomous vehicles, metrics would likely be audited by legal or administrative authorities and testing along specific ethical or legal dilemmas would be included in the datasets. Due to autonomous vehicles being a safety-critical domain, extensive testing would need to occur with human test drivers on the road.

    c. Model deployment occurs when a model is deployed on specific target devices [12]. Updates for existing autonomous vehicle driving systems, along with new software packages, would be included in this step.

    d. Finally, model monitoring occurs with real-world execution to test for possible errors [12]. Big data collection from cities and mass sensor integration for self-driving vehicles would aid in this final step. Collecting debug information, data associated with collisions involving self-driving vehicles, and big data from self-driving vehicular sensors would help significantly with monitoring autonomous vehicle models.

The development of eleven foundational artificial intelligence engineering practices is showcased by *Horneman et al.* in a short publication on the topic [1]. Several foundational practices related to self-driving vehicles should be addressed. Firstly, securing AI systems by applying highly integrated monitoring and mitigation strategies should be performed since autonomous vehicles have many interacting AI and ML models inside [1]. Secondly, self-driving AI models should be built around checkpoints beforehand since accountability and verifiability will be needed at multiple steps of the decision-making process for a vehicle [1]. Constantly validate and evolve models and architectures automatically based on performance metrics like trip time, collision rates, rider satisfaction, etcetera, to improve system performance [1]. Continuous monitoring for specific applications may be necessary on the part of the rider in a self-driving vehicle [1]. A rider may need to monitor a self-driving vehicle to intervene during certain maneuvers like highway pileups, parking, or non-paved driveways to prevent the AI from doing something illegal. A self-driving AI may function like cruise control, meaning that human monitoring or intervention could greatly help during specific driving transitions. Implementing loosely coupled solutions to AI systems so that they can be changed and modified fast without impacting other components is a good design principle [1]. Considering that the boundaries between components in AI systems deteriorate faster than in traditional software systems, it is essential to attempt to partition different software components to prevent coupling between AI and ML models [1].

Requirements for AI systems scale up significantly throughout the system's lifespan, and developers underestimate the number of resources needed for a system nine out of the ten times a system is used [1]. Evaluating every aspect of a system for potential ethical issues and accounting for societal values when implementing a system at a design and architectural level is recommended to prevent any strange occurrences from occurring with self-driving vehicles [1].

IV. Software System Implementation

Due to the complex interactions within such systems, software system implementation for large-scale machine learning and artificial intelligence systems still needs to be better understood and is understudied in literature. Below is an introduction to the topic of AI based software implementations, with a focus being presented on self-driving autonomous vehicles. Artificial intelligence paradigms related to machine learning knowledge representation, such as deontic logic, are discussed to elaborate on systems of information representation available to autonomous vehicles. Fundamental implementation theories related to human irrationality and cooperation for autonomous vehicles are discussed to show how autonomous vehicles can act as typical driving agents on roadways. Fundamental concepts related to validation systems and ethical practices are shown with information regarding their uses or experimental details showing how they performed in test environments. The goal of learning how software systems were implemented and what was used within the systems was used to explain foundational concepts for logical, environmental, and big data-based analysis for autonomous vehicles. Learning how data was used to produce reliable results and the ethics related to vehicles, including how vehicles were made to make reliable trajectories and maneuvers, is explored. Future research directions are lightly described, along with current impacts for autonomous vehicles.

A. Autonomous Vehicle Cooperation and Irrationality Models

In an article presented by *Chen et al. (2021)*, the analysis and correct modelling of human irrationality can enable better communication between humans and self-driving autonomous vehicles on the road since irrational agents communicate more information and reward than perfectly rational agents when correctly modelled [18]. The critical point from the article is that robots can infer reward functions by observing human behaviour and assuming humans are approximately rational [18].

Systemic irrationality in humans is pseudo-irrational since irrational humans can communicate more information about reward functions than perfectly rational humans [18]. We can learn that irrationality is a suitable vector for information spread and that irrational robots can leverage their behaviour to communicate with human participants. Rational humans always behave the same regardless of speed preference, yet myopic humans behave differently depending on speed preference [18]. We can learn human reward functions through observations, as human behaviour can increase the information being conveyed to other drivers [18]. We can infer human reward parameters by observing how they behave on the road. A self-driving agent can leverage other humans on the road since they behave in predictable ways, given a reward function. For example, more aggressive drivers are likelier to cut other people off. Suppose an agent can infer human reward states. In that case, they can navigate by driving differently to provoke a human into making a different choice than they would have if the self-driving agent did not consider other people's objective functions.

The framework of the system uses components of the Bellman update modified to cover different types of irrationalities [18]. Irrationalities include: changing the max into a SoftMax to capture noise (Boltzmann), changing transition functions to seize optimism and pessimism or illusionary control, changing reward to capture a nonlinear perception of gains and losses (Prospect), changing average reward over time to a maximum (External), and changing the discounting to capture more myopic decision-making [18] The illusion of control – modelled as a Bellman update – calculates the rate at which people overestimate their ability to control random events [18].

Modelling of optimism and pessimism is performed by using an optimism parameter to show – when ω moves to positive or negative infinity – the human becomes more optimistic or less optimistic, respectively [18]. Myopic discounting is used to model human myopia and is performed by decreasing gamma in the Bellman update [18]. The formulation presented gives a good idea of how to model human driving performance and preference for a self-driving vehicle. Training self-driving models based on various human strategies in a simulated environment and deploying them on the road is one way to create a driving system. Another implementation could include a real-time learning agent that can learn from other drivers on the road by using the framework presented to interpret human behaviours and objective costs.

