Automotive Application of Foundation Models

Arunachalam Thirunavukkarasu[1], Domenik Helms[1], Dominik Grundt[1], Christian Ojeda Bernal[2], Leonard Schroven[3], Sami Wilk[3], Sven Mantowsky[4], Syed Saqib Bukhari[4], and Róbert Lajos Bücs[5]

[1]Deutsches Zentrum für Luft-und Raumfahrt e.V. (DLR)
[2]Valeo Schalter und Sensoren GmbH
[3]Capgemini
[4]ZF Friedrichshafen AG
[5]Aptiv Services Deutschland GmbH

21 November 2024

Abstract

The rapid advancement of generative AI is catalyzing a transformative shift within the automotive industry, moving it from traditional design and development frameworks toward modern, AI-driven architectures. This paper provides a comprehensive overview of the integration, scalability challenges, and evaluation of generative AI in automotive applications, highlighting both the technological advancements and the practical limitations posed by constrained hardware environments typical of vehicle systems. To adapt large deep learning models for these edge devices, we explore three optimization techniques—pruning, quantization, and low-rank approximation—that ensure efficient AI performance without significant compromises. Additionally, we address the evolving electrical/electronic (E/E) architectures that are enabling such integration and examine early instances of generative AI deployment within the sector.

A crucial aspect of integrating generative AI into real-world automotive applications is the plausibility of AI outputs, which must mimic human-like data while avoiding pitfalls such as hallucinations, data leakage, and unintended replication. We discuss methods for evaluating plausibility, including fidelity, diversity, and novelty, and the challenges posed by domain-specific biases and subjective assessments. Ensuring AI reliability, trustworthiness, and privacy safeguards is central to this work, with a focus on making generative AI both a scalable and reliable solution for the automotive industry's ongoing digital transformation.

1 Introduction

1.1 Optimization

The advent of generative artificial intelligence (AI) is ushering in a new era of innovation in the automotive industry. From revolutionary design processes to the development of autonomous driving systems, generative AI offers unparalleled opportunities to enhance vehicle performance, safety, and customer experience. As automotive manufacturers increasingly turn to AI to maintain a competitive advantage, the need to scale these technologies across production lines and vehicle fleets becomes ever more pressing.

Generative AI is transforming the development process across various industry sectors, with the automotive industry being no exception. This technology significantly impacts the entire vehicle development pipeline of original equipment manufacturers (OEMs), playing a crucial role from the early design stages to in-built functions that assist drivers throughout the vehicle's lifespan.

Despite the great benefits and potential

of generative AI in the automotive industry, its application is hindered by the nature of the automotive domain, specifically in scaling its use within constrained hardware environments. The transformation brought by the scalability of generative AI in the automotive industry can be categorized into two main areas: in-vehicle functions and external applications.

In-vehicle functions are often constrained by the limitations inherent in vehicle architecture. Whether in high-end performance vehicles or economical cars, these systems operate under tight constraints regarding memory, energy, and computational power. All these areas are critical for the effective application of generative AI functions, as they demand robust hardware resources. This white paper aims to explore the current challenges of scaling generative AI in the automotive industry, with a particular emphasis on in-vehicle functions.

Generative AI is also making significant strides in external applications within the automotive industry. These applications range from design and manufacturing stages to customer interactions and predictive maintenance. By leveraging generative AI, automotive companies can create more efficient and innovative designs, optimize manufacturing processes, and provide personalized experiences to customers.

This white paper focuses on the scalability of generative AI in the constrained hardware environments typically found within vehicles. Issues such as limited memory, energy, computational power, and network communication capabilities present significant obstacles. Our analysis includes insights into current applications, technical challenges, and emerging strategies to overcome these barriers.

The structure of this paper is as follows: In Section 2, We present case studies documenting the initial applications of generative AI in the automotive pipeline. These examples illustrate the practical benefits and challenges encountered, providing valuable insights for future implementations. In Section 3, we discuss the primary challenges associated with scaling generative AI in the automotive context, focusing on hardware limitations, safety considerations, and integration complexity. Section 4, explores the architectural adaptations needed for scalable AI integration, covering changes in both hardware and software designs to support AI-driven systems. In Section 5, we present strategies for deploying generative AI in resource-constrained environments, with a special focus on model optimization techniques. By examining current approaches to adapt large models to limited hardware, we aim to provide a comprehensive understanding of how the automotive industry can effectively leverage generative AI in in-vehicle functions. In Section 6, we look ahead to future directions and trends in generative AI for automotive applications, identifying emerging research areas and technological advancements. Finally, Section 7 concludes the paper by summarizing our findings and outlining recommendations for the industry.

1.2 Plausiblity

In order to evaluate the plausibility of the output from a generative AI model, it is essential to first define what it plausibility means. Broadly, plausibility refers to the overall quality of the generated dataset, which can be broken down into several criteria that synthetic data must meet to be considered high quality.

At its core, plausibility refers to the extent to which the synthesized content accurately reflects the real world. In other words, it can be understood as the absence of inaccuracies, biases, or hallucinated inference outputs of the generative AI model. Ensuring plausibility means that the data produced is not only realistic but also free from errors that would make it diverge from real-world phenomena.

In this context, methods for determining the plausibility for generative AI models are examined in the specific use cases of generated sensor data, scenarios and planning as well as maps and traffic guidance. These areas are particularly important as they require the synthetic data to be consistent with real-world dynamics and interactions, ensuring that the generative model's outputs are not only plausible but also practically applicable in real-world scenarios.

Fidelity

Fidelity, also referred to as precision in the literature, measures how realistic the generated data is. A dataset has high fidelity if it is difficult to distinguish from genuine data, while it has low fidelity if it is easily identified as being fake. Measures for the fidelity of generative models include α-precision [4], density [105] and the realism score [62].

Diversity

Diversity, also known as recall or coverage, measures how well a model is able to generate the full spectrum of data which can be observed in the real world. A model has high diversity if the generated data represents the entire variety of possibilities. It has low diversity if the model fails to generate data of a certain structure and only spans a subset of the real-world observations. Approaches for measuring diversity include β-recall [4] and coverage [105]. Additionally, there are several proposed metrics that aim to measure diversity intrinsically, without reference to a true dataset or distribution. For instance, the Vendi score [53] and Rényi kernel entropy [72].

Novelty

Commonly used terms for this concept include rarity, authenticity, and generalization. This dimension captures a model's ability to create truly new data. A model exhibits high novelty if the generated data is distinctly different from the training data. Metrics used to assess novelty include the rarity score [62] and authenticity [4]. Novelty also relates to privacy. When training data contains intellectual property, sensitive health data, or similar information, insufficient novelty in the generated data poses a privacy risk, as the training data could potentially be reproduced.

1.2.1 Classification of methods

Quality metrics for generative learning can be classified into different groups. These groups are not mutually exclusive, meaning the same metric can belong to multiple categories.

Qualitative vs. Quantitative

Especially in A.I. image generation, it is common practice to assess the quality of a model based on evaluation of a subset of the generated data by a human. This approach is highly subjective and dependent on the person evaluating the pictures while also being highly random due to the strong influence of the selected samples. Qualitative analysis might be a viable step to get a first impression of the data, but it should not be used as the sole method to gauge the performance of a model. Besides, when evaluating sensor data, it is unclear to what extent qualitative evaluation is a practicable approach, given that humans have a very good understanding of which images look real and which do not but have a very poor intuition about how realistic sensor data would look. This, in contrast, would require expert domain knowledge.

Sample-Level vs. Distribution-Level

While some evaluation metrics can be assessed on individual samples, others are properties of a probability distribution or the dataset as a whole. While sample-level metrics can be used to construct quality metrics at the dataset level, it's unclear how to go in the other direction. Sample-level metrics include α-precision and β-recall while distribution-level metrics include FID, KID, and IS.

Reference-based vs. Intrinsic

Some metrics require a reference dataset or distribution of labels to be computable. This includes, for example, FID. Other metrics try to measure more intrinsic attributes of the model, such as the intrinsic notions of diversity mentioned before, like the Vendi score [53] and Rényi kernel entropy [72].

Learned vs. Learning-free

It is common practice to transform data into a feature space before computing simpler evaluation metrics. This feature map typically comes in the form of the last layer of popular trained neural networks like VGG16 or DINOv2-ViT-L/14. Examples include the Inception Score and α-precision. The hope is

that the distance of the transformed data in the feature space has more meaning than the relative position of the original data in the observation space. Other metrics forgo any learnt components.

Domain-Dependent vs. -Independent

Domain dependence is an issue that needs attention, especially regarding learned features. A learned feature embedding is always trained on data from a specific domain. In fact, applying the inception score to models not trained on ImageNet can yield misleading results [17]. This is particularly relevant for niche domains like sensor data, where fewer publicly available models exist. Therefore, it is desirable to rely on evaluation metrics that are agnostic of the working domain.

Model-Agnostic vs. Model-Dependent

Existing approaches in the literature of quality assessment for generative models, as well as deep fake detection and image forensics, can be divided into techniques that are independent of the choice of generative models as well as methods that are tailored to certain architectures. This is due to the fact that different generative algorithms tend to leave different artifacts in the generated data. In GANs, this can, for example, be inconsistency of the corneal specular highlights between the two eyes [75]. Also, there is evidence that evaluation metrics for model quality seem to be biased towards certain architectures [139].

2 Current use of Generative AI Applications in Automotive Industry

Generative AI is driving transformative changes across various facets of the automotive industry, from synthetic data generation to enhancing in-driving processes through embedded functionalities. Some of these applications require real-time data processing and generation to effectively manage diverse real-world scenarios. The current use of generative AI in automotive applications can be categorized into several key areas:

Synthetic Data Generation

Generative AI plays a crucial role in generating synthetic datasets where specific conditions are scarce or challenging to reproduce. For instance, the WEDGE dataset [101], which consists of 3360 images in 16 extreme weather conditions, was constructed using generative vision-language models, provides diverse weather scenarios essential for training autonomous driving systems. This capability is vital for testing and improving the resilience of autonomous vehicles in varied and extreme weather conditions.

Super-Resolution of Sensor Data

Another significant application lies in the super-resolution of sensor data. Techniques explored in [69] leverage generative AI to enhance the resolution of images captured by vehicle sensors. This enhancement improves the accuracy and reliability of object detection and recognition systems.

World Model Simulation for Autonomous Driving

Generative AI is pivotal in simulating world models for autonomous driving. For example, projects like GAIA-1 [66] use generative AI to create realistic driving environment simulations. These simulations are instrumental in training and validating autonomous driving algorithms, thereby facilitating safer and more efficient development.

Vehicle Design Assistance

Generative AI is also revolutionizing the vehicle design process. The Toyota Research Institute has developed a generative AI tool capable of accelerating and enhancing vehicle design phases. This tool proposes innovative designs and optimizes existing ones, significantly reducing time and costs associated with vehicle development [8]. Similarly, Audi has employed generative models to design rims, generating photo realistic designs or recombining existing ones to enhance both aesthetic appeal and functional performance [10].

3 Challenges of scaling and verifying Generative AI in the Automotive Industry

3.1 Challenges in scaling (optimization):

Scaling generative AI in the automotive industry presents a complex array of challenges, particularly when considering the constraints of in-vehicle hardware. However, the scalability of generative AI extends beyond hardware limitations. It inherits several challenges from the broader field of generative AI while also encountering industry-specific obstacles. These challenges revolve around integration with existing systems, data management, and computational resource demands. Additionally, the automotive industry faces unique issues such as stringent safety regulations, the need for seamless interaction between AI-driver and traditional engineering approaches. Addressing these areas is essential for the successful large-scale implementation of generative AI in the automotive sector. This section focuses primarily on the inference stage, which poses significant challenges.

3.1.1 Computational Resource Demands and Inference Costs

Deploying generative AI models can be prohibitively expensive due to the need for significant computational resources [155]. Current applications often rely on traditional cloud computing, which introduces latency and security concerns due to data transmission over networks [164]. According to McKinsey [19], the demand for compute hardware used for AI inference is twice as large as that used for training in data centers meanwhile up to four times in the edge market, and is expected to keep growing. Additionally, in [52] indicates that NVIDIA estimates inference costs can account for up to 80-90% of the total cost of a neural network.

3.1.2 Data Management, Privacy and Security Concerns

Scaling generative AI systems necessitates the collection and processing of vast amounts of data, raising concerns about data privacy, security, and intellectual property [55]. Ensuring robust data protection measures is essential to prevent unauthorized access and breaches. Privacy concerns are particularly acute when dealing with user-generated data, such as speech recognition inputs or perception data from sensors [76].

3.1.3 User Acceptance and Trust

Gaining widespread user acceptance and trust in AI-driven functionalities is essential for scaling [138][47]. Ensuring transparency, reliability, and user-friendly interfaces is critical to building consumer confidence and facilitating broader adoption. Additionally, the interpretability of generative AI models is crucial for understanding their decision-making processes and ensuring they operate safely and predictably [165].

3.1.4 Latency and Real-Time Performance

Inference latency is a critical issue in the automotive industry, where real-time performance is often required [155]. Depending on the architecture of generative models, the time needed from input reception to output generation can vary significantly. In automotive applications, delays in inference can compromise safety and user experience. Therefore, achieving low-latency performance is crucial for real-time applications such as advanced driver-assistance systems (ADAS) and autonomous driving [165]. Safety-critical applications place strict requirements on the throughput and latency of deep learning models. Generative AI systems must analyze large streams of data in real time with low latency to ensure safe vehicle operation. Any delay or error in processing can lead to hazardous situations.

The primary challenge related to the deployment of generative AI in vehicles is the limited computational power of in-vehicle hardware. Unlike data centers equipped with high-performance GPUs and TPUs, in-vehicle systems operate under stringent resource constraints. The diversity of hardware platforms within vehicles further complicates the deploy-

ment of AI models, as these models must be compatible with a wide range of processors and architectures.

Scaling generative AI solutions in the automotive sector involves integrating them with existing vehicle systems and infrastructure [76]. This requires overcoming compatibility issues and ensuring seamless operation alongside legacy technologies. Achieving this integration while maintaining system performance and reliability is a significant challenge.

3.1.5 Safety Validation

Validating the safety and effectiveness of generative AI at scale requires extensive testing [76], potentially involving billions of miles of driving data. The complexity of AI systems makes it difficult to exhaustively test and verify performance across all possible scenarios. This poses a significant challenge for large-scale deployment in the automotive industry.

3.1.6 Energy Efficiency

With the rise of electric vehicles and the constant growth of in-vehicle sensors [1], energy efficiency has become increasingly important. Generative AI models require substantial computational power, which in turn consumes significant energy. Efficient energy use is crucial for maintaining vehicle performance and extending the operational lifespan of electric vehicles.

Scaling generative AI in the automotive industry requires addressing a diverse set of challenges. These include overcoming the limitations of in-vehicle hardware, ensuring data privacy and security, gaining user acceptance, and achieving real-time performance with low latency. The high computational demands of AI inference, the need for extensive validation and testing, and the imperative for energy further complicate the landscape. Successful large-scale implementation of generative AI in this sector necessitates robust integration with existing vehicle systems and infrastructure, while maintaining performance, reliability, and safety. By tackling these multifaceted challenges, the automotive industry can unlock the transformative potential of generative AI, driving innovation and enhancing both vehicle functionality and user experience.

3.2 Challenges in verification:

3.2.1 Verification of Generative AI models:

The problems of evaluating the quality and diversity of foundation model outputs are closely related to the difficulty of evaluating human performance for specific tasks. To evaluate human performance a lot of standardized tests or benchmarks have been developed like the SAT for college admissions in the United States, the bar exam, German Abitur exams and many more. Therefore it is quite intriguing to evaluate a large language model (LLM) on the same benchmarks, since it should mimic human behaviour. In 2024 Katz et al. published an article that GPT-4 passed the bar exam [77] claiming that GPT-4 scored in the 90th-percentile, which gained a lot public attention. However, these claims are under critical investigation [102]. The standard procedure of evaluation generative AI modlels consists of three steps [3]. First the model responses for each prompt in the benchmark are collected. The a performance metric for each response is calculated. Finally, a single performance metric is aggregated from the performance over the benchmark, usually by averaging over all individual performances. One of the main problems when evaluating LLMs on the common benchmarks is potential data leakage or data contamination. As Zhou et al. pointed out, benchmarking performance of an LLM can be significantly boosted if the training data is contaminated with benchmarking data [177]. But as most commercial LLMs do not disclose their training data it is very difficult to directly prove data leakage. Oren et al. proposed a method to prove test contamination of black box LLMs by randomly reordering test set tasks and observe the performance differences on the canonically ordered test set and randomly shuffled test sets [112]. A study from 2024 claimed that at least 4.7M samples from 263 benchmarks have been exposed to LLMs duing training [15]. According to [119] there are three approaches for evaluating LLMs

- The **Trusted Model Owner** method, where the model is queried via an API with the benchmark and the model owner must be trusted to not use these queries for training.
- The **Trusted Dataset Owner** method, where the model is open-source but the benchmark dataset is kept private and the dataset owner must be trusted to host the model and perform the benchmarks correctly.
- The **Trusted Third Party** method, where a trusted third party hosts the computation environment for the model as well as the dataset and does the evaluation and shares the results.

Another problem with the current generation of LLMs is *hallucination*. A common definition for a hallucination is an output from a prompt of an LLM that is either in conflict with existing sources or cannot be verified by available knowledge sources [73, 91]. An evaluation done by Li et al. claims that, depending on the topic, ChatGPT provided unverifiable information in about 19.5% of the responses. One approach to counter the hallucination of LLMs is to provide them with additional ressources that were not available during training. This results in hybrid generation models with parametric memory from the LLM and non-parametric memory from external sources. This approach is called **R**etrieval **A**ugmented **G**eneration (RAG) [89].

3.2.2 Verification of Generative AI data models:

Verifying the data used to train generative AI (GenAI) models presents several critical challenges that directly impact the reliability, fairness, and ethical use of these models. This chapter explores these challenges in detail. Data Bias and Fairness addresses the difficulty of identifying and mitigating biases in training data, which can lead to unfair and socially inequitable outcomes if not properly managed. Data Quality and Accuracy highlights the challenge of ensuring that training data is consistent, accurate, and complete, as poor data quality can severely degrade model performance. Data Representativeness focuses on the challenge of ensuring that training datasets accurately reflect the diversity of the target population, which is essential for the model's applicability in real-world scenarios. Finally, Synthetic Data and Its Verification examines the growing use of synthetic data, emphasizing the challenges of verifying its representativeness and ethical use, particularly in avoiding false confidence and consent circumvention.