What the framework shows us is that certain irrationalities result in better inference when compared to rational agents. The author's experiments show that log loss of the posterior as a function of irrationality parameters - in the random MDP environment - shows that every irrationality except prospect bias has parameter settings that outperform the rational planner [18]. Therefore, modelling for optimism, illusionary control, and myopia increase objective function accuracy. Myopic value iterations produce different policies, whereas rational experts always detour around obstacles and attempt more significant rewards [18]. Myopia causes the myopic expert to attempt to get to a smaller source when it is non-zero [18]. Optimism bias causes humans to take the chance for more significant rewards. Boltzmann rationality produces different policies where larger theta values generate policies closer to rational humans as the Q-value difference increases [18]. Assuming noisy rationale leads to poor inference compared to true modes with random MDPs versus Boltzmann models showing that not knowing correct irrationality harms inference [18].

The authors find that if the robot accounts for human myopia – but gets the type of myopia wrong – results are still significantly better than assuming Boltzmann rationality [18]. Log loss models under parameter misspecification show that AI agents perform better than assuming Boltzmann-rational just by getting the type of planner right, even if the exact parameter is wrong [18]. We can learn that AI agents can perform better than assuming Boltzmann rationality by assuming irrational behaviour. We can derive two things; one is that, as a rule of

thumb, Boltzmann rationality is not a good model of human behaviour, and two is that modelling irrational human behaviour can significantly increase the information communicated to AI agents.

User studies with human participants are ideal for learning how to generate irrational human behaviour for an AI agent to model. Getting people to display myopic behaviour to train AI agents helps generate accurate models for reward objectives. The authors of the paper note that future work is needed to replicate the results in more realistic domains since the algorithmic models were tested in simulated environments [18]. Attempting to use data in a real work environment with a self-driving vehicle may be difficult since noise from sensors and constraints with vision models may make replicating the results from the article difficult. The simulated environment was tested using a two-dimensional roadway simulator without elevation, and results were simulated from a birds-eye perspective. A vehicle may be obstructed in real-world environments, and factoring perceptual disturbances to the vehicular sensors may be an area of research to increase model robustness. Vehicle AI should learn perfect models of rationality and attempt to drive using rational behaviours. Humans display irrational behaviour, which is, in effect, pseudo-irrational since it performs better than purely rational behaviour.

The effects of communication on the road using irrational behaviour help to increase information communication between behaviours. As humans are not purely rational, modelling humans as rational produces poor results, and modelling humans as irrational produces better results than modelling them as noisy-rational [18]. Irrationality has a feedback effect on agent-to-agent interactions since a mutual transfer of information occurs between irrational agents, which does not occur with rational-to-irrational agent-to-agent interactions. Irrationality can inform human behaviour by increasing mutual information between reward and policy [18]. Irrational behaviour also leads to increased degrees of freedom for reward functions, as rational behaviour does not give many choices to an agent. The article can help generate models for human decision-making during real-time driving situations on the roads with human participants. Real-time planning should rely on irrational biases for AI-to-human and AI-to-AI interactions to increase information transmission between agents.

Future work can focus on implementations using more realistic testing environments than two-dimensional top-down perspective simulators. Operational constraints like viewing angles, elevation, perspective, sensor noise, and adversarial examples could be included. Real applications of the objective function need to incorporate noise from sensors and missing data from areas the vehicle cannot see. Planning reflex bias using the presented system is an excellent area to move towards since human reflexes on the road are developed and skewed towards nonrational behaviour. Autonomous vehicles will need to respond if a driver cuts them off or a collision is about to occur. Considering that human reflexes are also likely to convey relevant information to other vehicles on the road, testing how humans respond during possible collisions or aggressive driving conditions would make good training conditions for a future driving system based on the author's model. in all, the model presented helps with understanding how to get autonomous vehicles to navigate the road with human participants in a suitable format that facilitates the communication of information.

Autonomous cars' actions affect what other vehicles do in response, the consequences of which are approximated using a human as an optimal planner [25]. In the article *Planning for Autonomous Cars that Leverage Effects on Human Actions*, it is shown that Autonomous vehicles can use information from drivers to anticipate actions, for example, planning to merge ahead of humans when anticipating human breaking or backing up at four-way stops when anticipating human proceeding [25]. A multitude of planners is tested using user studies, including features for the boundaries of the road, features for staying inside lanes, and features for avoiding other vehicles [25]. By planning using dynamic systems, self-driving autonomous agents can elicit desired changes in human states [25]. Features associated with higher rewards are shown with warmer colours in the heatmap below [25]. Case studies show emergent AI behaviours and consequential human responses such as [25]. The case studies include A car merging in front of a human to slow them down, A car directing a human to another lane using its heading as leverage against the human driver, The car backing up to make a human proceed first at an intersection [25].

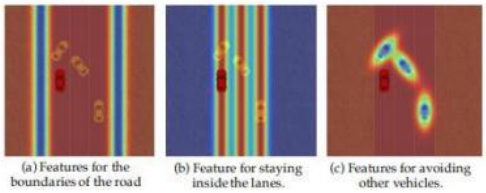


Figure 1: Heatmap of Features Associated with Reward [25]

Vehicles can make humans slow down, make humans go left or right, or make humans go first at intersections by changing the way they drive [25]. Using an interaction system, we can leverage this emergent behaviour with self-driving autonomous vehicles. If self-driving vehicles wish to minimize driving time en route to a given location or avoid a collision, they can leverage the above behaviour. Training a self-driving vehicle in a simulated environment to have a toolkit of various driving maneuvers can give a self-driving AI some leverage for human participants on the road. We can ultimately map reward states for making human agents cross intersections using the simulated model presented by the authors, as can be seen with the results from the author's experiments [25]:

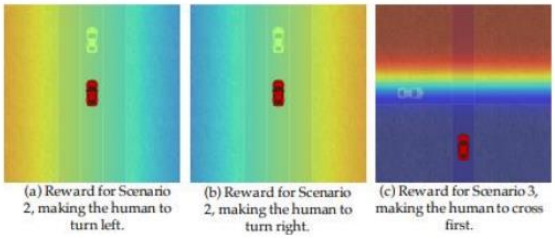


Figure 2: Heatmaps of Features Associated with Reward [25]

The authors perform user studies from the three scenarios above with singular manipulated factors and then measure the value along the user trajectory for the reward function [25]. It is

hypothesized that the method lets AI agents achieve their desired effects against human opponents [25]. Results show that human actions are deliberately influenced by AI agent movement [25]. The presented system can be leveraged to allow AI agents to control human behaviour, but some limitations remain. A testing environment should utilize more constraints such as viewing angles, elevation, adversarial examples, robustness constraints for decision-planning, and noise from vehicle sensors. However, if properly built, different types of driving behaviour could be tested for AI vehicles to learn how to control humans on the road. Parameters for using the techniques could be implemented into the AI systems driving module, and the AI could use the driving maneuvers as tools on the road. More so, simulated environments could allow AI systems to learn different types of defensive or offensive driving that may be difficult to test in real life. A planning system could be implemented using information from a simulated environment so that vehicles can move around human drivers based on the information it receives from their sensors.

Trajectories of human-driven vehicles from the second scenario show that AI agents can induce avoidance behaviour from users in a third lane by occupying two lanes [25]. Human cars cross the intersection before the autonomous vehicle in the third scenario when the autonomous vehicle affects the human to cross first [25]. Modelling different types of avoidance behaviour and mapping them onto a strategy system would be a good step for the self-driving system. Producing experimental data on different types of avoidance-based effects from self-driving autonomous vehicles can help generate models on leveraging human behaviour and is a possible direction for future research projects. One takeaway is the use of dynamics models where a robot can apply continuous controls which affect the state immediately through the dynamics model [25]. Model predictive control is used at every iteration and computes a finite horizon sequence of actions to maximize its reward [25]. For a robot to solve the finite horizon problem at each iteration, it would need access to what a human would do over the next N steps [25]. Since an agent cannot simulate those constraints, a vehicle must assume what a human would do [25]. Humans can take different actions depending on what a robot chooses, and it is assumed that humans are modelled as maximizing their reward function [25].

Interestingly, the authors chose to model humans as agents that rationally maximize their reward functions. The results from the article Human Irrationality show us that humans are not rational planners. Future research implementation for a project would need to correct the assumption that humans attempt to maximize their reward rationally since irrational behaviour increases information communicated between agents. The AI agent uses game-theoretic formulations to create predictions and relies on optimal planning [25]. The AI agent should include pseudo-irrational bias to improve communication, as presented in the first article. The authors use a short time horizon as an approximation since it results in the human not having access to a robot's entire plan, just its first few time steps [25]. The ideas of using a dynamic simulation environment are interesting as it allows for real-time simulation planning, which can be used for an analytic-planning module for a self-driving AI system.

A good idea presented by the authors is letting driver reward users access their reward function based on a reinforcement learning learned reward function from human data [25]. Such a system helps set a goal state. However, the authors, unfortunately, formulated the system as a rational optimization-based maximization which does not include irrationality presented by human drivers. The article shows that autonomous cars can generate human interpretable actions through optimization but needs to show how to formulate multiple human-driven vehicles interacting. A future point of research would be to show multiple-human drivers in a simulated environment. However, using optimization alone limits the ability to model a driving environment. The authors show that approximate solutions to optimize reward can be modelled in an analytical environment. A future project could use these models to predate the outcome of single AI-to-human driver interactions in constricted environments. However, introducing irrationality bias is a significant concern for the formulation presented by the authors. In conclusion, despite the limitations, it is encouraging to see autonomous cars generate human interpretable behaviours through optimization without relying on hand-coded heuristics.

A paper on the topic of A *Consciousness-Inspired Planning Agent for Model-Based Reinforcement Learning* introduces end-to-end model-based DRL agents that dynamically attend to relevant parts of their state during planning [26]. Bottlenecking mechanisms over set-based representations force agents to listen to only a few entities at each planning step [26]. Consciousness-inspired planning is modelled through three systems, an interpretable representation system, an attention-based bottleneck, and a decision-time planning system [26]. In interpretable representation, a system with set-based representations for latent codes is used by concatenating feature vectors and position codes to form set-based representations [26]. A set-based encoder is used as a concatenation that's treated as objects in a set to capture the features of observed entities [26]. A value estimator is designed by replacing the MLP before pooling with transformer layers such as multi-head self-attention (SA) with object-wise fully connected (FC) [26]. Set-based representations allow the creation of a system of features with corresponding positions in the system as a data structure. Value estimators are used to map observations, with a value estimator representing the permutation invariant set-to-vector architecture that maps latent states to value estimators [26]. Attention-based bottlenecks work with soft-selected objects into a subset with fewer objects through an attention mechanism [26]. A paraphrasing of the author's specifications is as follows:

> "Bottleneck stages are listed with operations coloured in purpose being conditioned on a chosen action. Firstly, a bottleneck set c_t is soft selected from the whole state (object set) s_t through semi-hard multi-head attention. Secondly, dynamics are applied to the bottleneck set c_t to from $\hat{c}_{t+1}$. Thirdly, the reward and termination signals are predicted from c_t, $\hat{c}_{t+1}$, and at. Finally, changes introduced in the $\hat{c}_{t+1}$ term are integrated with s_t to obtain the imagined state $\hat{s}_{t+1}$ with the help of attention. The two computational flows in stage three are naturally parallelizable." [26].

Selection and integration through attention mechanisms are performed as per the author's specifications. For selection via

Attention Mechanisms: "The bottleneck compressor obtains the bottleneck set c_t obtained by querying the whole set s_t with a learned query set of size k, using semi-hard multi-head attention. Selection is conditioned on the chosen action." [26]. For integration via Attention Mechanisms: "Designing bottleneck integrator shows that $\hat{s}_{t+1}$ is generated using the action augmented s_t to query the imagined bottleneck set $\hat{c}_{t+1}$. There is a similar operation of downscaling objects to features and copying positional trails." [26].