Data Bias and Fairness:
A significant challenge in generative AI is ensuring that the data used for training does not perpetuate or amplify societal biases. Ntoutsi et al. (2020) highlight that bias in AI systems, particularly those that are data-driven, can lead to unfair and potentially harmful outcomes. The challenge lies in identifying these biases within large and complex datasets, understanding their socio-technical origins, and developing strategies to mitigate them. Addressing bias requires a multifaceted approach that includes technical interventions, such as bias-aware algorithms, as well as broader ethical and legal considerations to ensure fairness and protect against human rights violations [109].

Data Quality and Accuracy:
Ensuring high data quality is a core challenge in training reliable generative AI models. Budach et al. (2020) emphasize that poor data quality—characterized by inconsistencies, inaccuracies, and incompleteness—can significantly degrade the performance of machine learning models. The challenge is particularly acute when dealing with large-scale datasets, where errors in training data can lead to flawed model predictions. The authors advocate for a rigorous, data-centric approach to AI development, emphasizing the need for continuous monitoring and improvement of data quality across multiple dimensions to ensure the trustworthiness and effectiveness of AI systems [27].

Data Representativness:
A critical challenge in developing generative AI models is ensuring that the training data is representative of the target population. Li et al. (2017) explore the difficulties of working with imbalanced datasets, where certain

classes are underrepresented, leading to biased and unreliable model performance. The challenge lies in both identifying these imbalances and implementing effective strategies, such as undersampling and synthetic data generation, to enhance the representativeness of the dataset. The authors demonstrate that without addressing these imbalances, models are prone to inaccuracies and can perform poorly in real-world scenarios [90].

Syntethic Data and its Verification: The increasing reliance on synthetic data presents new challenges in ensuring the validity and ethical use of such data in AI training. Yanisky-Ravid and Liu (2020) discuss the risks associated with synthetic data, including the challenge of verifying its representativeness and the potential for "diversity-washing," where synthetic data falsely appears to increase dataset diversity. Another significant challenge is the circumvention of consent requirements, which raises ethical and legal concerns, especially in regulated environments. These issues complicate the governance of AI systems, making it difficult to ensure that synthetic data is used responsibly and that models trained on such data are trustworthy and fair [160].

4 Adapting Automotive Architectures for Scalable AI Integration

The integration of generative AI into vehicle architectures represents an emerging frontier in automotive technology. Although most research to date primarily addresses architectures optimized for autonomous driving, the foundational concepts and design considerations from these studies remain highly relevant for incorporating generative AI capabilities. Key principles such as architecture centralization, multi-core accelerators, system updates, security, virtualization, data communication and software architecture are not only essential for enabling autonomous driving but are equally critical for the implementation of generative AI within vehicles. By adapting these architectural frameworks, the automotive industry can harness generative AI to enhance diverse vehicle functionalities, and pave the way for more intelligent and personalized vehicles.

4.1 From Distributed to Centralized Architectures

Current vehicle E/E (electrical/electronic) architectures comprise over 100 Electronic Control Units (ECUs) [7, 11, 130, 132]. This architecture is typically referred to as decentralized, as software functions run independently on their respective ECUs. While decentralized architectures offer benefits such as modularity and redundancy, they also present significant challenges, particularly in terms of scalability. The traditional one-to-one mapping of functions to ECUs limits flexibility and makes it difficult to meet the evolving demands of modern vehicles [16]. Additionally, decentralized systems struggle to accommodate the increasing requirements of future vehicles [9]. The industry is now witnessing a shift driven by the integration of new software features, including AI technologies. This trend is propelling the adoption of more powerful computing platforms and transforming distributed E/E architectures into centralized systems. In particular, domain-based E/E architectures are emerging, where robust domain controllers replace several individual ECUs, enabling more complex functionalities and supporting zonal architectures through the use of central compute units. According to McKinsey, it is projected that by 2032, 30% of all vehicles produced globally will rely on zonal controllers and 40% on domain-based architectures, both requiring centralized, high-performance computing units. In contrast, only 29% of vehicles will continue utilizing traditional distributed ECUs [29]. One of the primary drivers of this architectural evolution is the increasing integration of AI applications, such as generative AI and autonomous driving systems, which demand higher computational power and system centralization [130]. Hardware limitations are a key constraint for deploying advanced AI models in vehicles, particularly given the vast computational resources required by generative models with billions of parameters. In recent years, there has been a surge in the release of AI accelerators designed

specifically to enhance AI capabilities within automotive environments. A comprehensive review of AI accelerators launched in the past few years by various manufacturers [122, 123, 124, 125, 126] categorizes these technologies based on peak power (watts) and peak performance (GOps/sec), revealing five distinct clusters of devices. These categories range from low-power chips for sensor processing to full-scale data center systems. Notably, AI accelerators tailored for autonomous vehicles typically occupy a mid-range position in terms of power and performance, with most using Int8 precision for inference tasks. However, certain accelerators utilize FP16 for training, especially where higher computational precision is required, with peak power consumption ranging from 10 to 100 watts. Proposed architectures, such as those detailed in [84] and [111], illustrate different approaches to integrating multicore System-on-Chips (SoCs) into vehicle systems. The first approach adopts a domain-based E/E architecture, utilizing specialized domain controllers equipped with AI accelerators that communicate with sub-ECUs to manage domain-specific functions. The second approach integrates dedicated cores within a central ECU for each function supplied by different vendors, forming what is termed an integrated architecture.

4.2 System Updates

The average lifespan of a vehicle is typically over a decade [16]. During this period, technological advancements, particularly in AI, progressed rapidly. It is reasonable to anticipate similar advancements in the application of AI within the automotive industry. As such, enabling continuous software updates is essential for ensuring that vehicles remain compatible with the evolving landscape of generative AI applications. The update process should be seamless, requiring minimal intervention and downtime. However, current E/E architectures lack the capability to perform dynamic, online updates while the vehicle's software is in active use [11]. To address this limitation, [11] proposes a container-based software update mechanism. The author suggests that virtualization using containers offers the scalability and isolation required for deploying AI features in modern vehicle architectures. This approach allows for dynamic updates and efficient resource management, paving the way for the continued integration of advanced AI functionalities in future automotive systems.

4.3 Virtualization

Hardware virtualization technology is essential for integrating both safety-critical and non-safety-critical software domains within a single computing unit while optimizing hardware resource utilization [122]. One emerging technology particularly relevant for the next generation of automotive architectures, including those leveraging generative AI, is containerization. Container technology has seen widespread adoption due to its ability to simplify software development workflows, especially in resource-constrained environments like automotive systems [32]. As the industry shifts toward centralized E/E architectures, it is crucial to consider not only changes in hardware and software structures but also the significant impact on the development processes of future automotive systems. Unlike decentralized architectures, where feature isolation is more straightforward due to the physical separation of functions across multiple ECUs, centralized systems face greater challenges in integrating and isolating new features [16]. In [32], the authors demonstrate that adopting modern software development practices, such as containers that encapsulate reusable services, offers numerous advantages. These include flexible service partitioning by allowing containers to be relocated between ECUs, and significantly improving the speed of delivering software updates to operational vehicles.

4.4 Security

As vehicles transition from hardware-centric designs to software-defined architectures, software becomes the primary driver for new features [130]. With the increasing complexity and connectivity of advanced vehicles, they become more vulnerable to security threats [137]. Current reference architectures are inadequate for addressing the security needs of modern systems that incorporate AI-driven functionalities (e.g., autonomous driving) due

to their lack of precision and detail in defining security requirements [137]. Additionally, there is a recognized deficiency in the integration of safety perspectives within automotive software development, necessitating new architectural approaches that can meet both safety and security demands [7]. Traditional system architectures have primarily focused on fail-safe designs. However, modern architectures must be fail-operational [16]. This means that systems should not only ensure safety in case of a failure but also maintain high levels of functionality, ensuring continued operation even when failures occur. Furthermore, regular software updates are crucial to mitigating security and safety risks while also enabling the addition of new functionalities post-sale [11]. The architecture proposed in [84] suggests that safety-critical functions should be duplicated across physically separate ECUs, ensuring that the failure of any single ECU does not impact the delivery of safety-critical services. Similarly, in [111], the extension of a centralized ICT architecture recommends deploying multiple processing units in parallel to meet safety requirements through redundancy. Importantly, the development of new safety standards must account for emerging technologies such as AI models, which cannot be fully addressed by existing functional safety criteria like ISO 26262. AI models, often treated as black-box algorithms, challenge traditional assumptions regarding system behavior predictability and, thus, require new approaches to safety regulation in the automotive industry [16].

4.5 Data Transmission

The evolution of vehicle architectures is closely tied to the increasing demands of data transmission driven by AI models and the growing input throughput generated by the expanding array of sensors needed to support these models. As a result, the demand for data bandwidth has surged significantly [124]. Traditionally, vehicle communication relies on the Controller Area Network (CAN) protocol [111]. While CAN has been the dominant standard for decades, it is gradually becoming insufficient for modern needs due to the limited data rates it supports. The growing complexity and data requirements are pushing the industry towards adopting more capable protocols, such as automotive Ethernet, which is becoming the leading standard in the vehicle industry [130]. Although current data transmission technologies have been adequate for traditional vehicle communication, they are expected to fall short in the near future. The sheer volume of data and the increasing number of interconnections between ECUs in advanced vehicle architectures will exceed the capabilities of legacy systems [11]. As automotive systems evolve, the adoption of higher bandwidth and more scalable communication protocols will be essential to support the integration of AI-driven functionalities and the sophisticated sensor networks that modern vehicles demand.

4.6 Software Architecture

The shift towards advanced vehicle functionalities, such as AI-driven systems and autonomous driving, necessitates a fundamental transformation in automotive software architecture. Traditional approaches, while reliable for conventional systems, are increasingly unable to meet the complexity, scalability, and flexibility requirements of modern vehicles. This section explores two key architectural paradigms—layered architecture and service-oriented architecture (SOA)—and their roles in enabling the next generation of software-defined vehicles, focusing on how they can accommodate the unique demands of intelligent and autonomous systems.

4.6.1 Layered Approach

In [137], the authors highlight that integrating intelligent functions into existing vehicle platforms often involves simply adding a layer of intelligence for data processing, planning, and decision-making. However, the limitations of traditional software architectures are insufficient to meet the demanding requirements of AI-based features. Consequently, their proposed architecture is designed from the ground up based on the software requirements defined by the SAE J3016, a recommended practice for building autonomous vehicles. The architecture draws heavily from

established robotics frameworks. Similarly, [111] proposes a layered software architecture for autonomous driving. Layered designs have proven effective in software development as they allow individual layers to be replaced or modified without disrupting the overall system logic. This flexibility enables developers to make updates in one part of the system without affecting other layers. Their proposed architecture consists of three distinct layers: the OS layer, the data transmission layer, and the algorithm layer.

4.6.2 Service-Oriented Architecture (SOA)

According to [32], today's signal-based architectures have reached their limits in handling the complexity of internal communication in modern vehicles. They advocate for a shift to service-oriented architectures (SOA), with services encapsulated within containers and managed through a data-centric publish-subscribe approach. This model aligns with modern software development practices and can address the growing need for rapid software feature deployment, helping automakers remain competitive. [130] underscores the importance of enabling the addition of new software features throughout the vehicle's lifecycle—something that current architectures struggle to support. To achieve this flexibility, a shift to an SOA is essential for transforming vehicles into software-defined platforms. However, the study also highlights emerging security risks associated with SOA. As a practical solution, the authors propose a hybrid architecture combining traditional signal-based communication with service-oriented approaches to support new functions, such as autonomous driving.

4.6.3 Software-Defined Vehicles (SDVs)

The concept of Software-Defined Vehicles (SDVs) represents a major shift, where the functionality of a vehicle is determined by software rather than being tightly integrated with hardware. Mirroring modern E/E architectures, SDVs rely on centralized computing units, with software organized into services that can be independently updated or replaced. Key to enabling SDVs is the adoption of advanced software development frameworks, such as the AUTOSAR Adaptive Platform. This platform addresses critical aspects of SDV, including dynamic and scalable software architecture, over-the-air updates, and integration with high-performance systems through high-bandwidth communication protocols. Despite its modern capabilities, the AUTOSAR Adaptive Platform alone does not meet all automotive needs, particularly when it comes to strict safety and timing requirements, which remain strengths of the classic AUTOSAR platform. For this reason, a hybrid approach that combines both platforms within a service-oriented architecture is recommended [**Bridging2024**, 130]. This hybrid solution can offer the flexibility and performance needed for future automotive systems while maintaining robust safety standards.

5 Strategies for Integrating Generative AI in Automotive Systems

Deploying generative AI effectively within automotive hardware constraints demands innovative strategies. Optimal solutions emphasize edge deployment, specialized hardware utilization, model architecture optimization, and advanced software techniques.

5.1 Edge Deployment

Deploying AI models at the edge minimizes latency and enhances real-time performance, critical for in-vehicle applications. It facilitates localized data processing, reducing reliance on cloud-based solutions and enhancing data privacy and security.

5.2 Hardware Selection and Adaptation

Choosing suitable hardware is crucial, as inference times vary significantly across different platforms. Benchmarking models across various hardware options helps identify optimal choices for specific applications. Adapt-

ing model architectures to fit hardware specifications can further optimize performance, aligning reinforcement learning reward metrics with hardware capabilities.

5.3 Optimized Inference Frameworks

Using dedicated inference frameworks distinct from training frameworks, such as converting from TensorFlow or PyTorch to optimized platforms like TensorRT, significantly improves efficiency. Hardware-specific optimizations in these frameworks enhance performance during inference.

5.4 AI Compilers

AI compilers convert models into Intermediate Representations (IR) like ONNX, generating hardware-specific optimized code. Techniques such as operation fusion and caching optimize conversions, requiring deep compiler understanding. Adapting model architectures to compiler capabilities, including switching activation functions, boosts efficiency.

5.5 Dedicated Accelerators

Utilizing dedicated accelerators like NVIDIA's Jetson Nano with integrated Image Processing Units (IPU) improves efficiency and energy consumption over general-purpose processors. These accelerators are tailored for specific tasks, enhancing performance in AI applications.

5.6 Knowledge Distillation

Knowledge distillation reduces model size and inference time by transferring knowledge from a complex model (teacher) to a smaller, efficient model (student) without significant accuracy loss. Careful selection and architecture development of student networks optimize this process.

By leveraging edge deployment, selecting appropriate hardware, optimizing inference frameworks, employing AI compilers, and implementing knowledge distillation, the automotive industry can effectively integrate generative AI into vehicles, enhancing performance, safety, and user experience.

5.7 Model Optimization

5.7.1 Pruning

In general, pruning methods vary between structured and unstructured approaches. Structured pruning removes groups of parameters to enable the usage of generalized hardware like ARM, CPU or GPU processors. Unstructured pruning targets individual weights and parameters of a model architecture, which leads to a necessity of specialized hardware to gain a benefit from the optimized models. Additionally to the distinction between structured and unstructured pruning, these methods often differ in their use of training data. Traditional pruning methods often require retraining or an iterative processes using training data. This results in high computational efforts and the need of large computing resources, especially in combination with Large Language Models (LLM) and Stable Diffusion (SD). To optimize optimize a model more efficiently, we distinguish the publications between Pruning "with retraining" and "without retraining".

5.7.2 Pruning without Retraining

First, we focus on methods that do not require retraining. These methods offer computing and cost efficient alternatives for a majority of users with access to consumer GPU or datacenter GPUs like V100s [110].

WANDA : A Simple and Effective Pruning Approach for Large Language Models (LLMs) [142]. WANDA combines weight magnitudes with input activations to evaluate weight importance and compares weights on a per-output basis. This pruning method induces sparsity without retraining or weight updates, identifying effective sparse networks within pretrained LLMs. This method outperforms traditional magnitude pruning, achieving comparable performance to state-of-the-art methods while being significantly faster. It even demonstrates high zero-shot task accuracy and lower perplexity in language modeling tasks. Additionally the authors showed that large and sparse networks outperform smaller and denser models in general. A last

and crucial point to be highlighted is the simple application, to almost any kind of model architecture.

BONSAI : Everybody Prune Now - Structured Pruning of LLMs with Only Forward Passes [43]. This is a gradient-free, structured pruning method. It estimates the importance of each module and selectively pruning them using only forward passes. Thereby it allows practitioners to prune models almost without memory any constraints beside the inference requirements that already exist. To further improve pruning itself, it is suggested by the authors to prune in an iterative manner. Even though post-training is not explicitly necessary, a fine-tuning can help to recover some lost performance. For fine-tuning, like many others, LoRA [68] is used to enable an memory-efficient fine-tuning. Similar to the previously introduced WANDA method, a model pruned with BONSAI has very good zero-shot capabilities.

FLAP : FLuctuation-based Adaptive structured Pruning for LLM [6] is a retraining free structured pruning method. The most crucial part of this approach is the "Baseline Bias compensation", a compensatory mechanism for the damage inflicted on the output feature maps by the removed components after pruning. This is applied to channels with a low fluctuation and is combined with a standardized search across different layers and modules of the model to ensure consistent comparison of scores, since these scores can vary greatly throughout the model. This results in an increased inference speed up to 66% for the LLaMA-7B with a pruning ratio of 50%.

F3Pruning : A Training-Free and Generalized Pruning Strategy [140]. This method is designed for Text2Video (T2V) models, which often suffer from high inference costs due to their large size. The authors identified three crucial observations: temporal attention (TA) values approaching zero indicate temporal redundancy, decrease of aggregate TA values implies decline in importance and removing redundant TA leads to reasonable attention redistribution. Attention values directly act as an indication of the weight importance. By pruning these redundant TA-weights, F3Pruning improves inference speed and especially video quality. Evaluations on three datasets demonstrate its effectiveness, achieving a 1.35x speedup in inference time and a 22% improvement in video quality metric FVD on the UCF-101 dataset when applied to the transformers-based model CogVideo.

ZeroTPrune : Zero-Shot Token Pruning through Leveraging of the Attention Graph in Pre-Trained Transformers [156]. The method is divided in two stages: The I-Stage, to evaluate and prune unimportant tokens (like background). They use Weight Page Ranking (similar to Google Page Rank) to determine unimportant tokens and combine this algorithm with the attention graph of the pretrained transformer. The second stage, the S-Stage, determines and prunes tokens that are similar to each other (like repetitive structure tokens of a white wall). As an example of the final results, the method achieve an increased throughput of 45.3% for the DeiT-S model [152] with an accuracy loss of 0.1%.

These methods provide efficient alternatives to traditional pruning techniques by eliminating the need for retraining, thus reducing computational costs while maintaining or even improving model performance.