Finally, decision-time planning is achieved by running a tree search at decision-time using MPC with the earned model [26]. The organization of the components for CP agents shows that the transition model includes a reward-termination estimator, a dynamics estimator, and an optionally conscious bottleneck [26]. The system is akin to a human mind, with three layers corresponding to perception, inner representation, and the conscious planning models [26]. Experimentation shows that ODD performance under gradients of difficulty validates ODD generalization as they result in bottlenecked CP showing a clear performance advantage over UP [26]. The gradient under turn-and-forward tasks shows that the new set of dynamics can be seen as a composition of the original and is, therefore, easier to solve and more effective in planning [26]. Finally, performance under a gradient of difficulty in key-chest unlocks tasks shows that y-axis values are undiscounted cumulative episodic returns. The data is more differentiable than an original turn-or-forward setting, with patterns remaining consistent across all settings [26].

Figure 3: Top-Down View of System Architecture [26]

One limitation of the work is that experiments focus on mini-grid environments due to a need to validate the approach. Future works would extend those ideas to temporally extended models, which could simplify the planning task and be better suited as a conceptual model of C1. Finally, we note that the architectures used can require careful tuning for new types of environments. Apart from technical details, *Zhao et al. (2021)* show a viable system to simulate attentional systems that could be encountered with human drivers. The design lets planning systems generalize learning task-solving in unseen environments by focusing on relevant objects [26]. By generalizing objects that should be focused on, the authors create a system where self-driving agents obtain relevant details in real-time. Mimicking human-based attention-inspired bottlenecks is a good step toward self-driving AI systems, allowing AI participants to focus on relevant details. The system implemented is described in elaborate detail in the report, and only a selection of relevant topics is skimmed in the above summaries. Such a system could attempt to trim data from vehicle sensors and focus on details relevant to a driving module. A self-driving vehicle planning module could use the system to filter through the vast quantities of data it receives from its sensors. We could obtain a heap of data from vehicular sensors – or run a simulated environment which simulates sensory input – and use the framework presented by the authors to trim only relevant details for a logical-AI system to work on. Essentially, the system presented acts as a filter to trim raw data into relevant data. The relevant data can be fed into a logical real-time planning module for navigating the vehicle. Experiments on implementing the system should be discussed in more detail, but a general framework is required to elaborate.

In a paper on *Social Coordination and Altruism in Autonomous Driving*, autonomous vehicles obtain socially desirable behaviours by considering the utility of other vehicles around them during decision-making [27]. As ambiguity is involved with a human driver's cooperativeness with autonomous vehicles, the problem can become complex [27]. A social value orientation (SVO) ring can demonstrate different behaviour based on human-robot preferences to account for other behaviours [27]. Diameters of circles show the number of people who have corresponding SVO's [27]. The SVO angular phase quantifies an agent's level of altruism [27]. Note that the simulation includes three vehicle types: egoistic agents that only care about themselves, cooperative agents that care about themselves and their allies, and cooperative sympathetic agents that care about themselves, allices, and humans [27]. Below is a case study of an egoistic AV that disregards the merging vehicle and continues with its optimal speed such that merging vehicles cannot merge but still faces unsafe conditions [27]:

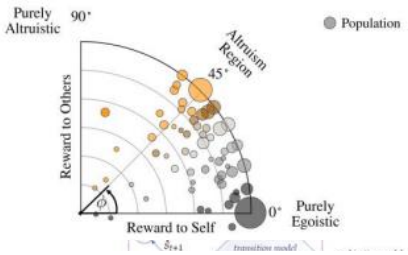
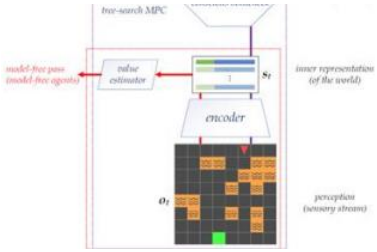


Figure 4: SVO Ring [27]



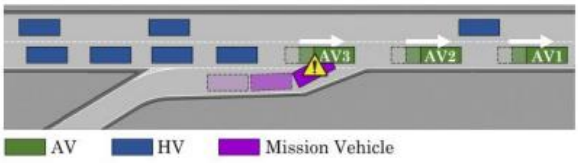


Figure 5: Diagram of Egoistic AV from Case Study [27]

Agents in the model learn complex tasks through experience and maximization of generic reward functions rather than requiring task-specific specialized program functions [27]. The system trains autonomous agents by decentralizing the reward function between learning to drive and learning social coordination [27]. The report finds that sympathetic, cooperative autonomous vehicles have better performance in creating socially optimal results when compared to egoistic autonomous vehicles [27]. Egoistic autonomous vehicles only account for individual interest leading to poor performance. Altruistic autonomous vehicles improve traffic flow and safety compared to egoistic vehicles since they can communicate and coordinate. Altruistic autonomous vehicles and human vehicles form alliances and affect the behaviour of other vehicles on the road [27]. The article shows that social coordination and altruism are needed for maintaining autonomous vehicles on the road since multi-vehicle environments cannot coordinate without altruism.

Furthermore, the idea of learning through a distributed reward system to allow learning policies from a clean slate was a good presentation. Future use of the research presented includes coordination systems for autonomous agents.

Finally, in an article on *Cooperative Autonomous Vehicles that Sympathize with Human Drivers*, we see that the system presented introduces altruistic behaviour for autonomous vehicles. A sympathetic, cooperative driving (SymCoDrive) paradigm trains scenarios only from experimental learning and without explicit coordination [28]. A multi-agent training and policy dissemination process is introduced to train an agent within a simulated environment and input data into a three-dimensional convolutional network which captures temporal dependences in a training episode [28]:

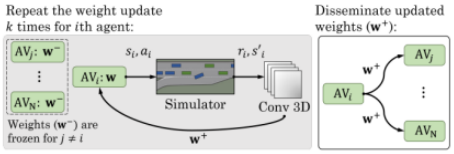


Figure 6: Multi-Agent Training and Policy Dissemination [28]

The three-dimensional Convolutional Feature Extractor Network (FEN) architecture is used to capture spatial and temporal information [28]:

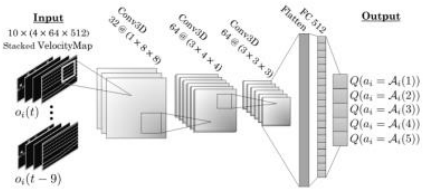


Figure 7: Three-Dimensional FEN Architecture [28]

Experiments show that between egoistic, cooperative-only, and sympathetic, cooperative autonomous agents, sympathetic and cooperative agents successfully merge to the highway while cooperative-only and egoistic against failing most attempts [28].

The article presents autonomous and human vehicles in the same environment and shows that merging behaviours result in agents learning altruistic behaviour. A framework for understanding and generating cooperative altruistic behaviour in self-driving autonomous vehicles is essential for future research in decision-planning systems for vehicles on the road. The examples in the article include merging simulations, a place where cooperative behaviour is fundamental to prevent crashing. Future research on more general situations (like driving down the road) or more specific situations (like in a roundabout) could be completed in a future project. One key takeaway is using a three-dimensional convolutional feature extractor network to capture spatial and temporal information. Learning to capture the movement of vehicles over time to build a reference for sympathetic and cooperative behaviour is very important and exciting for autonomous-vehicle systems.

B. Autonomous Vehicle Knowledge Representation

Driving style classification for the identification and use of self-driving autonomous vehicles is a new and underexplored area. *Mohammadnazar et al. (2021)* explore how driving style can impact safety, mobility, energy consumption, and emissions with large-scale connected vehicle systems by applying unsupervised machine learning [2]. Location-based big data and machine learning can classify driving styles on scales ranging from calm to aggressive [2]. A framework for developing Basic Safety Messages (BSMs) to quantify driving behaviour and classify driving styles using unsupervised machine learning has been created [2]. Temporal driving volatility is proposed as a concept that can be applied to vehicles to classify them into six volatility measures [2]. Scores are proposed to classify driving performance consistency across drivers, and different roadway types are shown to have different thresholds for aggressive or calm driving [2]. Several steps are used, including data acquisition, pre-processing, quantifying driving volatility, and driving style classification [2]. Clustering methods, such as k-means clustering, are used to aid with driving style classification [2]. Results from the papers show that driving volatility measures have higher values in commercial streets than residential streets and highways, so driving on commercial streets is more volatile than on other roadways [2]. Results from the study show that we can extract driving volatility features from raw BSM data to quantify driving behaviour at different road types [2]. We are provided with an essential first step towards understanding driver behaviours for self-driving autonomous vehicles so that they can understand the behaviour of human participants along the road.

The implementation and use of deontic logic analysis for autonomous systems' safety are explored by *Shea-Blymyer et al. (2020)* to help us better understand the logic of how agents should use beat certain obligations and refrain from specific impermissible behaviours [17]. The use of dominance act utilitarianism (DAU) is used to encode and reason about the obligations of autonomous systems [17]. The paper uses DAU to analyze Intel's RSS rules – a set of rules that would result in zero traffic collisions if all vehicles followed – to derive undesirable consequences of these rules [17]. DAU can illustrate how to help design systems that have specific obligations. We can show how to model check DAU obligations with the system presented by the authors [17]. The system implemented RSS rules one, two, three, and six, which stipulate that vehicles cannot hit someone from behind, that they do not cut in recklessly, that the right-of-way is given but not taken, and that to change lanes we should not wait for a perfect gap, respectively [17]. Ultimately, it is seen that there is a system to derive autonomous vehicle obligations, permissions, and undesirable outcomes on the road without programming specific rules exhaustively. The ability to derive obligations using deontic logical reasoning helps autonomous vehicles make inferences and engage in logical knowledge representation needed to prevent undesirable outcomes when driving. The article by *Shea-Blymyer et al. (2020)* is a departure from more traditional methods of AI for autonomous vehicles as it uses logical reasoning and knowledge representation methods to logically conclude inferences for driving rather than classical ML methods.

C. Machine Learning Validation

As was presented by *Wang et al. (2017)*, proactive decision support systems can predict future states and mitigate unwanted

states by taking specific actions proactively, as was shown with the implementation and use of a proactive decision support method with reinforcement learning and state partition [14]. Proactive decision support systems can predict future states and mitigate or eliminate undesirable states by taking some actions proactively [14]. The paper proposed using proactive decision support methods based on deep reinforcement learning and state partition [14]. Two types of state partition and parallel execution methods are proposed to address huge state space problems. The experimental evaluation concluded that traffic congestion control applications show the method works well in accuracy and performance [14]. The method from the paper was used to predict traffic flow congestion levels of systems. It was concluded that the decision support method with reinforcement learning executed actions to reduce congestion levels [14]. The study shows the value of big data since we can get data from online streaming services for decision support. We also got to see applications for traffic congestion and control using big data, which allows more AI topics to be applied to autonomous vehicles. Future research on topics related to collision control, weather conditions, and emergency controls applied to autonomous vehicles can be opened from this discovery.

D. Machine Learning Ethics

Liabilities, ethics, and culture-aware behaviours for specifying rulebooks for self-driving cars are the focus of research in the paper presented by *Censi et al. (2019)*. New ways to conveniently define the desired behaviour of autonomous vehicles, a rulebook of pre-ordered sets of rules, compositional properties of the rulebooks, and operations allowed on rulebooks to preserve previously introduced constraints are introduced by *Censi et al. (2019)* using a domain-independent framework. Rulebooks are defined according to realizations, rules, rulebooks, and induced pre-orders on realizations [16]. Realizations aim to define desired agent behaviour - that is, we want to objectively prescribe what we can measure, the externality of actions and outcomes in the world [16]. Preference relations are defined on sets of possible outcomes and are called sets of realizations [16]. A realization for a self-driving autonomous vehicle is a real-world trajectory [16]. Rules function as atoms of behavioural specifications, and they map to real numbers so that realizations can be used as input to rules that act as functions [16]. If a rule is applied to two realizations, then the realization with the lower values represents the realization that minimizes the violation to a greater extent than the other realization [16].