5.7.3 Pruning with Retraining

Shortened LLaMA : A simple depth pruning for LLM [80]. The authors propose a depth pruning method, a technique for reducing the computational needs of LLMs by removing unnecessary layers or single transformer blocks (multi-attention head with a feed-forward network) from the model. The decision which block should be pruned is determined by a weight importance score based on different metric (e.g. Talyor and Perplexity). The comparison is primarily made for width pruning, like WANDA or FLAP, which seemingly doesn't achieve great speedups while using small batch sizes (e.g. in almost real time applications). The speedup for the LLaMA-7B is 1.5x with a pruning ration of 35% while

WANDA and FLAP seem to decrease the inference time. Additionally, the authors tested two fine-tuning methods, Continued Pretraining (CPT) and LoRA, while CPT requires many computing resources. With CPT, the Vicuna-7B [36] achieves an 1.5 higher average accuracy than WANDA and FLAP with pruning ratios over 50%.

LAPTOP-Diff : Layer Pruning and Normalized Distillation for Compressing Diffusion Models [171]. Different from the other methods, LAPTOP-Diff focus on compressing diffusion models (SDM) used in image synthesis. It introduces layer pruning and a one-shot pruning criterion for automatic model compression. The ensure a good one-shot pruning performance based on a model property called "additivity". It implies that a network's output distortion caused by many perturbations approximately equals the sum of output distortions caused by each single perturbation. The method realizes a one-shot layer pruning algorithm to remove a subset of prunable layers that satisfies a given pruning ratio while minimizing performance decline. Nevertheless, the paper uses retraining to regain accuracy. Results show that compressing the U-Nets of SDXL [117] and SDM-v1.5 [127] with LAPTOP-Diff achieves a minimal 4.0% decline in PickScore at a pruning ratio of 50%, whereas comparative methods result in a minimal PickScore decline of 8.2%.

LoRAPrune : Structured Pruning Meets Low-Rank Parameter-Efficient Fine-Tuning [172]. This publications tackles the problem of Low-rank adaption (LoRA) compatibility with current pruning methods. This is because they use unstructured pruning on LLMs, which prevents merging LoRA weights, or rely on pre-trained weight gradients for pruning, causing significant memory overhead. The key aspect is to combine the LoRA weights and combine them with Taylor expansion to estimate the importance of the weights. Another Additionally to the aforementioned aspects, they quantize the frozen weights to 8-bit, to save even more memory. They achieve a speed up proportional to their pruning rate (50% equals about 50% faster inference time) and can almost match the average performance of the LLaMA-7B model with a 20% pruned model.

LLMPruner : On the Structural Pruning of Large Language Models [99]. This one one of the first structural pruning papers for LLMs, especially considering task agnostic pruning. The first key element is the depency graph to analyze the models structure properly and prune coherent elements. the second key element is the grouped importance estimation for these coherent elements. After pruning the structure they use LoRA to regain the performance of the model. Additionally to the presented approach, the author did quite an extensive analysis of the impact of pruning rates, datasets and aggregation strategies in their method. The practical test including fine-tuning with different models (e.g. LLaMA-7B, Vicuna-7B) was carried out on a single NVIDIA RTX4090 with 24GB, which allows many researcher to test this approach without any sophisticated hardware. The results show that even with a pruning ration of 20%, the pruned model almost maintains 95% of the performance of the original model.

SIMPLE : In [147] the paper introduces a structured pruning framework called SIMPLE (Sparsity-Induced Mask Learning for Pruning Generative Pre-trained Language Models). The research aims to compress large pre-trained language models (PLMs) by examining all compressible components, including attention heads, feedforward networks (FFNs), with a specific emphasis on the hidden dimensions shared across various modules. The proposed approach leverages learnable masks applied to compressible components, followed by sparse training to identify redundant structures within the network. This method facilitates the extraction of PLMs at various scales using different thresholds, with subsequent task-specific fine-tuning for enhanced performance. The approach ranks the significance of compressible components by applying learnable masks to their outputs and optimizing them with a sparsity-induced penalty. These masks are applied in a one-shot man-

ner, allowing for flexible pruning of models by adjusting the thresholds based on the learned mask values. Additionally, a causal distillation objective is introduced to reduce accumulated compression errors and refine the fine-tuning of pruned models. Extensive experiments on language modeling, summarization, and machine translation tasks using GPT-2 and BART/mBART architectures demonstrate the effectiveness of the SIMPLE framework.

Can pruning make Large Language Models more efficient? : This paper [56] explores the use of weight pruning as an optimization strategy for Transformer architectures, focusing on improving computational efficiency, reducing environmental impact, and enhancing deployability on resource-limited platforms. The study examines various weight pruning techniques and their effects on model performance, size, and computational requirements. Weight pruning is presented as a solution to the challenges posed by the large parameter counts in Transformer models. The paper discusses the benefits and trade-offs, noting how pruning can enhance computational efficiency and reduce model size, while requiring careful management to maintain performance. The study compares unstructured and structured pruning approaches, detailing strategies for determining which weights to prune and highlighting the role of key hyperparameters like pruning thresholds, rates, schedules, and retraining epochs. Experimental results, focused on language modeling and question answering tasks, showcase how pruned models perform relative to their original counterparts. The paper concludes by emphasizing the importance of refining weight pruning techniques and integrating them with other optimization methods to develop more efficient and environmentally sustainable deep learning models.

PuMer : PuMer [30] is a token reduction framework designed to optimize large-scale vision-language (VL) models that rely on Transformers for memory-intensive cross-modal interactions between text and images. The primary goal of PuMer is to accelerate inference and reduce the memory demands of these models. PuMer implements two main strategies for reducing input tokens. First, it applies text-informed pruning, which retains only relevant image tokens based on the input text by using text-to-image cross-attention scores to identify important tokens. Second, it uses modality-aware merging, which independently combines semantically similar text and image tokens within each modality, compressing redundant information through a lightweight bipartite soft matching algorithm. These techniques are integrated via non-parametric token reducer modules, added at multiple cross-modal layers of the VL model. These modules progressively prune and merge tokens during cross-modal interactions, thereby reducing the computational burden. Experiments on two VL models across four vision-language tasks show that PuMer improves inference speed by up to 2x and reduces memory usage by over 50%, with less than a 1% drop in accuracy compared to the original fine-tuned models.

ECoFLaP : ECoFLaP [144] (Efficient Coarse-to-Fine Layer-wise Pruning) is a two-stage weight pruning method designed to reduce the computational costs and carbon footprint of large vision-language models (LVLMs). The authors address the limitations of existing layer-wise pruning techniques by introducing a strategy that uses global importance scores to set sparsity ratios across different layers or blocks. ECoFLaP's approach involves a two-stage process: first, sparsity ratios are determined based on global importance scores; then, local layer-wise unstructured weight pruning is applied. This method delivers substantial performance gains in high-sparsity scenarios compared to existing pruning techniques. The paper also tackles the unique challenges of pruning vision-language models, highlighting the architectural complexity and differences in data characteristics across modalities. A detailed evaluation of current global and layer-wise pruning approaches is provided, with an analysis of their limitations and effects on various models and datasets.

GOHSP : GOHSP[169] is a unified framework for vision transformer (ViT) models that tackles the challenge of their large-scale size, which limits deployment in resource-constrained environments. Structured pruning is proposed as an effective method for compressing model size and enhancing practical efficiency. While structured pruning has been widely studied in CNNs and RNNs, it remains underexplored for ViT models. This research highlights the heterogeneous structured sparsity patterns within ViT modules and offers a framework to address these complexities. The proposed solution includes a graph-based ranking algorithm to assess the importance of attention heads and a unified approach to jointly optimize different types of structured sparsity across ViT modules. The framework is validated through performance comparisons between GOHSP and other structured pruning methods, showing GOHSP's superiority in maintaining accuracy while significantly reducing model parameters.

5.8 Quantization

Modern state-of-the-art machine learning models have achieved superb results in various perception tasks such as image classification, natural language processing, speech recognition, and many others. However, the immense computational complexity and vast memory requirements of such models make it extremely challenging to deploy them on resource-constrained embedded systems.

Various compression and optimization techniques have emerged to tackle the above challenges, out of which one of the most commonly applied approaches is neural network quantization. This method defines transformations for calculations between the original floating-point and a more concise fixed-point arithmetic representation. Such a reduction of numerical precision can significantly compress models, with the drawback of potentially losing network accuracy. Various approaches arose to address this limitation by making neural networks more quantization-friendly and trying to maintain floating-point accuracy on fixed-point precision.

A group of techniques, called Quantization-Aware Training (QAT), tries to simulate the arising numerical error already in the training process, while another method called Post-Training Quantization (PTQ) focuses on actually implementing the floating-point to fixed-point transition of a readily trained model in the most lossless way possible. This literature review focuses on the latter approach.

Nagel et al. [106] provide an extensive general overview on quantization and provide generic pipelines for performing both QAT and PTQ for arbitrary networks. Authors claim that PTQ is sufficient in most cases for achieving 8-bit quantization with close to floating point accuracy and should be the preferred way over QAT if the model performance is acceptable. Alternatively, QAT requires fine-tuning and access to labelled training data, but enables lower than 8-bit quantization with decent accuracy. Authors also recommend using asymmetric quantization for activations and symmetric quantization for weights. Per-tensor quantization is most common in practice, except if the distribution of weights varies significantly from channel to channel, in which case it is advisable to use per-channel quantization. This is prevalent in depthwise separable layers since only a few weights are responsible for each output feature. The authors also study the effect of range initialization on QAT performance for ResNet18 and MobileNetV2. They conclude that for models which suffer large performance degradation with plain PTQ may need advanced PTQ techniques like Cross-Layer Equalization (CLE) to initialize QAT. In most other cases, an improved PTQ initialization has little to no-effect on final QAT performance. When using SGD-type optimizers, the learning rate for the quantization parameters should be reduced compared to the rest of the network parameters. To avoid such differential learning rate adaptations, the use of optimizers like Adam or RMSProp is recommended.

Li et al. [96] introduce BRECQ. This novel PTQ approach aims to quantize pre-trained neural network weights to below 8-bit widths without retraining. Firstly, the authors analyze the second-order error caused by quantization using Taylor expansion. They show that the second-order error can be trans-

formed into the outputs of neural network layers, instead of the parameter space, as considered by other methods. Based on this analysis, they propose to reconstruct the outputs of neural network blocks one-by-one during PTQ, as a balance between modeling cross-layer dependencies and avoiding overfitting. This block-wise reconstruction outperforms layer-wise or network-wise approaches. Next, they use advanced calculations (revolving around the Fisher Information Matrix) to represent second-order information. Moreover, they incorporate a genetic algorithm to search for mixed-precision configurations to satisfy hardware performance constraints. Extensive experiments on image classification (ImageNet) and object detection (COCO) show that BRECQ achieves state-of-the-art accuracy for 4-bit and even 2-bit PTQ, outperforming even some QAT methods while being orders of magnitude faster. In summary, BRECQ pushes the limits of post-training quantization through better theoretical modeling and optimization techniques such as block reconstruction and mixed-precision search. This enables extremely low-bit quantized models to be rapidly generated from pre-trained networks.

In [23], Bhalgat et al. aim at tackling the problem of low-bit quantization (2, 3 or 4 bits) of deep neural networks which include activation functions that can take negative values (e.g., Swish). Existing quantization methods such as Learned Step-size Quantization (LSQ) [50] assume unsigned activations, quantizing all negative values to zero and causing hence significant loss of accuracy. In their work, authors propose a learnable asymmetric quantization scheme with trainable scale and offset parameters to accommodate negative activations without using an extra sign bit. This allows quantizing architectures such as EfficientNet with Swish activations at very low bit-widths. Furthermore, they also propose an MSE-based initialization scheme for the quantization parameters that leads to significantly better stability and lower variance in final performance across training runs compared to previous initializations. The aforementioned learnable offset allows the quantization range to adapt separately per layer to the skewed distribution of negative and positive activations. This provides an 1.6-1.8% accuracy improvement over LSQ for 4-bit weight an activation quantization of EfficientNet-B0, and up to 5.6% gain for 2-bit weight and activation quantization. They also propose initializing scales / offsets from the activation statistics rather than actual values, converging closer to the final trained values and exhibiting lower variance across runs, especially at extreme low bit-widths. The asymmetric quantization scheme maintains previous performance on ReLU architectures like ResNet while significantly improving quantization of architectures using signed activations.

LLM.int8() : The authors of LLM.int8() [44] introduce a novel approach to quantize large language models (LLMs) from 16/32-bit precision to Int8 without compromising performance. Their work addresses the challenge of outlier weight values that can arise during quantization, which often lead to degradation in model accuracy. Their approach is based on a hybrid quantization technique involving two key methods. The first method employs vector-wise quantization. Instead of applying a single normalization constant for the entire tensor, this technique utilizes separate normalization constants for each row or column in the weight matrices. By applying matrix multiplication with row/column-specific normalization constants, outlier values in one row or column can be isolated, preventing them from negatively impacting the rest of the matrix. This ensures more stable quantization, even in the presence of outliers. However, the authors observed that vector-wise quantization alone becomes insufficient when quantizing LLMs with over 6.7 billion parameters. For models above this scale, the zero-shot accuracy of the quantized models significantly deteriorates. To address this limitation, they introduce a second technique called mixed-precision decomposition. This method targets outlier parameters by leveraging mixed-precision arithmetic. The authors identified that outliers are systematically concentrated in certain hidden dimensions during the transformer operation. By isolating these high-value parameters and performing matrix

multiplication in FP16 for these specific dimensions, they manage to preserve model accuracy. Since 99.9% of the dimensions do not contain such outliers, the method remains both computationally efficient and memory-friendly.

SmoothQuant : SmoothQuant [162] introduces a novel training-free post-training quantization algorithm that effectively quantizes both the weights and activations of large language models (LLMs) to Int8. By converting both weights and activations to Int8, the quantized model achieves a significant reduction in storage—halving the memory requirements compared to FP16—and also accelerates inference by leveraging integer kernels that are widely supported across various hardware platforms. The authors emphasize that while quantizing activations in LLMs with fewer than 6.7 billion parameters is relatively straightforward, extending this process to larger models poses challenges due to the presence of outlier values across different activation channels. This challenge makes activation quantization non-trivial in these larger models. To address this issue, the authors propose a novel approach that shifts the quantization difficulty from activations to weights through a mathematically equivalent per-channel scaling transformation. This technique redistributes the complexity, making activation quantization more manageable while introducing only a slight increase in the difficulty of weight quantization, which remains within a tractable range. This trade-off allows SmoothQuant to effectively quantize both weights and activations while maintaining high performance. Compared to other methods that rely on mixed-precision activation quantization, SmoothQuant significantly reduces both inference latency and memory usage, making it a highly efficient solution for deploying large-scale LLMs.

AQLM : AQLM [48] is a post-training quantization algorithm specifically designed to perform extreme quantization on the parameter weights of large language models (LLMs). This algorithm achieves compression levels as low as two to three bits per parameter, making it one of the most effective techniques for reducing model size while maintaining performance. According to the authors, AQLM is the first quantization method that is Pareto-optimal in terms of model size versus accuracy when the quantization is performed at less than three bits per parameter. AQLM builds on prior work in the area of Nearest Neighbor Search (NNS) optimization, applying multi-codebook quantization to compress model weights. The technique works by dividing each vector in key weight matrices into sub-vectors, which are then approximated using a finite set of learned vectors. These learned vectors are stored in codebooks, and the original weight can be represented by selecting a single codeword from each codebook and summing them. In this manner, the algorithm encodes each weight efficiently, leading to significant reductions in model size. While AQLM sets a new benchmark in representing weight parameters using only 2 to 3 bits, the authors acknowledge some limitations. Despite improving the state-of-the-art in extreme quantization, AQLM is computationally more demanding than other post-training quantization methods due to the complexity of its multi-codebook representation. This increased computational cost may present challenges for real-time deployment and inference, especially in resource-constrained environments.

HQQ : In Half-Quadratic Quantization of Large Machine Learning Models [12], the authors propose a novel quantization technique known as Half-Quadratic Quantization (HQQ). This method introduces a data-free calibration approach, addressing the limitations associated with using external calibration datasets. According to the authors, relying on external datasets can introduce biases that negatively affect the generalization capabilities of the model. Additionally, using a calibration dataset for large-scale models is computationally expensive, making it impractical to evaluate multiple models efficiently.

To achieve effective quantization without external data calibration, HQQ focuses on minimizing quantization error primarily in the weight parameters rather than in the acti-

vation layers. The method incorporates a sparsity-promotion loss, which better captures the distribution of outlier weights, leading to more accurate quantization. To optimize the quantization parameters, the authors adopt a Half-Quadratic splitting optimization approach, which efficiently solves the proposed optimization formulation. To validate their method, the authors benchmarked HQQ against existing quantization techniques on both vision models and large language models (LLMs). In vision model evaluations, HQQ was tested on the ImageNet dataset and compared with the BitsAndBytes (BNB) approach. The results demonstrate that HQQ significantly outperforms BNB in low-bit quantization scenarios, achieving a +3.1% improvement in top-1 accuracy with a 4-bit ViT-B-32 model in zero-shot performance. For extreme low-bit quantization, HQQ's 3-bit ViT-H-14 even surpasses the full-precision ViT-L-14 by +2.4% top-1 accuracy, while its 2-bit version outperforms the full-precision ViT-B-32 by +5.2% top-1 accuracy. For LLM validation, HQQ was compared against BitsAndBytes, GPTQ via AutoGPTQ, and AWQ via AutoAWQ, using the perplexity metric on the WikiText dataset. HQQ demonstrated strong performance in terms of perplexity and memory efficiency, showing its competitiveness in both vision and language models under low-bit quantization conditions.

ViDiT-Q : The authors of *ViDiT-Q*[176] introduce a quantization method specifically tailored for diffusion transformers (DiTs), addressing the challenges these models face when deployed on edge devices. Diffusion transformers, particularly in multi-frame video/image generation, demand substantial computational and memory resources, making efficient deployment difficult. The authors observed that existing quantization techniques developed for U-Net-based convolutional neural networks (CNNs) cause significant quality degradation when applied to DiTs. To overcome this, they propose a method based on three quantization schemes designed to address the four key levels of data variance they identified as critical for effective DiT quantization.

1. **Token-Wise Quantization**: The first quantization scheme targets the notable token-wise variance observed in diffusion transformers. The activation tokens exhibit substantial differences, especially in DiT's linear layers, which account for over 77% of latency and nearly 100% of memory consumption. By quantizing tokens individually, this method mitigates variance-induced performance loss.

2. **Dynamic Quantization**: The second scheme dynamically calculates quantization parameters online for activation quantization, addressing challenges related to classifier-free guidance variance for conditional generation and timestep-wise variance. Timestep-wise variance refers to the significant changes in activations across the different stages of the diffusion process, which dynamic quantization effectively handles.

3. **Timestep-Aware Channel Balancing**: The third component introduces timestep-aware channel balancing by applying different masks for various timesteps. This scheme is designed to manage channel-wise variance, which stems from differences in weights and activations across channels and timesteps throughout the diffusion process.

While ViDiT-Q improves quantization performance, the authors noted that lower bit-width quantization can still degrade model quality. Upon investigation, they found that certain layers, even those with low quantization error, are highly sensitive to quantization. To address this issue, they propose a mixed-precision model called ViDiT-Q-MP. In this approach, higher bit-width quantization is assigned to these sensitive layers, preserving model accuracy while still benefiting from the efficiency of lower bit-width quantization in other layers.

AWQ : The authors of AWQ [97] emphasize the growing need to deploy large language models (LLMs) on edge devices to reduce

cloud computing costs and enhance user privacy. To address the hardware limitations of such devices, they propose a quantization approach designed for resource-constrained environments. The authors argue that not all model weights are equally important, and that by preserving only 1% of the most critical weights, the model size can be significantly reduced with minimal impact on accuracy. Unlike conventional approaches that focus on weight distribution, AWQ prioritizes the model's activation distribution when identifying these critical weights. The rationale is that weight channels associated with larger activation magnitudes process more essential features, making them more crucial to maintain during quantization. AWQ employs a per-channel scaling method that automatically searches for the optimal scaling factors to minimize quantization error. This process is data-driven and does not rely on additional training or regression techniques, which allows the quantized model to maintain its generalization capabilities. By effectively preserving the most significant weights while applying optimized scaling to the remaining parameters, AWQ enables efficient deployment of LLMs on edge devices without sacrificing performance.

5.9 Low-rank approximation

Low-rank approximation is a technique in linear algebra and numerical analysis used to approximate a matrix with another matrix that has a lower rank. This is useful in reducing the complexity of the data while retaining its essential features. Low-rank approximation is also applied to tensors, which are generalizations of matrices to higher dimensions. Tensor compression using low-rank approximation is essential in various fields, including machine learning, signal processing, and computer vision, where high-dimensional data is common.

A tensor is a multi-dimensional array. For example: A vector is a 1st-order tensor, A matrix is a 2nd-order tensor, A 3rd-order tensor could be a 3-dimensional array. Just like matrices can be decomposed using SVD, tensors can be decomposed using several methods. The most common tensor decomposition techniques for low-rank approximation are

1. Canonical Polyadic Decomposition (CPD)
2. Tucker Decomposition
3. Tensor Train (TT) Decomposition
4. Tensor Ring Decomposition
5. Term Decomposition (BTD)

This section gives an overview of the current state-of-the art publications speaking about the widely used and implemented tensor compression techniques for neural network compression.

Low rank Expansions: The authors introduce two schemes in this paper [71] to express the weight kernels in the CNN layers as a linear combination of low rank kernels. The first scheme exploits the spatial dimension of the kernel by decomposing it as rank one filters. The second scheme takes advantage of redundancy in both the input and output channels. In both these schemes, a set of basis filters have been used that is smaller than the total number of output filters. This reduced weight kernel sizes are then approximated by performing local fine-tuning operation on those decomposed layers. Their optimization techniques resulted in 2.5x times speedup of the AI model with nil accuracy loss and a tradeoff of 1% classification accuracy for 4.5x speedup improvements.

Low rank regularization: The authors of this paper [145] have proposed an algorithm to determine the low rank tensor decomposition of the convolutional kernels. Instead of the iterative methods used in CP decomposition for determining the rank, this method gives a closed form solution as it uses Singular Value Decomposition. This is because obtaining an exact solution is much more effective than using iterative method.

In this method, the low rank approximation is performed on the trained kernel weights. The 4D convolutional kernel is reduced to 2D matrix and SVD is applied on these weight matrices, which is converted back to two 4D weight kernels. Therefore, a single convolution layer is expanded into two consecutive layers,

which are then fine-tuned to improve the lost accuracy. To overcome the issue of exploding or vanishing gradients, which might arise due to increased number of layers, the authors have utilized Batch Normalization.

By using this technique, they have achieved 91.31% accuracy in CIFAR-10 dataset using low rank NIN model. Apart from that significant speedup and better results are obtained from other CNN models such as AlexNet, VGG, GoogleNet using the dataset CIFAR10 and ILSVRC12.

Exploiting linear structure within CNN: This method in [42] is introduced for improving the test time evaluation of large convolutional neural networks. The author focuses on two different techniques that can be applied for the first CNN layer with colour channels and another to handle redundancy in other higher CNN layers by applying filter clustering on these layers. Both these methods aim to reduce the over-parametrization by exploiting the linear structure.

Methods like SVD and filter clustering have been used in this methodology to reduce the size of weight kernels in the CNN layers. A reduced form of Mahalanobis distance metric and data covariance distance metric is used as the approximation criterion of these weight kernels. The tensor approximation is performed using either matrix decomposition where first three kernel dimensions are considered and folded accordingly to perform SVD or all the three dimensions are decomposed using the iterative outer product decomposition.

They have achieved a speedup on both CPU and GPU while keeping the accuracy loss within 1% of the original model. The number of parameters is reduced by a factor of 2 to 3 times in CNN layers and 5 to 10 times in the fully connected layers.

GSVD: Instead of directly approximating the linear filters in the CNN models, the methods described in this paper [174] also include non-linear units. Unlike the previous methods, where the weight kernels are directly approximated, this method utilizes the low rank space of the output response to approximate the layer weights. After that, the corresponding non-linear units are also approximated based on the Generalized Singular Value Decomposition (GSVD) to perform asymmetric reconstruction that accounts for the error from previous approximated layers. A Rank selection is proposed, which is based on empirically finding that there is a relation between PCA energy and Classification accuracy.

The methods described in this paper reduces the errors accumulated in a very deep network due to weight approximations. A model speedup of 4x is achieved in VGG-16 model with an increase of 0.3% in top-5 error using ImageNet dataset.

CP Decompositions: To limit the size of over parametrized CNN models, the authors in [87] have utilized tensor- linear algebra concept to decompose the layers. Each of these layers with 4D weight kernels have been approximated using low rank Canonical-Parafac decomposition (CP).

This process is a two-step approach where the weight kernels in the convolutional layers are decomposed into four smaller rank 1 tensors and then are replaced on the corresponding layer. The rank values for CP decompositions are fixed by using non-linear least squares (NLS) optimization, which minimizes the L2-norm of the approximation residual for a user-defined fixed rank. The same procedure is performed in all the layers, which are then fine-tuned completely to improve the model accuracy. In the CP-decomposition operation to approximate the kernel values non-linear least squares technique is used.

The approach obtains 8.5x CPU speedup of the whole network with only minor accuracy drop of 1% on a 36-class dataset.

CP with ALS algorithm: A tensor could be approximated using CP decomposition by applying alternating least square technique, which solves a series of linear equation. The authors in this paper [104] have used QR decomposition (decomposing a matrix into an orthogonal Q and upper triangular R matrices) technique along with SVD to compute the minimum norm solution. By applying this decomposition within CP-ALS, computation cost is slightly increased but a numerically sta-

ble solution is achieved. This technique could be better suited where CP-ALS is used for solving normal equations when ill-condition is detected.

Stable Low-Rank CP Decomposition: Two of the major problems of weight kernel approximation in CNN using CP-decomposition has been addressed in this paper [166]. To select a suitable rank for the tensors, sensitivity of a convolutional layer is determined and to improve the stability of this approximation optimization is performed with this sensitivity constraint. This sensitivity value dictates the order in which each of the layers has to be approximated followed by iterative fine-tuning. Using ImageNet dataset in VGG and ResNet competitive results are obtained in both compression size and accuracy. However, this sensitivity calculation is based on a heuristic metric and iterative fine-tuning is computationally expensive.

Compression of deep CNNs: The authors in this paper [81] have used large convolutional networks such VGG Net, Google Net have been compressed using tucker decomposition on the trained weight tensors. The ranks for this decomposition are determined by Bayesian matrix factorization based on which the tucker decomposition is applied to a particular layer. Tucker-2 decomposition is applied for all the higher convolution layers in which the mode-3 and mode-4 matricization are used as the factor matrices. One-shot fine tuning is performed once all the layers have been decomposed.

Tensor Train Decomposition: The authors in [113] describe another efficient tensor decomposition method called Tensor Train technique. Tensor decomposition using Canonical method suffer from the problem that finding a tensor rank/canonical rank for the approximation is NP hard problem. In tensor train method, a tensor is approximated using a collection of three-dimension tensors whose rank is calculated from certain auxiliary matrices. A series of unfolding operation and rank determination of these matrices using SVD is performed. Since, these operations does not involve any recursion but use a sequence of QR and SVD approximation techniques it gives numerical stability while reducing the weight parameters.

Tensor Regression networks: The last layer in an image classification model flattens the CNN layer and convert it into a fc-layer. Due to this the spatial information is lost and parameters are increased in this method. Tensor regression method as described in [83] is a special layer that preserves multi-modal information by replacing flattening and fc-layer. It enforces a low tensor rank structure on weight tensors in order to both reduce memory usage and to leverage the multilinear structure of the input. In the first step, tensor contraction is performed using tucker decomposition where, the last convolutional layer crushed to core tensors and factor matrices. In the second step tensor regression maps the feature map to the scalar output through a low-rank weight tensor. Preserving this multi-linear structure have proved to be helpful in MRI datasets obtained from U.K. biobank.

Compact and Computationally Efficient Representation of Deep Neural Networks: This technique in [161] introduces methods to reduce memory requirement and algorithm complexity involved in the dot product computation in weight matrices. By efficiently representing the weight matrices data structure with the help of low entropy statistics higher compression ratio is achieved. Both the dense representation and the sparse representation does not efficiently exploit the internal structure in weight matrices and two new matrix representation has been proposed namely, Compressed Entropy Row (CER) and Compressed Shared Elements Row (CSER) for achieving high compression and energy savings if the weight matrix distribution has low entropy. Experimental results show that 42 times compression ratios, 5 times speed ups and 90 times energy savings can be achieved by compressing the weight matrices using shared weight data structure.

Tensor Regression Networks with various Low-Rank Tensor Approximations:

In this paper [31], the fully connected layer is decomposed into Tensor Regression Layer (TRL). The author then emphazies a comprehensive study on the compression efficiency and regularization power of different low rank decomposition techniques such as Tucker, CP, Tensor Train that is applied to the TRL layer. This performance is compared with the Global Average Pooling Layer (GAP). When CIFAR-10 and CIFAR-100 dataset are trained in ResNet model, GAP layer showed better compression and accuracy than CP-TRL. Overall, the experiment results showed that the TRL performed better in shallow network while the GAP layer maintained better balance in compression and accuracy in deep networks using MNIST and CIFAR dataset.

Efficient Neural Network Compression: In this paper [79], the author introduces a method for optimizing the rank configuration of neural network layers using novel accuracy metrics that represent the relationship between accuracy and complexity. Unlike traditional methods that consider each layer separately, this approach evaluates the entire network to find the best rank configuration. The method is non-iterative, allowing for quick determination of the optimal configuration that meets constraints on FLOPs (Floating Point Operations) and memory while maintaining accuracy.The paper highlights significant efficiency improvements with the VGG-16 model, achieving 25% reduction in FLOPs and a 0.7% improvement in accuracy compared to the benchmark value. Remarkably, the process takes only 3 minutes on a CPU, a drastic improvement over the 4 hours needed with 8 GPUs by previous methods. The proposed method ia applicable for both lossless and lossy compression, making it highly suitable for real-world applications where computational resources and time are critical.

Compression - Learning the rank of each layer: In this paper [70], the authors propose a mixed discrete-continuous optimization framework to simultaneously optimize the ranks and the matrix elements allowing for an efficient search of optimal ranks.In each iteration, the rank for each layer is updated by evaluating the impact of changing the rank on the network's performance. Then, the weight matrix is updated with the low rank approximation and the respective performance is evaluated. The algorithm iterates through the rank adjustment and weight matrix update steps until convergence criteria are met. This approach showed significant improvements in selecting optimal rank values. They demonstrated that a compressed VGG network achieved faster performance than a ResNet with nearly the same classification error, using the proposed method.

Exploring Low Rank Training of Deep Neural Networks: Approximating the weight matrices by low ranks is usually performed after training for the deployment purpose. Many of the low rank approximation techniques decomposes the tensors into lower rank, apply SVD for weight initialisation for fine-tuning operation. This paper [74] has done a detailed study on the gap in performance when a low rank tensor is trained from the scratch and empirically demonstrate how spectral initialisation (a type of weights initialization) and L2 regularisation (training a decomposed network with different learning rates) improve performance.

DiscriminAtive Masking (DAM): As an improvement to the techniques introduced by [161], low entropy matrices are compressed using Huffman coding, which also reduces the computation complexity. Two novel methods were introduced in this paper [26], Huffman Address Map compression (HAM) and sparse Huffman Address Map (sHAM). These methods create binary tree to represent characters based on their frequencies in the input data. They start by determining the frequency of the unique address present in the weight data and using node techniques they create root of the Huffman tree, where Huffman binary codes are generated. Now, the address is replaced by the Huffman code, which results in the compressed representation of the address map.

Tensor Ring Decomposition: This technique in [175] has been developed as an up-

grade to the Tensor Train decomposition, which defines a method to arrange the tensor dimension in optimal format. Instead of the sequential multilinear product utilized in tensor train, a circular multilinear product is performed to treat all the cores equivalently. The 4D kernel weight tensor could be represented as three consecutive layers. Further for the core optimization, SVD and ALS-schemes are used and the entire process is verified to be inline with the classic multilinear algebra concepts. The total number of parameters are linearly dependent on the tensor order and experiments on various datasets shows that TR decomposition are effective for unsupervised feature representation because of their flexible TR-ranks and high compression ability.

Using Neural Architecture Search for tensor compression

Neural Architecture Search (NAS) is a promising technique for designing optimal neural network architectures tailored to specific requirements such as latency, memory usage, and power consumption. NAS automates the process of finding the best architecture and hyperparameters, which is particularly beneficial for the automotive industry where manual tuning is resource-intensive.

The general idea behind NAS involves exploring a space of potential architectures and selecting the optimal one based on an objective evaluation scheme. Notable models like MobileNet and EfficientNet were discovered using NAS, demonstrating its potential to produce highly efficient networks.

In the paper [151] "Using Network Architecture Search for Optimizing Tensor Compression", the author explores the integration of Network Architecture Search (NAS) with Tensor Compression (TC) techniques to optimize convolutional neural networks (CNNs) for embedded systems. A novel approach is proposed to control per-layer parameters of tensor compression using Tucker decomposition algorithm through NAS. The experiments conducted on vision processing benchmarks validate the effectiveness of the approach, showcasing considerable improvements in model compression and inference speed with negligible loss in accuracy. The proposed method achieves a reduction in parameter size by up to 85% with a minimal loss in accuracy (0.1-1% in Top-1 accuracy). The compressed models occupy only 20% of the original memory size required for the uncompressed models and increase inference speed by up to 2.5 times.

However, traditional NAS techniques are computationally demanding and costly, making them challenging to implement at scale. Future advancements in NAS are likely to focus on improving the efficiency of the search process, potentially through techniques such as reinforcement learning and evolutionary algorithms, to reduce computational requirements and costs.

6 Validation of generated sensor data

6.1 Problem Description

The following section gives a mathematical formulation of the problem as well as a high-level introduction to statistical divergences as distribution-based quality metrics for generative models.

6.1.1 Setup

The problem of quality assessment of sensor data can be formalized mathematically in the language of probability theory and statistics. Let us assume the existence of a process that generates real world data. In our situation, this might be a stream of data generated by a radar sensor on a vehicle. The sensor has a certain output range and its measurements take values in $\mathcal{X} \subset \mathbb{R}^d$. The data generating process which we would like to model can be thought of as a probability measure P on $\mathcal{X}$ assigning high probabilities to measurements that will frequently occur and low probability to rare data. A generative model for P is a second probability measure P_G. Both processes can be used to obtain data, or in the language of statistics, can be sampled from. That is, we assume the existence of two random variables $X_R \sim P$ and $X_G \sim P_G$ governed by to the two processes, which can be measured repeatedly. A set $X_{R,1}, X_{R,2}, ..., X_{R,N} \in \mathcal{X}$ of measurements of X corresponds to real-world measurements collected from an actual sensor

while a second set $X_{G,1}, X_{G,2}, ..., X_{G,M} \in \mathcal{X}$ sampled according to P_G corresponds to a set of synthetic data which is much more cost efficient to obtain. Quality assessment of generative models now amounts to quantifying how far apart P_G is from P. In other words, we need to calculate a distance between P and P_G. The difficulty is now two-fold. We need to define a suitable distance $d : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$, where $\mathcal{P}(\mathcal{X})$ is the set of probability measures on $\mathcal{X}$ and subsequently, we need to estimate it.

6.1.2 Statistical Divergences

Distances between probability distributions are commonly referred to as statistical divergences. Although, the term is used somewhat loosely and the precise meaning may depend slightly on the context, for our purposes we will stick with the very general definition that a divergence is any functional $d : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}^{\geq 0}$ such that for any $P, Q \in \mathcal{P}(\mathcal{X})$

(a) $d(P, Q) \geq 0$,

(b) $d(P, Q) = 0 \Leftrightarrow P = Q$.

Let us denote the densities of two probability measures $P, Q \in \mathcal{P}(\mathcal{X})$ by p and q, respectively. Popular examples of divergences include the Kullback-Leibler divergence

$$d_{\mathrm{KL}}(P, Q) = \int_{\mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right), \quad (1)$$

which can be interpreted as the expected loss of information due to approximating P by Q, the total variation distance

$$\delta(P, Q) = \sup_{A} |P(A) - Q(A)|, \quad (2)$$

which measures the maximal difference in assigned probabilities under some event A and the Wasserstein metric

$$d_{W_p}(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \left(\int_{\mathcal{X}\times\mathcal{X}} d(x, y)^p \mathrm{d}\gamma(x, y)\right)^{\frac{1}{p}}, \quad (3)$$

where $\gamma \in \Gamma(P, Q)$ is the set of all couplings of P and Q. This metric can be interpreted as quantifying the minimum cost of transforming one distribution into the another, where the cost is computed based on the amount of distribution weight that must be moved. Other popular choices for divergences include reverse Kullback-Leibler, Hellinger, Bregman and χ^2.