The rulebook system helps specify what should be done when rules have to be violated [16]. Rulebooks are pre-ordered sets of rules where the rulebook itself is a tuple consisting of a finite set of rules and a pre-order on the rulebook itself [16]. Any rulebook can be represented as a directed graph with nodes representing rules and edges representing higher and lower orderings between rules [16]. Rule importance and relationships can be derived from the graphical interpretation of rulebook tuples [16]. Finally, we note that induced pre-orders on realizations are used to formally define rulebook semantics by specifying pre-orders on realizations [16]. Not all relevant priorities between rules are specified since *rulebooks* are defined as pre-ordered sets of rules [16]. Rulebooks can be used as partial specifications since the intention to pre-order all realizations can be interpreted as verifying that one rulebook is at least as good – that the degree of violation of one rulebook is at least as good – as another [16].

Examples of the types of rules that are useful in the driving domain include the scenario where a rule is set (with high priority) to not collide with other objects, and another rule is set to not cross into the oncoming lane (with low priority) [16]. A vehicle following the rules would move into an empty oncoming lane to avoid a collision with another vehicle if and only if the oncoming lane does not have another vehicle on it that the current vehicle would inevitably collide with [16]. Such rules allow vehicles to avoid certain collisions without provoking accidents themselves [16]. Applying such a system of rules allows building a set of rules as a formalism which can scale up with complex traffic environments. High-level specification for autonomous vehicles is possible with few constraints on vehicle behaviours. It is essential since rules can be added and prioritized to follow legal, ethical, and philosophical norms for specific international or national standards.

E. Motion Planning

Functionality related to sophisticated decision-making when planning the motion of self-driving autonomous vehicles using a practical rule engine learning from expert driving decisions is the topic of study in the research article presented by *Bouchard et al. (2022)*.

A first layer determines sets of feasible parameterized behaviours, and a second layer reconciles parameters into singular behaviours [22]. Reports are generated on implementation in level-three autonomous vehicles, and field tests are performed in urban environments [22]. Each layer of the rule theory operates using a set of unordered rules that map sets of input properties to sets of parameterized output behaviours [22]. The resulting behaviours can be interpreted as high-level maneuvers and transformed into inputs for the second layer (the parameter layer) [22]. Essentially, a practical rule engine learned from expert driving decisions is proposed using an algorithm for making and manipulating rule-based behaviour planners utilizing a two-layer rule-based theory with resulting behaviours being transformed so that the high-level maneuvers can form input for the parameter layer [22]. The parameter layer then resolves different parameters to output a single high-level maneuver with its parameter [22]. Tuples of finite sets describe layers of the rule-based theory, and the maneuver layer outputs behaviours that pass compatibility tests with outside-world rules [22]. The overall function of the rule engine is summarized with the following diagrammatic representation [22]:

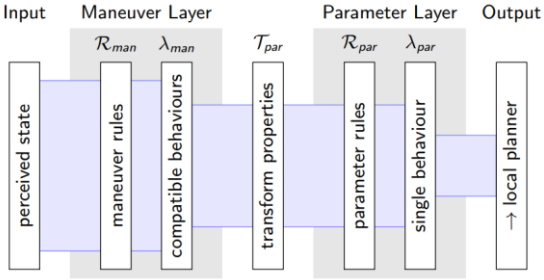


Figure 8: Diagrammatic Representation of Two-Layer Rule Engine [22].

We can break down the image with the following information:

1. Sensors are used to present perceived states of information to the maneuver layers – which in turn identify compatible sets of conservative behaviours with the maneuver rules and similar behaviours [22].
2. The transformation properties are used to transform and complete the resulting properties for the input to the parameter layer [22].
3. The parameter layer resolves a single behaviour by using the parameter rules and single behaviours [22].
4. Finally, the resulting information from the parameter layer is sent to the local planner for interpretation [22].

It is assumed that experts provide finite sets of training scenes with associated levelling functions to assign behaviours to every training scene to learn a theory for the maneuver [22]. A backward-chaining coverage function is used to facilitate learning [22]. A subset of the training scenes that trigger rule r is returned such that they cause the rule to contribute to the resolved result of its training theory associated with a layer with a corresponding resolution function and property transformation function [22]. The use of the property transformation function allows any layer to be trained with training scenes that are defined concerning the input of the rule engine [22]. The rule engine filters a set of candidate behaviours, and random selection is used to avoid giving undue bias to any particular derived solution [22]. The existence of a novel candidate is ensured by training scenes being uniquely defined by their properties and constraints that are added to rules until it specifies a unique scene [22]. Discrepancy identification is achieved by identifying scenes that exemplify discrepancies from either test suits, simulation tests, recordings of traffic flows, or open-road testing [22]. Maneuver diagnosis is achieved since forward, and backward chaining can identify training scenes that trigger rules that misclassify discrepancy scenes [22]. Knowledge extraction must be obtained by removing irrelevant properties that make scene data too specific to be used as scene variables that are irrelevant to concluding correct behaviour are still incorporated [22].

The analysis of conflicting scenes and eliminating irrelevant properties from discrepancy scenes is performed by setting them as undefined [22]. Rule engineering is used by getting algorithm one from the paper to update with input parameters set to the existing theory, and algorithm two necessarily selects discrepancy scenes as the first misclassification to repair until every training scene derives its expected behaviour [22]. Thus rule-based training development can further refine rules and connect rule theories with rule practices used in self-driving autonomous vehicles [22]. Experimental results from a prototype vehicle using many optimizations described by the text and caching results of rule antecedent evaluation show that the vehicle successfully learned driving policies [22]. It is concluded that autonomous vehicles can drive in urban environments with only 267 maneuver rules [22]. Field tests reveal the viability of rule-based theories within ODDs. The rule engine could serve queries at up to 300 Hz [22]. During public test drives, safety drivers needed to intervene 58 times, comparable to the performance of end-to-end deep learning approaches [22]. Overall, autonomous vehicles were shown to be able to handle tasks which require interventions, such as temporarily encroaching into the oncoming lanes but had difficulties due to possible limitations external to the rule engine [22]. The demonstrated project reveals the feasibility of the given approach as it demonstrated level-three autonomous vehicle behaviour [22]. Future research on the given approach could be used to refine rules based on statistical preferences or to produce more complex rulesets for more complex vehicle interactions.