6.1.3 Choice of divergence

Defining a suitable concept of a distance between probability measures requires knowledge of the domain at hand, i.e. the type of sensor. Divergences can be used to either directly quantify the distance between model and target distribution or they can be a building block of a more complex quality metric. The choice of divergence will greatly impact the resulting quality assessment and can be used to reward or punish certain properties of the model distribution. For example the Kullback-Leibler (KL) divergence $d_{\mathrm{KL}}(P, P_G)$ will punish P_G for assigning low probability to events where P assigns very high probability. Hence, low KL divergences indicates good coverage of the target distribution, i.e. no modes are overlooked by the model. The reverse direction $d_{KL}(P_G, P)$ punishes P_G strongly in situations that appear likely under P_G but unlikely under P. Hence, low reverse KL divergence indicates concentration on modes of the target distribution with little probability mass outside of it.

Similar intuitions can be given for other divergences. Look for example at the χ^2-divergence

$$d_{\chi^2}(P, Q) = \int_{\mathcal{X}} \frac{(p(x) - q(x))^2}{q(x)}. \quad (4)$$

Here, the integrand becomes very large if the model $q(x)$ strongly underestimates the true probability $p(x)$. Hence, it is favorable to use in situations where missing a high probability event is associated with great adversity. As different divergences measure different things, it may be worthwhile no to focus on a individual divergence, but to evaluate multiple ones to get a more complete picture of the situation.

6.1.4 Divergences in feature space

Furthermore, divergences must not be used as-is. They can be a building block of a

more complex concept of distance. Also, divergences do not necessarily have to be computed on the original data space $\mathcal{X}$. It is a common practice to map the observed or generated data into a lower-dimensional feature space $\mathcal{F}$ first, before computing a distance, a likelihood ratio or any other quantity of interest. This is due to the fact that lower level statistics like peak signal to noise ratio (PSNR), structural similarity (SSIM), mean squared error (MSE) and the like operate on the raw data without any consideration of semantic information within the data, in this case pictures. They mostly respond to differences in brightness, saturation, color and other physical attributes of the image which are suitable to detect on a per pixel basis. However, they might have little accordance with whatever metric is being optimized during the modelling process. For example in the context of image generation, where the goal is to generate images that appear to be realistic to humans, slightly deforming or rotating an object can yield images which humans perceive to be almost identical, yet giving drastically different low level statistics. Perception metrics try to overcome this obstacle by incorporating machine learning components into the metrics themselves. The idea is to learn a mapping of raw observations into some more meaningful representation in a lower dimensional feature space $\mathcal{F}$. The hope is that in the feature the notation of distance better respect the structure of the underlying data. Mathematically, this amounts to learning a feature map $f : \mathcal{X} \to \mathcal{F}$ and defining a distance in the feature space $d_f(x, y) := d_{\mathcal{X}}(f(x), f(y))$ based on a distance d in the original space.

6.1.5 Computing divergences

After agreeing on a useful distance metric, it might be difficult to compute its value. Probability distributions Q on $\mathcal{X}$ are essentially determined by their respective probability density function $q : \mathcal{X} \to [0, 1]$ which captures information about how likely a specific observation is. In most practical applications however, we do not have access to the density. That means, for a given observation $x \in \mathcal{X}$ we are most likely not able to tell how likely that specific observation is under the process P. A similar situation holds true for the model distribution P_G. Here, depending on the generative model in use we might have access to the model density for example when using VAEs or Normalizing Flows. However, for many generative models for example GANs or Diffusion models we do not have access to the density either. Hence, we rely on samples $X_{R,1}, ..., X_{R,N}$ and $X_{G,1}, ..., X_{G,M}$ respectively to estimate the divergence $d(P, P_G)$. There exists are a variety of estimators to compute these divergences from samples, e.g. [116], [37], [46]. Nevertheless, it must be noted that computing exact divergences is a difficult problem for most popular divergences and may in some cases require exponentially many samples [103]. Another problem when using statistical divergences to this end, is that of disjoint support. That is, if both distributions generate data which would have been impossible under the respective other one. In this situation some of the most popular divergences can be infinite or undefined due to our inability to compute likelihood ratios. This is problematic because two generative models P_{G_1} and P_{G_2} can be of vastly different quality. However, if both models output data which would not quite have been possible under the real distribution both could end up having the same pessimistic divergence $d(P_{G_1}, P) = d(P_{G_2}, P) = \infty$ obfuscating possible differences in quality of results. This underscores the importance of careful design of the distance function for the respective use case.

6.2 Metrics for Generated Image/Camera Data

6.2.1 Distribution comparison

Inception Score (IS) Salismans et al. proposed the Inception Score (IS) [135] as a metric for measuring both quality and diversity of generated images

$$\mathrm{IS} = \exp(\mathbb{E}_x \, d_{\mathrm{KL}} \left(p_\theta(y|x), p_\theta(y) \right), \quad (5)$$

The first term in KL divergence $p_\theta(y|x)$ is the conditional class distribution where an Inception network trained on ImageNet is used. The second term $p_\theta(y)$ is the marginal class

distribution

$$p_\theta(y) = \int_x p(\theta)(y|x)p_G(x)\mathrm{d}x, \tag{6}$$

where $p_G(x)$ is the distribution of generated samples. The IS becomes large if two conditions are fullfilled. 1. $p_\theta(y|x)$ is close to a Dirac-δ-distribution 2. $p_\theta(y)$ is close to a uniform distribution. The IS has the advantage that no samples are needed from which the generator model was trained on. Disadvantages are that it was developed for generate images showing a class from the ImageNet class list. It was not developed for evaluating images in the automotive domain where multiple objects occur in the same image.

Fréchet Inception Distance Because of the shortcomings of the IS for more complex image generation tasks, Heusel et al. introduced the Fréchet Inception Distance (FID) [65]. The FID compares the statistics of real-world and generated samples by computing the Wasserstein-2 distance between two multivariate Gaussians in $\mathbb{R}^n$ with regard to the L^2-norm. First the dimensionality of the samples is reduced by encoding them with the Inception network and use only the feature vectors from the third pooling layer. The mean μ_G and μ_R and the covariances Σ_G and Σ_R for the feature vectors of the encoded generated and real-world samples. The Fréchet distance is explicitly solvable under the assumption that the feature vectors of the generated and real-world are normally distributed with $\mathcal{N}(\mu_G, \Sigma_G)$ or $\mathcal{N}(\mu_R, \Sigma_R)$ respectively and therefore the FID is calculated by

$$\begin{aligned}\mathrm{FID} =& \|\mu_G - \mu_R\|_2^2 \\ &+ \mathrm{Tr}\left(\Sigma_G + \Sigma_R - 2\left(\Sigma_G\Sigma_r\right)^{\frac{1}{2}}\right).\end{aligned} \tag{7}$$

Heusel et al. state that at least as many samples as the dimensionality of the feature vector should be used but reliable results are achieved when using more then 10,000 samples[1].

Kernel Inception Distance The Kernel Inception Distance (KID) [24] is based on the Mean Maximum Discrepancy (MMD) [59].

[1]Corresponding github repository of [65]

The MMD is a distance measure between two probability distributions $P(X)$ and $Q(Y)$. For the KID score the MMD is typically calculated on the feature vectors of an Inception V3 network where the Inception network is used as a mapping function ϕ from the image space to the feature space. With m samples x from $P(X)$ and n samples y from $Q(Y)$ the MMD is calculated via

$$\begin{aligned}\mathrm{MMD}(P,Q) =& \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}k(\phi(x_i),\phi(x_j) \\ &+ \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}k(\phi(y_i),\phi(y_j) \\ &- \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}k(\phi(x_i),\phi(y_j),\end{aligned} \tag{8}$$

with the kernel k. Typically a polynomial kernel e.g.

$$k(\phi(x),\phi(y)) = \left(\frac{1}{2}\phi(x)^T\phi(y) + 1\right)^3 \tag{9}$$

is used [24].

Memorization-Informed Fréchet Inception Distancce One problem of the FID score is, that good FID scores can be achieved by memorizing the training samples. To account for this shortcoming Bai et al. proposed the Memory-Informed Fréchet Inception Distance (MFID) [14]

$$\mathrm{MFID} = \frac{\mathrm{FID}(\mathrm{S_G}, \mathrm{S_R}}{s(S_G, S_R)}, \tag{10}$$

where s is the memorization distance between a set of generated samples S_G and a reference set of real samples S_R given by

$$s(S_G, S_R) = \frac{1}{|S_G|}\sum_{x_G\in S_G}\min_{x_R\in S_R}\left(1 - \frac{|\langle x_G, x_R\rangle|}{|x_G|\cdot|x_R|}\right). \tag{11}$$

6.2.2 Performance on Downstream Tasks

Another plausibility method that is available for labeled data is the evaluation on downstream tasks. E.g. the pix2pixHD GAN model was evaluated on the generated

Cityscapes data with a semantic segmentation model, that was trained on real world data [157]. The mean intersection over union scores are reported as a quality measure how well a model trained on real data performs on synthetic data.

A similar approach is to train a model on synthetic data or a mixture of real and synthetic data and observe the downstream task performance on real data. Burdorf et al. trained object detectors on various ratios of real and synthetic data [28]. They analyzed the performance of the object detectors for varying amounts training samples. The improvement of the performance of the object detector with the increasing number of training examples varied with the mixture ratio of real and synthetic data in the training dataset, thus indicating the quality of the training dataset.

Fan et al. did an extensive study on various image generation models [51]. They investigated the classification performance of a model trained and evaluated on the Imagenet dataset vs. models trained on generated images by Stable Diffusion [128], Imagen [131], and Muse [33]. Similar to [28] they introduced the scaling ability as a metric to compare the three models with different methods for generating text prompts of classifier free guidance scales.

6.3 Metrics for Sequential Sensor Data like Videos

For more details, please refer to Subsection 7.2.2 on Video Quality Assessment.

6.4 Model Specific Plausibility Methods [by Mert]

Many generative models have been used to generate sensor data for autonomous driving systems. In the field of sensor data generation, the approaches can be broadly categorized into two methods: world models [2, 67, 93, 158] and simulation methods [54, 100, 108, 143]. The plausibility metrics used to evaluate these methods also differ based on the specific approach.

Simulation methods can produce both stationary and moving objects in the autonomous driving environment. However, world models offer a more holistic approach. They not only generate static and dynamic elements within a scene, but also predict how these objects will change and interact over time. For this reason, the plausibility metrics of world models must also capture the dynamic elements in the environment, which differs from the metrics used for simulation methods.

6.4.1 Plausibility Methods for World Models

In this section, we cover the methods that are specific to and required by the generated world models. General metrics such as Fréchet Inception Distance or Inception Score are covered in Section 7.2.

Key Point Matching: These metrics assess the quality of the generated data. Common quality metrics include: Key Point Matching (KPV), as outlined in [159], is a method for assessing the geometric consistency and accuracy of generative models that produce multiview data from a single frame. This evaluation technique focuses on the alignment and correspondence of key points across different views generated by the model.

The procedure starts with the extraction of key points from each view using a robust key point detector, such as SIFT or ORB. Let V_1 and V_2 be two views generated by the model, and K_1 and K_2 denote the sets of key points detected in these views, respectively.

The core of the KPV method involves matching these key points across views. The matching process typically involves calculating the Euclidean distance between the descriptors of key points in K_1 and K_2 and then identifying the closest pairs. The matching precision P_{KPV} and recall R_{KPV} are computed using the following formulas:

$$P_{\text{KPV}}(V_1, V_2) = \frac{\text{number of correct matches}}{\text{number of detected matches}} \tag{12}$$

$$R_{\text{KPV}}(V_1, V_2) = \frac{\text{number of correct matches}}{\text{number of true matches}} \tag{13}$$

These metrics, P_{KPV} and R_{KPV}, represent the precision and recall of key point matches

between the views, respectively. The overall KPV score, which combines these metrics, is calculated using the F1-score:

$$\mathrm{KPV}(V_1, V_2) = 2 \cdot \frac{P_{\mathrm{KPV}}(V_1, V_2) \cdot R_{\mathrm{KPV}}(V_1, V_2)}{P_{\mathrm{KPV}}(V_1, V_2) + R_{\mathrm{KPV}}(V_1, V_2)} \tag{14}$$

The KPV score thus offers a quantitative measure of how geometrically consistent the generated multiview images are. As demonstrated in [159], the KPV method effectively highlights discrepancies in key point correspondences, providing a robust metric for evaluating the spatial accuracy and consistency of multiview generative models.

7 Validation of generative scenarios and planning algorithms

7.1 Introduction

This chapter gives an overview over different measures and approaches to validate generative models for scenarios and planning with regard to plausibility and quality of the generated output. Similarly to the previous chapter, plausibility refers to the degree of reflecting the real world of the synthesized content, or in other words, it can be understood as the absence of inaccurate, biased or hallucinated synthesized inferences by the generative model. The types of generated outputs considered are either sequences of sensory data, such as video content, or the dynamic behavior of road users generated for prediction and planning, or scenario creation. Therefore, a particular emphasis within this chapter is on measures that reflect plausibility with regard to the consistency of temporal and dynamic aspects of the generated output.

The remainder of this chapter is structured as follows. Section **??** surveys different metrics of the model's output with regard to general metrics, video quality metrics, content, motion and temporal consistency metrics, matching and diversity metrics. Section 7.3 addresses particularly physical fidelity with regard to the physical feasibility of generated trajectory data. In Section 7.4, normative fidelity is introduced that extends the realm of physical fidelity emphasizing the formalization of normative specifications of generated scenarios and trajectories. Finally, section 7.5 addresses the inference performance of the machine learning models.

7.2 Metrics

7.2.1 Metrics in general

Evaluating generative models is a crucial aspect of understanding their performance and utility. Metrics provide a quantitative basis for comparing different models and assessing their strengths and weaknesses. In general, metrics for generative models can be categorized into several types, each focusing on different aspects of the model's output and behavior.

- Quality Metrics: These metrics assess the quality of the generated data. Common quality metrics include:
  - Inception Score (IS): Measures the diversity and quality of generated images by using a pre-trained classifier to evaluate the generated samples.
  - Fréchet Inception Distance (FID): Compares the distribution of generated images to real images, providing a measure of how similar the generated images are to the real ones.
  - Perplexity: Often used in language models, it measures how well a probability distribution predicts a sample.
- Diversity Metrics: These metrics evaluate the variety within the generated data. High diversity indicates that the model can produce a wide range of outputs.
  - Mode Score: An extension of the Inception Score that also considers the diversity of the generated samples.
  - Coverage: Measures the extent to which the generated samples cover the distribution of the real data.
- Consistency Metrics: These metrics ensure that the generated data is consistent

with the input or the conditions provided to the model.

  - BLEU Score: Commonly used in machine translation, it measures how closely the generated text matches a set of reference texts.
  - ROUGE Score: Used for evaluating text summarization, it compares the overlap of n-grams between the generated summary and reference summaries.

- Human Evaluation: Despite the availability of quantitative metrics, human evaluation remains a critical component. Human judges assess the quality, coherence, and relevance of the generated data, providing insights that automated metrics might miss.
- Adversarial Robustness: This metric evaluates how well the generative model can withstand adversarial attacks. It is crucial for applications where security and reliability are paramount.
- Computational Efficiency: This metric considers the resources required to train and generate data from the model. It includes factors like training time, memory usage, and inference speed.

By employing a combination of these metrics, researchers and practitioners can gain a comprehensive understanding of a generative model's performance. Each metric provides unique insights, and together they help in identifying the strengths and limitations of different models, guiding improvements and innovations in the field.

7.2.2 Video Quality Assessment

Fréchet Video Distance (FVD) Using the Fréchet Inception Distance, the distributions of real world samples and generated samples are compared in terms of similarity. For the application to images, a multivariate Gaussian distribution is (and can only be) considered, since a closed form solution exists (see (7)). For applying (7) to generated images, Heusel et al. [64] utilize Inception networks by receiving responses of both generated and real world samples in hidden layers. Using a multivariate Gaussian distribution on these responses provide the required co-variances and means for calculating (7). Considering generated videos, the visual coherence of visual content over time, i.e., sequence of images and frames is relevant. Therefore, Unterthiner et al. [153] utilizes Inflation 3D Convnets (I3D) in order to receive feature representations of sequential data and hence establish the metric called Fréchet Video Distance (FVD).

Kernel Video Distance (KVD) Given the fact, that FVD is only reasonable useful if a multivariate Gaussian distribution can be fitted well to the extracted feature representation of I3D networks, Unterthiner et al. [153] propose the Kernel Video Distance (KVD) based on the Maximum Mean Discrepancy (MMD) (see (8)) and a polynomial kernel (see (9) according to [24]. Following, Kernel Video Distance (KVD) [153] is calculated results by the application of MMD to the feature representations received by I3D networks (similar to FVD approach).

Video Inception Score (VIS) Similar to Fréchet Video Distance (FVD), there is an adoption of Inception Score (IS) (see (5)) for sequential data. Saito et al. [133] are utilizing Conv-3D networks to extract representative features. A comparison of FVD and VIS can be found in [133].

7.2.3 Content and Semantic Consistency

Liu et al. [98] address the evaluation of text-to-video models with the EvalCrafter framework, but the following metrics for content consistency can also be applied to ground-truth data or training data sets instead of given text prompts.

Detection Score([98]) The proposed Detection Score (S_{Det}) analyses the average object presence across videos, using (15).

$$S_{Det} = \frac{1}{M_1} \sum_{i=1}^{M_1} \left(\frac{1}{K} \sum_{k=1}^{K} \sigma_{t_k}^i \right) \qquad (15)$$

[...] M_1 is the number of prompts with objects, K is the number of frames where detection is performed, and $\sigma^i_{t_k}$ is the detection result for frame t_k in video i (1 if an object is detected, 0 otherwise)" [98].

Here, the number of prompts with objects can be exchanged with the number of objects in ground truth or training data.

Count Score([98]) The proposed Count Score (S_{Count}) analyses the average object count difference, using (16).

$$S_{Count} = \frac{1}{M_2} \sum_{i=1}^{M_2} \left(1 - \frac{1}{K} \sum_{k=1}^{K} \frac{|c^i_{t_k} - \hat{c}^i|}{\hat{c}^i} \right) \tag{16}$$

"*[...] M_2 is the number of prompts with object counts, $c^i_{t_k}$ is the detected object count at frame t_k in video i, and $\hat{c}^i$ is the ground truth object count for video i.*" [98].

Here, the number of prompts with objects counts can be exchanged with the number of objects counts in ground truth or training data.

Color Score([98]) The proposed Color Score (S_{Color}) analyses average color accuracy, using (17).

$$S_{Color} = \frac{1}{M_3} \sum_{i=1}^{M_3} \left(\frac{1}{K} \sum_{k=1}^{K} s^i_{t_k} \right) \tag{17}$$

"*[...] M_3 is the number of prompts with object colors and $s^i_{t_k}$ is the color accuracy result for frame t_k in video i (1 if the detected color matches the ground truth color, 0 otherwise).*" [98].

Here, the number of prompts with object colors can be exchanged with the number of object colors in ground truth or training data.

CLIP-Temp Semantic Consistency in general is the concept of having a consistent interpretation of semantic content of an object in a sequence of frames. In context of CLIP-Temp [49], it refers to the ability of a system or model to maintain the semantic content of an object across a sequence of frames in a video. Specifically, the semantic consistency is created from the cosine similarity of consecutive frames of the generated videos and then taking the average of all frames [49, 118].

7.2.4 Motion

Average Flow Score Average Flow Score (AFS) describes the motion realism of a video, based on motion information. The average flow of the individual frames of a video is calculated. The result is derived from various metrics, such as the optical flow of a video and error metrics, such as warping error.[98]

In order to calculate the AFS, the errors are accumulated over all frames by calculating the optical flow and warping errors, and then averaged over the number of frames in the video. Thereby, optical flow is a vector representation that shows the movement of an object element over the course of frames[148]. Here, RAFT[148] is a well-known framework for determining optimal flow by using multiple neural networks.

7.2.5 Temporal Consistency

Warping Error Warping error is an evaluation metric used to check for temporal inconsistency. It calculates the pixel-wise differences between warped and predicted images by mapping one frame to another based on optical flow calculations[88]. In detail, it quantifies the discrepancy between the predicted motion and the current frame[150].

With (18) [88], the consistency between two functions is evaluated by measuring the average discrepancy in their performance when applied to the same inputs.

$$E_{pair}(O_t, O_s) = \frac{1}{N} \sum_{i=1}^{N} M_{t,s}(x_i) || O_t(\dot{x}_i) - W(O_s)(x_i) ||_1 \tag{18}$$

W being the backward warping with optical flow [141], N the number of pixels, and occlusion map [129] $M_{t,s}$ for a pair of images O_t and O_s [88]. Occlusion map [129] in this context is used to ensure consistency of style transfer across multiple frames.

Followed by (19) [88], the average consistency of a series of functions can be calculated, resulting in the so-called warping error.

$$E_{warp}(\{O_t\}_{t=1}^{T}) = \frac{1}{T-1}\sum_{t=2}^{T}\{E_{pair}(O_t, O_1) + E_{pair}(O_t, O_{t-1}\}) \quad (19)$$

7.2.6 Diversity (different input should yield different output)

Measuring diversity plays a crucial role in many fields of machine learning, extending beyond generative modeling to dataset curation, reinforcement learning, and active learning. Traditional distribution-level metrics, including statistical divergences and heuristic measures such as Inception Scores and Fréchet Inception Distance, implicitly assess both fidelity and diversity of the modelled distribution in reference to the data distribution. This ambiguity has been recognized as problematic, since low fidelity (e.g., due to *mode invention*) or low diversity (e.g., due to *mode collapse*) can result in similar metric values, complicating the comparison of different generative models [139].

Recognizing the need for more nuanced evaluation, researchers have developed a variety of sample-based diversity metrics. The first approach, first introduced by Sajjadi [134] , distangles statistical divergence into precision and recall components. Precision assesses the support of a generated sample in the data distribution, corresponding to fidelity. Meanwhile, recall assesses the support of the data in the modelled distribution, corresponding to diversity. Data and generated samples are usually mapped into an embedding space that is supposed to capture more meaningful distances. Since the support of neither data nor modelled distribution might be available, it is instead estimated via a k-nearest neighbour (kNN) procedure. Multiple improvements have been suggested to make the support estimation more robust to outliers and preserve meaning in higher dimensional embedding spaces, including improved precision/recall [85], α-precision/β-recall [4], density and coverage [105], symmetric precision/recall [78] and probabilistic precision/recall [114].

Different from the previously mentioned approach are the reference-free, information-theoretic metrics that evaluate diversity in terms of pairwise similarity between generated samples. These measures include the Vendi score [53], defined as the exponential of the Shannon entropy of the eigenvalues of the normalized similarity/kernel matrix. A direct generalization adapts this definition to the Rényi entropy of order q [72, 115] which allows adjustment to sample prevalence, tying diversity measure together with memorization measures. These information-theoretic measures have further theoretically attractive properties, as they can be understood as effective number of unique elements (ranging in value between 1 to sample size), are monotonic, permutation-invariant and generalize over partitions. Furthermore, they are differentiable and, thus, could be used to enforce diversity during training.

Lastly, diversity and other properties can be indirectly measured via domain-specific downstream tasks. For example, Ravuri [120] proposed a classification-based downstream evaluation for image generation, using ground-truth labels as reference.

Despite these advances, many problems remain unsolved. Designing suitable embedding functions, pairwise similarity functions or downstream tasks for the domain can be sensitive to the training data and numerous design choices [86]. Marginalizing out these design choices still requires prohibitive computational costs. In the current state-of-the-art it is generally recommended to use multiple metrics in complement to each other to account for their respective deficiencies and appropriately capture the fidelity-diversity trade-off in current generative modelling approaches.

7.3 Physical Fidelity

In the fields of trajectory prediction and trajectory planning, the evaluation of physical fidelity is given different levels of importance due to the distinct objectives and applications of each process. The planning algorithm ensures physical feasibility and emphasizes optimal paths within physical constraints to achieve smooth and efficient navigation. On the other side, prediction focuses on ac-

curately modeling real-world behavior to ensure safety and reliability without considering physical fidelity as a key factor. (one figure here for better understanding). While planning naturally optimizes the physical feasibility, we will mainly discuss the prediction domain.

Multiple prior works [39, 57, 154] evaluate the percentage of physically feasible predicted or generated trajectories using specific feasibility metrics:

Curvature Violation: The curvature over a trajectory segment is given by

$$\kappa = \frac{|\Delta\theta|}{\|\Delta\mathbf{p}\|}$$

where $|\Delta\theta|$ denotes the absolute change of object heading and $\|\Delta\mathbf{p}\|$ denotes the Euclidean distance between two waypoints. A trajectory contains a curvature violation if $\kappa > 0.3\text{m}^{-1}$ at any point along the trajectory.

Turning Radius Infeasibility (TRI): The turning radius equals the reciprocal of curvature $\frac{1}{\kappa}$ and along the predicted trajectories using two approaches: one that uses the predicted yaw output from the model (TRI-h), and the other that doesn't require yaw predictions and instead uses the circumradius constituting three consecutive waypoints (TRI-c). If the radius is less than a certain threshold τ, it is treated as a violation. The threshold as the approximate minimum turning radius threshold for a midsize sedan, $\tau = 3.5\text{m}$.

Lateral Speed Violation: Decomposing the speed vector v_t at t^{th} timestep into its lateral and longitudinal components, a trajectory contains a lateral speed violation if at any timestep t, the instantaneous lateral speed is greater than 1m/s.

Centripetal Acceleration Violation: Decomposing the acceleration vector a_t into its traversal and centripetal components, a trajectory contains a centripetal acceleration violation if at any timestep t, the instantaneous centripetal acceleration is greater than 10m/s^2.

Traversal Acceleration Violation: Decomposing the acceleration vector a_t into its traversal and centripetal components, a trajectory contains a traversal acceleration violation if at any timestep t, the instantaneous traversal acceleration is smaller than -12m/s^2 or greater than 8m/s^2.

7.4 Normative Fidelity

In order to successfully achieve system acceptance, trust and the shift in responsibility from humans to AI driving functions when driving, in addition to the typical quality metrics presented so far, normative properties also need to be considered. For example, a physically plausible trajectory does not necessarily correspond to social norms. Furthermore, this also includes prevailing traffic rules, system requirements, societal norms or expert knowledge, such as knowledge about physiological thresholds in humans, which can impair system acceptance if they are exceeded. Consequently, it is important that such properties are checked when generating trajectories or scenarios.

Normative properties are often not formally available and must first be formalized for an automatic assessment, e.g. runtime monitoring. Formal languages that can express spatio-temporal properties are predestined for such formalization. These properties enable exactly the description of normative properties, for example the relations between road users and also road users and their environment over time.

Two strands have been established in the literature for the specification of normative properties. The most established strand is temporal logic with extensions for spatial properties. Based on the basic languages Linear Temporal Logic (LTL) and Signal Temporal Logic (STL), derivatives such as STSL [95], SpaTeL [61] and SSTL [107] have been developed for the specification of spatio-temporal properties. Using existing syntheses for runtime monitors, the languages are already established for applications of program verification and verification of cyber-physical systems [18]. STL derivatives also allow investigating the robustness properties of systems in relation to specifications. For example, the extent to which a generated trajectory or a generated scenario deviates from the specification can be investigated. In [170], Zapridou et al. present a runtime monitoring system based on STL-based requirements for an au-

tonomous vehicle's Adaptive Cruise Control. Their goal was to monitor the robustness of PID controller requirements, which are invariant over speed and distance, within the vehicle.

The second strand consists of graphical specification languages. Goyal et al. [58] use a 3x3 grid editor to specify abstract scenarios. Their developed VIVAS framework enables the specification and monitoring of LTL-based abstract test scenarios, focusing on discrete-time system requirements. Klischat et al. [82] use Lanelets to visually specify traffic scenarios based on a formal specification and a map. They convert logical predicates into mixed-integer logic, formulating a quadratic optimization problem. The solutions to this problem are concrete traffic scenarios, which can be used as test scenarios and training data. Multi-Lane Spatial Logic (MLSL) [136] is another visual specification language. It allows for the specification of system requirements for system controllers in cooperative systems at urban road intersections. Traffic Sequence Charts [40], is a visual yet formal scenario specification language which has been already used for the specification of system requirements [22], physical properties, and normative properties [25]. Recently, also a runtime monitoring for TSC specifications [60] has been developed.

In summary, there are already a number of approaches that may be used to check the plausibility of generated trajectories and scenarios. In the strand of temporal logics such as LTL and STL, there are already important extensions such as syntheses for runtime monitors and the possibility of investigating robustness properties. Graphical specification languages are still behind in these aspects due to their younger existence. However, when formalizing (and validating the formalization) of normative properties, graphical languages can enable the integration of other stakeholders such as psychologists, lawyers or other traffic experts, who usually have no experience with temporal logic formulas.

7.5 Inference Efficiency

The performance of machine learning (ML) inference in autonomous driving is typically evaluated using the metrics latency, throughput, and accuracy [121, 173]. The MLPerf Inference benchmark outlines four scenarios for evaluating the performance of ML models: single-stream, multi-stream, offline, and server [121]. They define the requisite query count and sample size to ensure statistically robust results and measure latency by averaging the time it takes for a model to produce inferences, whereas accuracy denotes the percentage of inputs that are correctly processed.

The **single-stream** scenario measures the performance of a model when it is processing a single input at a time. This is useful for evaluating the performance of models that are used in real-time systems, such as autonomous driving, where it is important to process inputs quickly and accurately. The metric is the 90th-percentile latency of the query stream.

The **multi-stream** scenario measures the performance of a model when it is processing multiple inputs simultaneously. This is useful for evaluating the performance of models that are used in batch processing systems, where it is important to process large numbers of inputs as quickly as possible, e.g. analyzing data from multiple sensors simultaneously. The performance metric is the integer number of streams that the system supports while meeting the QoS requirement.

The **offline** scenario measures the performance of a model when it is processing a large batch of inputs. This is useful for evaluating the performance of models that are used in offline processing systems, where latency is not as important as throughput. The metric for the offline scenario is throughput measured in samples per second and is defined as the number of inputs that the model can process in a given amount of time.

The **server** scenario measures the performance of a model when it is deployed in a server environment, where it may be serving multiple clients simultaneously. This is useful for evaluating the performance of models that are used in cloud-based systems, where it is important to maximize the utilization of server resources. The server scenario's performance metric is the Poisson parameter that indicates the queries-per-second (QPS) achievable while meeting the QoS requirement.

For autonomous driving systems, the single-stream and multi-stream scenarios are the most relevant, as they measure the performance of models in real-time systems where low latency and high accuracy are critical. The offline and server scenarios may be less relevant, as autonomous driving systems typically do not have the luxury of processing inputs in batch or in a server environment. However, the offline scenario may still be useful for evaluating the performance of ML models that are used in offline data processing pipelines, such as those used for training and validating autonomous driving systems. The server scenario may also be relevant for evaluating the performance of ML models that are used in cloud-based autonomous driving systems, such as those used for remote monitoring and control.

8 Plausibility check of generated static road maps and traffic routing

8.1 Metrics

As for other data types, plausibility of generated static environment data can be determined on sample level and on distribution level (see section 1.2.1).

8.1.1 Sample Features

Consideration of static environment features on sample level is relevant for two objectives. On the one hand, it allows to quantify how well generated samples lie within the boundary conditions and thus comply with certain general standards, which may serve as an initial filter of low-quality samples. On the other hand, selection and comparison of suitable sample features are prerequisites for meaningful distribution level metrics (see section 8.1.2) by providing a distance metric in feature space.

One promising approach of environment and road network data management is usage of (attributed) graphs. To calculate graph distances, generic graph features as well as urban planning specific features are used. Chen et al. and Li et al. e.g. focus on degrees, clustering coefficients and orbit counts [35, 92]. Chu et al. consider e.g. node degrees, the node density or accessible road length in a variable neighbourhood or a transportation convenience coefficient which is calculated based on euclidean and shortest path distance of node pairs [38]. In addition to that, Li et al. use correlations of graph structure and node labels [92]. Further examples of urban planning features are the number, size and aspect ratio of blocks, total and average street length and density, redundancy, metric reach, connected road ratio and link-node-ratio [5, 45, 63, 149, 168].

Road networks can be enriched by adding 3D assets which are represented by e.g. three-dimensional point clouds. As point clouds are themselves a distribution in space, to determine their distance, distribution comparison methods (see section 6.1) are applicable on the sample level as well. Regularly used is especially the Earth Mover's Distance (EMD) which is a Wasserstein metric calculating the optimal solution to the transport problem of distribution transformation. In addition to this, neighbour-based approaches rely on e.g. the Chamfer Distance (CD) which calculates the squared nearest neighbour point distance [2].

8.1.2 Distribution Comparison

The sample features mentioned in section 8.1.1 can be used to compare feature distributions over a generated and a reference sample set. Consequently, the divergence-based distribution comparison methods explained in section 6.1 are used for static environment features as well and are not covered in detail here.

3D environment generators are e.g. often validated via FID or KID (see section 6.2) [41, 146, 163].

For point cloud distribution comparison, the Jensen-Shannon Divergence (JSD), a symmetrical KL divergence, coverage (see section 7.2.1) and MMD (see eq. 8) are proposed [167]. A preferred alternative is the 1-Nearest-Neighbour Accuracy (1-NN) which calculates the fraction of samples having a nearest neighbour of their own type (generated or reference) in feature space [167, 178]. In case of well-mixed distributions, it converges to 50 % due

to equal probability of the next neighbour's type.

8.1.3 Plausibility Check of HD Map

Prior works have concentrated on generating static road maps for vehicles using model-based methods [20, 21, 94]. Becker et al. hierarchically modeled the road network, incorporating roundabouts, junctions, and connecting roads, detailed with road and intersection definitions [21]. Li et al. employed procedural generation techniques to create road networks featuring diverse map elements such as straight roads, ramps, forks, roundabouts, curves, T-intersections, and intersections [94]. Both approaches ensure the geometric plausibility of the generated road maps by selecting realistic parameters for map elements, such as road curvature and lane width.

Topology plausibility is ensured through deliberate design choices. For instance, curvature continuity is maintained by matching the curvatures of segments at their connecting points [21].

Additionally, Bahari et al. modify real-world maps from Argoverse Motion Dataset [34], validating these modifications with a Model Predictive Control (MPC) framework utilizing a bicycle dynamic model [13].

While recent studies focus on the plausibility of road geometry and topology, the plausibility of generated traffic signs and traffic lights remains under-researched. Addressing this gap is crucial for the development of comprehensive and realistic vehicular simulations.

9 Future Directions and Trends

The scalability of generative AI in the automotive industry is poised to evolve with several emerging trends and technologies. These future trends aim to address current limitations, including cost, computational efficiency, and architectural optimization, thereby enhancing the deployment of AI in vehicles.

Specialized Hardware

While the use of specialized hardware like deep learning accelerators can significantly enhance training speeds and improve inference times, it comes with high costs and inherent memory limitations. These accelerators, such as GPUs and TPUs, offer superior performance but are more expensive than traditional CPU units. Future advancements in specialized hardware are expected to focus on reducing costs and enhancing memory capabilities to make these technologies more accessible and efficient for automotive applications.

Standardized framework Different AI model architectures—such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers—each have unique structures and patterns of information flow, making it essential to tailor plausibility checks to these architectures. A standardized framework would ideally establish adaptable protocols for measuring plausibility across various models, potentially involving benchmarks, diagnostic tools, and interpretability tests that address each architecture's strengths and limitations. Such frameworks would enable consistent and transparent evaluations, ultimately helping researchers and practitioners ensure that compressed models remain trustworthy and perform well in diverse, real-world scenarios.

Efficient Training and Inference Techniques

Future trends will also emphasize developing more efficient training and inference techniques. This includes methods such as adaptive Inference where Dynamic models that can adjust their complexity based on the input can improve efficiency. For example, a model could use simpler pathways for less complex inputs and more complex pathways for challenging inputs.

Integration of AI with Other Technologies

Integrating AI with emerging technologies such as 5G, blockchain, and advanced sensor systems will further enhance the capabilities of generative AI in vehicles. For example, 5G connectivity can provide the necessary bandwidth for real-time data processing, while

blockchain can ensure secure and transparent data management.

Sustainable AI

As the environmental impact of AI becomes a growing concern, there will be a push towards developing more sustainable AI solutions. This includes optimizing energy consumption during both training and inference phases and exploring renewable energy sources for powering AI systems.

Enhanced Data Privacy and Security

Future advancements will also focus on improving data privacy and security. Techniques such as federated learning, where models are trained across decentralized devices without sharing raw data, can help protect sensitive information [164]. Ensuring robust data protection measures will be critical as AI systems continue to handle more personal and vehicle data.