F. Data Validation and Testing

The data validation problem for machine learning systems is the topic of study for *Breck et al. (2019)*, and it is illustrated that the presented data validation system helps detect anomalous data inputted into machine learning pipelines. The topic of monitoring and evaluating the data fed into machine learning systems to prevent anomalous data from impacting ML pipelines is aided by using data validation systems [6]. Data validation for the following types is included in the system:

1. Single-batch validation is used to detect anomalies in single batches of data [6].
2. Inter-batch validation is used to detect significant changes between training and serving data – or between successive batches of the training data [6].
3. Model testing is used to determine if any assumptions in the training code are not reflected in the data [6].

Single-batch validation expects data characteristics to remain stable within each batch. It works by generalizing the traditional notion of a schema from database systems to validate batches of data by comparing it against the schema [6]. The inter-batch validation operates by finding drafts in the distribution of feature values across multiple batches of data [6]. Drifts can occur due to issues associated with the data, such as training-serving skew, feature skew, distribution skews, and more [6]. Essentially, quantifying distribution distance uses metrics to detect distribution skews between the training and serving distributions [6]. Model unit testing is used to detect mismatches between expected and actual data states and to find software errors in generating training or serving data [6]. Mismatches between expected data and the assumptions made in training code are detected with a fuzz-testing strategy that triggers hidden assumptions in the code that do not agree with schema constraints [6]. The system analyzed petabytes of data per hour by catching significant anomalies early and helping diagnose model-quality problems caused by data errors [6].

Empirical evaluations produced by *Breck et al. (2019)* are used to provide evidence of the effectiveness of their implemented method. Case studies have elaborated on the use of the system as it found missing features in the Google Play recommender pipeline, used data debugging to lead to model wins for generating video recommendations, stand-alone data validation for product teams, and feature-store migration for validating migration of features to new stores [6]. The system shows promise for diagnosing, debugging, and detecting flaws associated with data for ML pipelines and can be used in conjunction with big data systems for autonomous vehicles.

Autonomous vehicles would have many data from their sensors and big data systems connected to the internet. A validation system would help find anomalous data associated with invalid sensory output, adversarial attacks, and formatting errors to prevent ML models for autonomous vehicles from failing.

The topic of testing machine learning models and challenges associated with testing, existing solutions, and gaps in current research are discussed in a paper by *Braiek & Khomh (2020)*. Breakthroughs in deep learning and reinforcement learning have allowed such systems to be used in safety-critical systems [7]. To ensure ML systems' reliability, software testing concepts – like code coverage, mutation testing, and property-based testing – have been used to help ML engineers detect and correct faults in ML programs [7]. It is noted that ML models' errors can be due partly to conceptual mistakes when creating model implementations and writing code [7]. Detection of conceptual and implementation-based errors in ML models can be achieved with approaches that aim to detect conceptual errors in ML models by assuming the models are implemented into programs without errors and focusing on providing mechanisms to detect potential errors in the calibration of the models [7]. Such systems include black-box and white-box testing approaches for ML models [7].

Approaches aiming to detect errors in ML code implementations mostly rely on numerical testing, property-based testing, metamorphic testing, mutation testing, coverage-guided fuzzing, and proof-based testing techniques to detect issues in ML code implementations [7]. Finite-difference techniques that leverage the optimization-based nature of most ML problems and property-based testing using the inferences of properties of computation using theory and formatting invariants that should be satisfied by the code are elaborated on by *Braiek & Khomh (2020)*. Metamorphic testing tests metamorphic relationships by testing numerical attributes and the associated transformations on the tested models [7]. Mutation testing investigates test data quality by injecting faults into the source of ML programs using source-level mutation operators [7]. Finally, proof-based testing can be used to specify computations of ML programs and construct formal proofs of written theorems to define what programs mean to be correct and error-free [7]. Machines can also verify the model's validity without requiring human intervention in the proof-based testing approach [7].

As can be seen, there are several ways to test ML models for errors or biases in different parts of the model, the code, the assumptions used to build the model, and more. Developers could look into more details for specific methods. However, the domain of self-driving autonomous vehicles would greatly benefit since models become complex and challenging to test in global test cases. Setting standardized benchmarks for autonomous vehicle models is also essential since they cannot be considered safe without establishing benchmarks and standards associated with potential errors. The above systems provide valuable tools for testing machine learning models.

CONCLUSION

The paper presents a literature review of the methods used by previous authors for self-driving autonomous vehicles. Topics related to development models, requirements engineering, software engineering models, autonomous vehicle cooperation and irrationality models, machine learning validation, machine learning ethics, motion planning systems, and data validation were discussed.

The report does not cover test cases that integrate the various models and systems discussed. The interactions between the systems would produce highly complex emergent behaviours and would warrant future studies with empirical or theoretical analysis to verify system a potential system's feasibility. Other methods unrelated to modern machine learning and artificial intelligence, such as classical AI and algorithm models, could be included in the discussion. Many modern works do not introduce highly related topics that have already been solved by the classical algorithm and AI techniques and instead focus on ML methods. The report did not compare related methods with direct empirical analysis to determine quantitative comparisons between methods. The report also did not compare methods through algorithmic analysis to determine optimality or worst-case performance. Methods relying *specifically* on distributed computing systems or database systems were not included and were outside the scope of the paper but could be discussed as distributed computing systems and databases could coordinate large-scale autonomous vehicles in a city.