By leveraging specialized hardware, optimizing neural architectures, enhancing training and inference techniques, integrating with other technologies, promoting sustainability, and ensuring data privacy, the industry can achieve scalable and efficient AI deployment. These trends will play a crucial role in driving the next wave of innovation in automotive technology

10 Conclusion

The automotive industry is at the early stages of adopting generative AI, driven by advancements in E/E architectures designed to handle the significant computational demands of such models. Currently, generative AI is primarily applied in the early stages of product development, but scaling its use within vehicles presents a unique set of challenges. Overcoming hardware limitations is critical to ensuring system safety and optimal user interactions in these settings.

Techniques like quantization provide essential, non-intrusive solutions for enabling neural network inference on lower bit-width arithmetic, crucial for resource-constrained automotive hardware. These methods significantly improve inference speed, reduce memory usage, and decrease model size while addressing challenges such as mitigating performance degradation, managing dynamic ranges, and optimizing hardware-specific constraints. Similarly, pruning techniques, whether with or without retraining, offer efficient alternatives that reduce computational load while maintaining or improving model performance. Non-retraining approaches benefit resource-limited environments, while retraining-based methods enable more refined optimizations with added computational effort. Additionally, low-rank tensor decomposition offers a powerful framework for reducing computational complexity while maintaining model scalability, particularly in high-dimensional data settings. Future research should focus on optimizing these strategies for large-scale applications.

Applying plausibility checks to compressed AI models—such as those using pruning, quantization, and tensor compression—is critical for ensuring model reliability and interpretability. Existing literature highlights that while compression techniques significantly reduce computational demands and enable deployment on resource-constrained devices, they can impact the model's plausibility by altering the integrity of learned representations. By incorporating rigorous plausibility assessments, researchers and practitioners can verify that compressed models maintain alignment with human intuition and domain knowledge, which is essential for their safe and effective application in real-world settings.

By integrating the techniques studied in this paper for model compression and plausibility checks, the automotive industry has the potential to effectively scale generative AI both within vehicles and across broader applications and ensure the reliability of the scaled models. While these methods show promising results toward driving innovation, enhancing safety, and achieving cost-effectiveness, further exploration and refinement are needed to fully realize their benefits and expand AI integration throughout the automotive ecosystem.

Acknowlegements

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Climate Action within the project "nxtAIM - Next Generation AI Methods". The authors would like to thank the consortium for the successful cooperation.

References

[1] Salah Abdelhamid, Hossam S. Hassanein, and Gary Takahara. "Vehicle as a mobile sensor". In: *Procedia Computer Science*. Vol. 34. 2014, pp. 286–295 (cit. on p. 6).

[2] Panos Achlioptas et al. *Learning Representations and Generative Models for 3D Point Clouds*. June 12, 2018. DOI: 10.48550/arXiv.1707.02392. arXiv: 1707.02392 [cs]. URL: http://arxiv.org/abs/1707.02392. Prepublished (cit. on pp. 28, 35).

[3] Melissa Ailem et al. "Examining the robustness of LLM evaluation to the distributional assumptions of benchmarks". In: *arXiv preprint arXiv:2404.16966* (2024) (cit. on p. 6).

[4] Ahmed M. Alaa et al. "How Faithful is your Synthetic Data? Sample-level Metrics for Evaluating and Auditing Generative Models". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. PMLR. Copyright 2022 by the author(s). Baltimore, Maryland, USA: PMLR, 2022 (cit. on pp. 3, 32).

[5] Sawsan AlHalawani et al. "What makes London work like London?" In: *Computer Graphics Forum*. Vol. 33. 5. Wiley Online Library. 2014, pp. 157–165 (cit. on p. 35).

[6] Yongqi An et al. *Fluctuation-based Adaptive Structured Pruning for Large Language Models*. 2023. arXiv: 2312.11983 (cit. on p. 13).

[7] V. Antinyan. "Revealing the complexity of automotive software". In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Nov. 2020, pp. 1525–1528 (cit. on pp. 8, 10).

[8] Nicholas Arechiga et al. "Drag-guided diffusion models for vehicle image generation". In: *arXiv preprint arXiv:2306.09935* (2023) (cit. on p. 4).

[9] H. Askaripoor, M. Hashemi Farzaneh, and A. Knoll. "E/E architecture synthesis: Challenges and technologies". In: *Electronics* 11.4 (2022), p. 518 (cit. on p. 8).

[10] Audi AG. *Reinventing the Wheel? "FelGAN" Inspires New Rim Designs with AI*. https://www.audi-mediacenter.com/en/press-releases/reinventing-the-wheel-felgan-inspires-new-rim-designs-with-ai-15097. Accessed: 2024-08-29. 2022 (cit. on p. 4).

[11] N. Ayres, L. Deka, and B. Passow. "Virtualisation as a means for dynamic software update within the automotive E/E architecture". In: *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, Aug. 2019, pp. 154–157 (cit. on pp. 8–10).

[12] Hicham Badri and Appu Shaji. *Half-Quadratic Quantization of Large Machine Learning Models*. Nov. 2023. URL: https://mobiusml.github.io/hqq_blog/ (cit. on p. 18).

[13] Mohammadhossein Bahari et al. "Vehicle trajectory prediction works, but not everywhere". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17123–17133 (cit. on p. 36).

[14] Ching-Yuan Bai et al. “On training sample memorization: Lessons from benchmarking generative modeling with a large-scale competition”. In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 2534–2542 (cit. on p. 27).

[15] Simone Balloccu et al. “Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms”. In: *arXiv preprint arXiv:2402.03927* (2024) (cit. on p. 6).

[16] V. Bandur et al. “Making the case for centralized automotive E/E architectures”. In: *IEEE Transactions on Vehicular Technology* 70.2 (2021), pp. 1230–1245 (cit. on pp. 8–10).

[17] Shane Barratt and Rishi Sharma. “A Note on the Inception Score”. In: (Jan. 2018) (cit. on p. 4).

[18] Ezio Bartocci et al. “Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications”. In: *Lectures on Runtime Verification*. 2018. DOI: 10.1007/978-3-319-75632-5_5 (cit. on p. 33).

[19] Gaurav Batra. *Artificial-intelligence hardware: New opportunities for semiconductor companies*. https://www.mckinsey.com/industries/semiconductors/our-insights/artificial-intelligence-hardware-new-opportunities-for-semiconductor-companies/. Jan. 2019 (cit. on p. 5).

[20] Daniel Becker, Christian Geller, and Lutz Eckstein. “Road network variation based on hd map analysis for the simulative safety assurance of automated vehicles”. In: *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. IEEE. 2022, pp. 1–6 (cit. on p. 36).

[21] Daniel Becker et al. “Generation of complex road networks using a simplified logical description for the validation of automated vehicles”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–7 (cit. on p. 36).

[22] Jan Steffen Becker. “Partial Consistency for Requirement Engineering with Traffic Sequence Charts”. In: *Combined Proceedings of the Workshops at Software Engineering 2020 Co-located with the German Software Engineering Conference 2020 (SE 2020), Innsbruck, Österreich, March 05, 2020*. Vol. 2581. CEUR Workshop Proceedings. http://ceur-ws.org/Vol-2581/ase2020paper1.pdf. 2020 (cit. on p. 34).

[23] Yash Bhalgat et al. “LSQ+: Improving low-bit quantization through learnable offsets and better initialization”. In: *CoRR* abs/2004.09576 (2020). arXiv: 2004.09576. URL: https://arxiv.org/abs/2004.09576 (cit. on p. 17).

[24] Mikołaj Bińkowski et al. “Demystifying MMD GANs”. In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=r1lUOzWCW (cit. on pp. 27, 30).

[25] Philipp Borchers et al. “Using Traffic Sequence Charts for Knowledge Formalization and AI-Application”. In: *Lecture Notes in Networks and Systems*. to be published. Springer, p. 23 (cit. on p. 34).

[26] Jie Bu et al. “Learning Compact Representations of Neural Networks using DiscriminAtive Masking”. In: Oct. 2021 (cit. on p. 23).

[27] Lukas Budach et al. *The Effects of Data Quality on Machine Learning Performance*. 2022. arXiv: 2207.14529 [cs.DB]. URL: https://arxiv.org/abs/2207.14529 (cit. on p. 7).

[28] Sven Burdorf, Karoline Plum, and Daniel Hasenklever. “Reducing the amount of real world data for object detector training with synthetic data”. In: *arXiv preprint arXiv:2202.00632* (2022) (cit. on p. 28).

[29] O. Burkacky, J. Deichmann, and C. Lai. *Advanced semiconductors for the era of centralized E/E architectures*. https://www.mckinsey.com/industries/semiconductors/our-insights/advanced-semiconductors-for-the-era-of-centralized-e-e-architectures. June 2024 (cit. on p. 8).

[30] Qiao Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. "PuMer: Pruning and merging tokens for efficient vision language models". In: *arXiv preprint arXiv:2305.17530* (2023) (cit. on p. 15).

[31] Xingwei Cao, Guillaume Rabusseau, and Joelle Pineau. "Tensor Regression Networks with various Low-Rank Tensor Approximations". In: (Dec. 2017) (cit. on p. 23).

[32] V. Cebotari and S. Kugele. "On the nature of automotive service architectures". In: *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, Mar. 2019, pp. 53–60 (cit. on pp. 9, 11).

[33] Huiwen Chang et al. "Muse: Text-to-image generation via masked generative transformers". In: *arXiv preprint arXiv:2301.00704* (2023) (cit. on p. 28).

[34] Ming-Fang Chang et al. "Argoverse: 3d tracking and forecasting with rich maps". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8748–8757 (cit. on p. 36).

[35] Xiaohui Chen et al. *Efficient and Degree-Guided Graph Generation via Discrete Diffusion Modeling*. May 31, 2023. DOI: 10.48550/arXiv.2305.04111. arXiv: 2305.04111 [cs]. URL: http://arxiv.org/abs/2305.04111. Pre-published (cit. on p. 35).

[36] Wei-Lin Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. Mar. 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/ (cit. on p. 14).

[37] Lénaïc Chizat et al. "Faster Wasserstein Distance Estimation with the Sinkhorn Divergence". In: *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. Vancouver, Canada, 2020 (cit. on p. 26).

[38] Hang Chu et al. *Neural Turtle Graphics for Modeling City Road Layouts*. Oct. 4, 2019. DOI: 10.48550/arXiv.1910.02055. arXiv: 1910.02055 [cs]. URL: http://arxiv.org/abs/1910.02055. Pre-published (cit. on p. 35).

[39] Henggang Cui et al. "Deep kinematic models for kinematically feasible vehicle trajectory predictions". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 10563–10569 (cit. on p. 33).

[40] Werner Damm et al. "A Formal Semantics for Traffic Sequence Charts". In: *Principles of Modeling: Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*. Ed. by Marten Lohstroh, Patricia Derler, and Marjan Sirjani. Cham: Springer International Publishing, 2018, pp. 182–205. ISBN: 978-3-319-95246-8. DOI: 10.1007/978-3-319-95246-8_11 (cit. on p. 34).

[41] Jie Deng et al. *CityGen: Infinite and Controllable 3D City Layout Generation*. Dec. 3, 2023. DOI: 10.48550/arXiv.2312.01508. arXiv: 2312.01508 [cs]. URL: http://arxiv.org/abs/2312.01508. Pre-published (cit. on p. 35).

[42] Emily Denton et al. "Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation". In: 2 (Apr. 2014) (cit. on p. 21).

[43] Lucio Dery et al. *Everybody Prune Now: Structured Pruning of LLMs with only Forward Passes*. 2024. DOI: https://doi.org/10.48550/arXiv.2402.05406. arXiv: 2402.05406 [cs.LG] (cit. on p. 13).

[44] Tim Dettmers et al. "LLM. int8(): 8-bit matrix multiplication for transformers at scale". In: *arXiv preprint*

arXiv:2208.07339 (2022) (cit. on p. 17).

[45] Jennifer Dill. "Measuring network connectivity for bicycling and walking". In: *83rd annual meeting of the Transportation Research Board, Washington, DC*. 2004, pp. 11–15 (cit. on p. 35).

[46] R. Ding and A. Mullhaupt. "Empirical Squared Hellinger Distance Estimator and Generalizations to a Family of α-Divergence Estimators". In: *Entropy* 25 (2023). DOI: 10.3390/e25040612. URL: https://doi.org/10.3390/e25040612 (cit. on p. 26).

[47] Patrick Ebel. "Generative AI and Attentive User Interfaces: Five Strategies to Enhance Take-Over Quality in Automated Driving". In: *arXiv preprint arXiv:2402.10664* (2024) (cit. on p. 5).

[48] Vage Egiazarian et al. *Extreme Compression of Large Language Models via Additive Quantization*. 2024. arXiv: 2401.06118 [cs.LG] (cit. on p. 18).

[49] Patrick Esser et al. "Structure and Content-Guided Video Synthesis with Diffusion Models". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), pp. 7312–7322. URL: https://api.semanticscholar.org/CorpusID:256615582 (cit. on p. 31).

[50] Steven K. Esser et al. "Learned Step Size Quantization". In: *CoRR* abs/1902.08153 (2019). arXiv: 1902.08153. URL: http://arxiv.org/abs/1902.08153 (cit. on p. 17).

[51] Lijie Fan et al. "Scaling laws of synthetic images for model training... for now". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 7382–7392 (cit. on p. 28).

[52] Karl Freund. *Google Cloud Doubles Down On NVIDIA GPUs For Inference*. https://www.forbes.com/sites/moorinsights/2019/05/09/google-cloud-doubles-down-on-nvidia-gpus-for-inference/. May 2019 (cit. on p. 5).

[53] Dan Friedman and Adji Bousso Dieng. "The Vendi Score: A Diversity Evaluation Metric for Machine Learning". In: *arXiv preprint arXiv:2210.02410* (2022) (cit. on pp. 3, 32).

[54] Ruiyuan Gao et al. "Magicdrive: Street view generation with diverse 3d geometry control". In: *arXiv preprint arXiv:2310.02601* (2023) (cit. on p. 28).

[55] Durga Prasad Garikapati and Samir S. Shetiya. "Autonomous Vehicles: Evolution of Artificial Intelligence and the Current Industry Landscape". In: *Big Data and Cognitive Computing* 8.4 (2024), p. 42 (cit. on p. 5).

[56] Sahar Gholami. "Can pruning make large language models more efficient?" In: *Redefining Security With Cyber AI*. IGI Global, 2024, pp. 1–14 (cit. on p. 15).

[57] Harshayu Girase et al. "Physically feasible vehicle trajectory prediction". In: *arXiv preprint arXiv:2104.14679* (2021) (cit. on p. 33).

[58] Srajan Goyal et al. "Automatic Generation of Scenarios for System-level Simulation-based Verification of Autonomous Driving Systems". In: *Electronic Proceedings in Theoretical Computer Science* 395 (Nov. 2023), pp. 113–129. ISSN: 2075-2180. DOI: 10.4204/eptcs.395.8 (cit. on p. 34).

[59] Arthur Gretton et al. "A kernel two-sample test". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 723–773 (cit. on p. 27).

[60] Dominik Grundt et al. "Towards Runtime Monitoring of Complex System Requirements for Autonomous Driving Functions". In: Proceedings Fourth International Workshop on *Formal Methods for Autonomous Systems (FMAS)* and Fourth International Workshop on *Automated and verifiable Software sYstem DEvelopment (ASYDE)*, Berlin, Germany, 26th and 27th of September 2022. Ed. by Matt Luckcuck and Marie Farrell. Vol. 371. Electronic Proceedings in Theoretical Computer Science.

Open Publishing Association, 2022, pp. 53–61. DOI: 10.4204/EPTCS.371.4 (cit. on p. 34).

[61] Iman Haghighi et al. "SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems". In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. HSCC '15. Association for Computing Machinery, 2015, pp. 189–198. DOI: 10.1145/2728606.2728633 (cit. on p. 33).

[62] Jiyeon Han et al. "Rarity Score: A New Metric to Evaluate the Uncommonness of Synthesized Images". In: *Published as a conference paper at ICLR 2023*. {j.han, hwanil.choi, jaesik.choi}@kaist.ac.kr, {yunjey.choi, jhkim.ai, jungwoo.ha}@navercorp.com. Kim Jaechul Graduate School of AI, KAIST and NAVER AI Lab. 2023 (cit. on p. 3).

[63] Stefan Hartmann et al. "StreetGAN: Towards Road Network Synthesis with Generative Adversarial Networks". In: *Computer Science Research Notes* (2017) (cit. on p. 35).

[64] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf (cit. on p. 30).

[65] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 27).

[66] Andrew Hu et al. "Gaia-1: A generative world model for autonomous driving". In: *arXiv preprint arXiv:2309.17080* (2023) (cit. on p. 4).

[67] Anthony Hu et al. "Gaia-1: A generative world model for autonomous driving". In: *arXiv preprint arXiv:2309.17080* (2023) (cit. on p. 28).

[68] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685. URL: https://arxiv.org/abs/2106.09685 (cit. on p. 13).

[69] Hümeyra Hüsem and Zerrin Orman. "A survey on image super-resolution with generative adversarial networks". In: *Acta Infologica* 4.2 (2020), pp. 139–154 (cit. on p. 4).

[70] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán. "Low-Rank Compression of Neural Nets: Learning the Rank of Each Layer". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 8046–8056. DOI: 10.1109/CVPR42600.2020.00807 (cit. on p. 23).

[71] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. "Speeding up Convolutional Neural Networks with Low Rank Expansions". In: *CoRR* abs/1405.3866 (2014). arXiv: 1405.3866. URL: http://arxiv.org/abs/1405.3866 (cit. on p. 20).

[72] Mohammad Jalali, Cheuk Ting Li, and Farzan Farnia. "An Information-Theoretic Evaluation of Generative Models in Learning Multi-modal Distributions". In: *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*. Department of Electrical and Computer Engineering, Isfahan University of Technology, Department of Information Engineering, The Chinese University of Hong Kong, Department of Computer Science and Engineering, The Chinese University of Hong Kong. NeurIPS. The Chinese University of Hong Kong and Isfahan University of Technology, 2023. URL: https://neurips.cc/ (cit. on pp. 3, 32).

[73] Ziwei Ji et al. “Survey of hallucination in natural language generation”. In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38 (cit. on p. 7).

[74] Siddhartha Rao Kamalakara et al. “Exploring Low Rank Training of Deep Neural Networks”. In: *ArXiv* abs/2209.13569 (2022). URL: https://api.semanticscholar.org/CorpusID:252545358 (cit. on p. 23).

[75] Amina Kammoun et al. “Generative Adversarial Networks for Face Generation: A Survey”. In: *ACM Comput. Surv.* 55.5 (Dec. 2022). ISSN: 0360-0300. DOI: 10.1145/3527850. URL: https://doi.org/10.1145/3527850 (cit. on p. 4).