Future research directions can be applied using the information found within the paper. Information presented by the authors of the reviewed papers can be researched in more detail for specific information for future projects and directions of research. The ideas in the paper should help readers develop knowledge related to the current state-of-the-art with AI and big data related to self-driving autonomous vehicles to place the foundational building blocks for future research and learning endeavours.

ACKNOWLEDGMENT

The presented literature review was completed on behalf of Paulo Alencar as part of a course project requirement for CS846 – Software Engineering for Big Data and Artificial Intelligence, as was offered during the fall term of 2022 at the University of Waterloo.

REFERENCES

[1] A. Horneman, A. Mellinger, and I. Ozkaya, "AI Engineering: 11 foundational practices for decision makers," *SEI Blog*, 09-Dec-2019. [Online]. Available: http://insights.sei.cmu.edu/blog/ai-engineering-11-foundational-practices-for-decision-makers/. [Accessed: 07-Nov-2022].

[2] A. Mohammadnazar, R. Arvin, and A. J. Khattak, "Classifying travelers' driving style using basic safety messages generated by connected vehicles: Application of unsupervised machine learning," *Transportation Research Part C: Emerging Technologies*, vol. 122, 2021.

[3] A. Vogelsang and M. Borg, "Requirements Engineering for Machine Learning: Perspectives from data scientists," *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pp. 245–251, 2019.

[4] CACM Staff, "Big data," *Communications of the ACM*, vol. 60, no. 6, pp. 24–25, May 2017.

[5] D. Monroe, "Deceiving ai," *Communications of the ACM*, vol. 64, no. 6, pp. 15–16, Jun. 2021.

[6] E. Breck, M. Zinkevich, N. Polyzotis, S. Whang, and S. Roy, "Data Validation for Machine Learning," *Google Research*, 2019. [Online]. Available: https://research.google/pubs/pub47967/. [Accessed: 07-Nov-2022].

[7] H. B. Braiek and F. Khomh, "On testing machine learning programs," *Journal of Systems and Software*, vol. 164, p. 110542, 2020.

[8] H. Jaeger, “Deep neural reasoning,” *Nature*, vol. 538, no. 7626, pp. 467–468, Oct. 2016.

[9] H. Villamizar, T. Escovedo, and M. Kalinowski, “Requirements Engineering for Machine Learning: A Systematic Mapping Study,” *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 29–36, 2021.

[10] O. Etzioni, “Point: Should AI technology be regulated?,” *Communications of the ACM*, vol. 61, no. 12, pp. 30–32, Nov. 2018.

[11] R. Akkiraju, V. Sinha, A. Xu, J. Mahmud, P. Gundecha, Z. Liu, X. Liu, and J. Schumacher, “Characterizing machine learning processes: A maturity framework,” *Springer International Publishing*, vol. 12168, pp. 17–31, Sep. 2020.

[12] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, “Software engineering for machine learning: A case study,” *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 291–300, May 2019.

[13] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner, “Software engineering for AI-based systems: A survey,” *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 2, pp. 1–59, Apr. 2022.

[14] Y. Wang, S. Geng, and H. Gao, “A proactive decision support method based on Deep Reinforcement Learning and state partition,” *Knowledge-Based Systems*, vol. 143, pp. 248–258, 2018.

[15] Z. Peng, J. Yang, T.-H. P. Chen, and L. Ma, “A first look at the integration of machine learning models in complex autonomous driving systems: A case study on Apollo,” *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1240–1250, 2020.

[16] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, “Liability, ethics, and culture-aware behavior specification using rulebooks,” *2019 International Conference on Robotics and Automation (ICRA)*, Mar. 2019.

[17] C. Shea-Blymyer and H. Abbas, “A deontic logic analysis of Autonomous Systems' safety,” *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020.

[18] L. Chan, A. Critch, and A. Dragan, “Human irrationality: Both bad and good for reward inference,” *arXiv.org*, 12-Nov-2021. [Online]. Available: https://arxiv.org/abs/2111.06956. [Accessed: 08-Nov-2022].

[19] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” *arXiv.org*, 15-Mar-2018. [Online]. Available: https://arxiv.org/abs/1708.06374v5. [Accessed: 08-Nov-2022].

[20] F. Bouchard, S. Sedwards, and K. Czarnecki, “A Rule-Based Behaviour Planner for Autonomous Driving,” *Springer*, 2022.

[21] C. B. Molina, J. R. Almeida, L. F. Vismari, R. I. Gonzalez, J. K. Naufal, and J. B. Camargo, “Assuring fully autonomous vehicles safety by design: The Autonomous Vehicle Control (AVC) module strategy,” *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2017.

[22] S. Lan, C. Huang, Z. Wang, H. Liang, W. Su, and Q. Zhu, “Design automation for intelligent automotive systems,” *2018 IEEE International Test Conference (ITC)*, 2018.

[23] R. Salay, R. Queiroz, and K. Czarnecki, “An analysis of ISO 26262: Machine Learning and safety in automotive software,” *SAE Technical Paper Series*, 2018.

[24] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why should I trust you?: Explaining the predictions of any classifier,” in *ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[25] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, “Planning for autonomous cars that leverage effects on human actions,” *Robotics: Science and Systems XII*, 2016.

[26] M. Zhao, Z. Liu, S. Luan, S. Zhang, D. Precup, and Y. Bengio, “A consciousness-inspired planning agent for model-based reinforcement learning,” *arXiv.org*, 04-Nov-2021. [Online]. Available: https://arxiv.org/abs/2106.02097. [Accessed: 14-Dec-2022].

[27] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, “Social Coordination and altruism in autonomous driving,” *arXiv.org*, 04-Apr-2022. [Online]. Available: https://arxiv.org/abs/2107.00200. [Accessed: 14-Dec-2022].

[28] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, “Cooperative Autonomous Vehicles that sympathize with human drivers,” *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.