[76] Nidhi Katiyar et al. “AI in Autonomous Vehicles: Opportunities, Challenges, and Regulatory Implications”. In: *Educational Administration: Theory and Practice* 30.4 (2024), pp. 6255–6264 (cit. on pp. 5, 6).

[77] Daniel Martin Katz et al. “Gpt-4 passes the bar exam”. In: *Philosophical Transactions of the Royal Society A* 382.2270 (2024), p. 20230254 (cit. on p. 6).

[78] Mahyar Khayatkhoei and Wael Abd-Almageed. “Emergent Asymmetry of Precision and Recall for Measuring Fidelity and Diversity of Generative Models in High Dimensions”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 16326–16343. URL: https://proceedings.mlr.press/v202/khayatkhoei23a.html (cit. on p. 32).

[79] H. Kim, M. Khan, and C. Kyung. “Efficient Neural Network Compression”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2019, pp. 12561–12569. DOI: 10.1109/CVPR.2019.01285. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.01285 (cit. on p. 23).

[80] Bo-Kyeong Kim et al. *Shortened LLaMA: A Simple Depth Pruning for Large Language Models*. 2024. DOI: https://doi.org/10.48550/arXiv.2402.02834. arXiv: 2402.02834 [cs.LG] (cit. on p. 13).

[81] Yong-Deok Kim et al. “Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications”. In: May 2016 (cit. on p. 22).

[82] Moritz Klischat and Matthias Althoff. “Synthesizing Traffic Scenarios from Formal Specifications for Testing Automated Vehicles”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 2065–2072. DOI: 10.1109/IV47402.2020.9304617 (cit. on p. 34).

[83] Jean Kossaifi et al. “Tensor Regression Networks”. In: (July 2017) (cit. on p. 22).

[84] S. Kugele, D. Hettler, and J. Peter. “Data-centric communication and containerization for future automotive software architectures”. In: *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE, Apr. 2018, pp. 65–6509 (cit. on pp. 9, 10).

[85] Tuomas Kynkäänniemi et al. “Improved Precision and Recall Metric for Assessing Generative Models”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 3929–3938. URL: https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html (cit. on p. 32).

[86] Tuomas Kynkäänniemi et al. “The Role of ImageNet Classes in Fréchet Inception Distance”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://

openreview.net/pdf?id=4oXTQ6m%5C_ws8 (cit. on p. 32).

[87] Vadim Lebedev et al. "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition". In: (Dec. 2014) (cit. on p. 21).

[88] Chenyang Lei, Yazhou Xing, and Qifeng Chen. "Blind Video Temporal Consistency via Deep Video Prior". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1083–1093. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/0c0a7566915f4f24853fc4192689aa7e-Paper.pdf (cit. on p. 31).

[89] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474 (cit. on p. 7).

[90] Der-Chiang Li et al. "Detecting representative data and generating synthetic samples to improve learning accuracy with imbalanced data sets". In: *PLOS ONE* 12.8 (Aug. 2017), pp. 1–24. DOI: 10.1371/journal.pone.0181853. URL: https://doi.org/10.1371/journal.pone.0181853 (cit. on p. 8).

[91] Junyi Li et al. "Halueval: A large-scale hallucination evaluation benchmark for large language models". In: *arXiv preprint arXiv:2305.11747* (2023) (cit. on p. 7).

[92] Mufei Li et al. *GraphMaker: Can Diffusion Models Generate Large Attributed Graphs?* Jan. 27, 2024. DOI: 10.48550/arXiv.2310.13833. arXiv: 2310.13833 [cs]. URL: http://arxiv.org/abs/2310.13833. Pre-published (cit. on p. 35).

[93] Qifeng Li et al. "Think2drive: Efficient reinforcement learning by thinking in latent world model for quasi-realistic autonomous driving (in carla-v2)". In: *arXiv preprint arXiv:2402.16720* (2024) (cit. on p. 28).

[94] Quanyi Li et al. "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning". In: *IEEE transactions on pattern analysis and machine intelligence* 45.3 (2022), pp. 3461–3475 (cit. on p. 36).

[95] Tengfei Li et al. "STSL: A Novel Spatio-Temporal Specification Language for Cyber-Physical Systems". In: *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*. 2020, pp. 309–319. DOI: 10.1109/QRS51102.2020.00048 (cit. on p. 33).

[96] Yuhang Li et al. "BRECQ: Pushing the Limit of Post-Training Quantization by Block Reconstruction". In: *CoRR* abs/2102.05426 (2021). arXiv: 2102.05426. URL: https://arxiv.org/abs/2102.05426 (cit. on p. 16).

[97] Ji Lin et al. "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration". In: *MLSys*. 2024 (cit. on p. 19).

[98] Yaofang Liu et al. *EvalCrafter: Benchmarking and Evaluating Large Video Generation Models*. 2024. arXiv: 2310.11440 [cs.CV]. URL: https://arxiv.org/abs/2310.11440 (cit. on pp. 30, 31).

[99] Xinyin Ma, Gongfan Fang, and Xinchao Wang. *LLM-Pruner: On the Structural Pruning of Large Language Models*. 2023. arXiv: 2305.11627 [cs.CL]. URL: https://arxiv.org/abs/2305.11627 (cit. on p. 14).

[100] Aboli Marathe et al. "Wedge: A multi-weather autonomous driving dataset built from generative vision-language models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 3318–3327 (cit. on p. 28).

[101] Anirudh Marathe et al. "Wedge: A multi-weather autonomous driving dataset built from generative vision-language models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 3318–3327 (cit. on p. 4).

[102] Eric Martínez. "Re-evaluating GPT-4's bar exam performance". In: *Artificial Intelligence and Law* (2024), pp. 1–24 (cit. on p. 6).

[103] David McAllester and Karl Stratos. "Formal Limitations on the Measurement of Mutual Information". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 26–28 Aug 2020, pp. 875–884. URL: https://proceedings.mlr.press/v108/mcallester20a.html (cit. on p. 26).

[104] Rachel Minster et al. *CP Decomposition for Tensors via Alternating Least Squares with QR Decomposition*. Dec. 2021 (cit. on p. 21).

[105] Muhammad Ferjad Naeem et al. "Reliable Fidelity and Diversity Metrics for Generative Models". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. PMLR. Copyright 2020 by the author(s). Clova AI Research, Naver Corp., Technische Universität München, Germany, École polytechnique fédérale de Lausanne (EPFL), Switzerland. Online: PMLR, 2020. URL: https://proceedings.mlr.press/v119/ (cit. on pp. 3, 32).

[106] Markus Nagel et al. "A White Paper on Neural Network Quantization". In: *CoRR* abs/2106.08295 (2021). arXiv: 2106.08295. URL: https://arxiv.org/abs/2106.08295 (cit. on p. 16).

[107] Laura Nenzi et al. "Qualitative and Quantitative Monitoring of Spatio-Temporal Properties with SSTL". In: *Logical Methods in Computer Science* 14 (2017). DOI: 10.23638/LMCS-14(4:2)2018 (cit. on p. 33).

[108] Joshua Niemeijer et al. "Generalization by adaptation: Diffusion-based domain extension for domain-generalized semantic segmentation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 2830–2840 (cit. on p. 28).

[109] E. Ntoutsi et al. "Bias in data-driven artificial intelligence systems—an introductory survey". In: *WIREs Data Mining and Knowledge Discovery* 10 (3 2020). DOI: 10.1002/widm.1356 (cit. on p. 7).

[110] *NVIDIA Tesla V100 GPU Accelerator*. 2017. URL: https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf (cit. on p. 12).

[111] R. Obermaisser et al. "From a federated to an integrated automotive architecture". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.7 (2009), pp. 956–965 (cit. on pp. 9–11).

[112] Yonatan Oren et al. "Proving test set contamination in black box language models". In: *arXiv preprint arXiv:2310.17623* (2023) (cit. on p. 6).

[113] Ivan Oseledets. "Tensor-Train Decomposition". In: *SIAM J. Scientific Computing* 33 (Jan. 2011), pp. 2295–2317. DOI: 10.1137/090752286 (cit. on p. 22).

[114] Dogyun Park and Suhyun Kim. "Probabilistic Precision and Recall Towards Reliable Evaluation of Generative Models". In: *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 2023, pp. 20042–20052. DOI: 10.1109/ICCV51070.2023.01839. URL: https://doi.org/10.1109/ICCV51070.2023.01839 (cit. on p. 32).

[115] Amey P. Pasarkar and Adji Bousso Dieng. "Cousins Of The Vendi Score: A Family Of Similarity-Based Diversity Metrics For Science And Machine Learning". In: *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*. Ed. by Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li. Vol. 238. Proceedings of Machine Learning Research. PMLR, 2024, pp. 3808–3816. URL: https://

proceedings.mlr.press/v238/p-pasarkar24a.html (cit. on p. 32).

[116] F. Perez-Cruz. "Kullback-Leibler Divergence Estimation of Continuous Distributions". In: *2008 IEEE International Symposium on Information Theory*. Toronto, ON, Canada, 2008, pp. 1666–1670. DOI: 10.1109/ISIT.2008.4595271 (cit. on p. 26).

[117] Dustin Podell et al. *SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis*. 2023. eprint: 2307.01952 (cit. on p. 14).

[118] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *International Conference on Machine Learning*. 2021. URL: https://api.semanticscholar.org/CorpusID:231591445 (cit. on p. 31).

[119] Tanmay Rajore et al. "TRUCE: Private Benchmarking to Prevent Contamination and Improve Comparative Evaluation of LLMs". In: *arXiv e-prints* (2024), arXiv–2403 (cit. on p. 6).

[120] Suman V. Ravuri and Oriol Vinyals. "Classification Accuracy Score for Conditional Generative Models". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 12247–12258. URL: https://proceedings.neurips.cc/paper/2019/hash/fcf55a303b71b84d326fb1d06e332a26-Abstract.html (cit. on p. 32).

[121] Vijay Janapa Reddi et al. *MLPerf Inference Benchmark*. 2020. arXiv: 1911.02549 [cs.LG]. URL: https://arxiv.org/abs/1911.02549 (cit. on p. 34).

[122] A. Reuther et al. "AI accelerator survey and trends". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sept. 2021, pp. 1–9 (cit. on p. 9).

[123] A. Reuther et al. "AI and ML accelerator survey and trends". In: *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sept. 2022, pp. 1–10 (cit. on p. 9).

[124] A. Reuther et al. "Lincoln AI computing survey (LAICS) update". In: *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sept. 2023, pp. 1–7 (cit. on pp. 9, 10).

[125] A. Reuther et al. "Survey and benchmarking of machine learning accelerators". In: *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sept. 2019, pp. 1–9 (cit. on p. 9).

[126] A. Reuther et al. "Survey of machine learning accelerators". In: *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sept. 2020, pp. 1–12 (cit. on p. 9).

[127] Robin Rombach et al. *High-Resolution Image Synthesis With Latent Diffusion Models*. June 2022 (cit. on p. 14).

[128] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695 (cit. on p. 28).

[129] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. "Artistic Style Transfer for Videos". In: *Pattern Recognition*. Springer International Publishing, 2016, pp. 26–36. ISBN: 9783319458861. DOI: 10.1007/978-3-319-45886-1_3. URL: http://dx.doi.org/10.1007/978-3-319-45886-1_3 (cit. on p. 31).

[130] M. Rumez et al. "An overview of automotive service-oriented architectures and implications for security countermeasures". In: *IEEE Access* 8 (2020), pp. 221852–221870 (cit. on pp. 8–11).

[131] Chitwan Saharia et al. "Photorealistic text-to-image diffusion models with deep language understanding". In: *Advances in neural information process-*

ing systems 35 (2022), pp. 36479–36494 (cit. on p. 28).

[132] S. Saidi et al. "Future automotive systems design: Research challenges and opportunities: Special session". In: *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*. Sept. 2018, pp. 1–7 (cit. on p. 8).

[133] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. *Temporal Generative Adversarial Nets with Singular Value Clipping*. 2017. arXiv: 1611.06624 [cs.LG]. URL: https://arxiv.org/abs/1611.06624 (cit. on p. 30).

[134] Mehdi S. M. Sajjadi et al. "Assessing Generative Models via Precision and Recall". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 5234–5243. URL: https://proceedings.neurips.cc/paper/2018/hash/f7696a9b362ac5a51c3dc8f098b73923-Abstract.html (cit. on p. 32).

[135] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 26).

[136] Maike Schwammberger. "Distributed Controllers for Provably Safe, Live and Fair Autonomous Car Manoeuvres in Urban Traffic". PhD thesis. Oldenburg University, Germany, 2021. URL: https://oops.uni-oldenburg.de/4961/ (cit. on p. 34).

[137] H. Stähle et al. "Towards the deployment of a centralized ICT architecture in the automotive domain". In: *2013 2nd Mediterranean Conference on Embedded Computing (MECO)*. IEEE, June 2013, pp. 66–69 (cit. on pp. 9, 10).

[138] Lukas Stappen et al. "Integrating generative artificial intelligence in intelligent vehicle systems". In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2023, pp. 5790–5797 (cit. on p. 5).

[139] George Stein et al. "Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=08zf7kTOoh (cit. on pp. 4, 32).

[140] Sitong Su et al. "F^3-Pruning: A Training-Free and Generalized Pruning Strategy towards Faster and Finer Text-to-Video Synthesis". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.5 (Mar. 2024), pp. 4961–4969. ISSN: 2159-5399. DOI: 10.1609/aaai.v38i5.28300. URL: http://dx.doi.org/10.1609/AAAI.V38I5.28300 (cit. on p. 13).

[141] Deqing Sun et al. *PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume*. 2018. arXiv: 1709.02371 [cs.CV]. URL: https://arxiv.org/abs/1709.02371 (cit. on p. 31).

[142] Mingjie Sun et al. *A Simple and Effective Pruning Approach for Large Language Models*. 2024. DOI: https://doi.org/10.48550/arXiv.2306.11695. arXiv: 2306.11695 [cs.CL] (cit. on p. 12).

[143] Shuo Sun et al. "Drivescenegen: Generating diverse and realistic driving scenarios from scratch". In: *IEEE Robotics and Automation Letters* (2024) (cit. on p. 28).

[144] Yoon Lee Sung, Jaemin Yoon, and Mohit Bansal. "Ecoflap: Efficient coarse-to-fine layer-wise pruning for vision-language models". In: *arXiv preprint arXiv:2310.02998* (2023) (cit. on p. 15).

[145] Cheng Tai et al. "Convolutional neural networks with low-rank regularization". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and

Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.06067 (cit. on p. 20).

[146] Jiapeng Tang et al. *DiffuScene: Denoising Diffusion Models for Generative Indoor Scene Synthesis*. Mar. 12, 2024. DOI: 10.48550/arXiv.2303.14207. arXiv: 2303.14207 [cs]. URL: http://arxiv.org/abs/2303.14207. Pre-published (cit. on p. 35).

[147] Chongyang Tao et al. "Structured pruning for efficient generative pre-trained language models". In: *Findings of the Association for Computational Linguistics: ACL 2023*. July 2023, pp. 10880–10895 (cit. on p. 14).

[148] Zachary Teed and Jia Deng. "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow (Extended Abstract)". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Sister Conferences Best Papers. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4839–4843. DOI: 10.24963/ijcai.2021/662. URL: https://doi.org/10.24963/ijcai.2021/662 (cit. on p. 31).

[149] Edward Teng and Rafael Bidarra. "A semantic approach to patch-based procedural generation of urban road networks". In: *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 2017, pp. 1–10 (cit. on p. 35).

[150] Hugo Thimonier et al. "Learning Long Term Style Preserving Blind Video Temporal Consistency". In: *2021 IEEE International Conference on Multimedia and Expo (ICME)*. 2021, pp. 1–6. DOI: 10.1109/ICME51207.2021.9428445 (cit. on p. 31).

[151] Arunachalam Thirunavukkarasu and Domenik Helms. "Using Network Architecture Search for Optimizing Tensor Compression". In: *7th International Embedded Systems Symposium (IESS)*. Ed. by Stefan Henkler et al. Vol. AICT-669. Designing Modern Embedded Systems: Software, Hardware, and Applications. Part 3: Engineering Applications of Artificial Intelligence Targeting Embedded Systems. Lippstadt, Germany: Springer Nature Switzerland, Nov. 2022, pp. 139–150. DOI: 10.1007/978-3-031-34214-1_12. URL: https://inria.hal.science/hal-04645298 (cit. on p. 24).

[152] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *CoRR* abs/2012.12877 (2020). arXiv: 2012.12877. URL: https://arxiv.org/abs/2012.12877 (cit. on p. 13).

[153] Thomas Unterthiner et al. *Towards Accurate Generative Models of Video: A New Metric Challenges*. 2019. arXiv: 1812.01717 [cs.CV]. URL: https://arxiv.org/abs/1812.01717 (cit. on p. 30).

[154] Balakrishnan Varadarajan et al. "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 7814–7821 (cit. on p. 33).

[155] S. K. Rajendrakumar Vuruma et al. "From Cloud to Edge: Rethinking Generative AI for Low-Resource Design Challenges". In: *arXiv preprint arXiv:2402.12702* (2024) (cit. on p. 5).

[156] Hongjie Wang, Bhishma Dedhia, and Niraj K. Jha. "Zero-TPrune: Zero-Shot Token Pruning through Leveraging of the Attention Graph in Pre-Trained Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2024, pp. 16070–16079 (cit. on p. 13).

[157] Ting-Chun Wang et al. "High-resolution image synthesis and semantic manipulation with conditional gans". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8798–8807 (cit. on p. 28).

[158] Xiaofeng Wang et al. “Worlddreamer: Towards general world models for video generation via predicting masked tokens”. In: *arXiv preprint arXiv:2401.09985* (2024) (cit. on p. 28).

[159] Yuqi Wang et al. “Driving into the future: Multiview visual forecasting and planning with world model for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 14749–14759 (cit. on pp. 28, 29).

[160] Cedric Deslandes Whitney and Justin Norman. “Real Risks of Fake Data: Synthetic Data, Diversity-Washing and Consent Circumvention”. In: *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’24. Rio de Janeiro, Brazil: Association for Computing Machinery, 2024, pp. 1733–1744. ISBN: 9798400704505. DOI: 10.1145/3630106.3659002. URL: https://doi.org/10.1145/3630106.3659002 (cit. on p. 8).

[161] Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. “Compact and Computationally Efficient Representation of Deep Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* PP (May 2019), pp. 1–14. DOI: 10.1109/TNNLS.2019.2910073 (cit. on pp. 22, 23).

[162] Guangxuan Xiao et al. “SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models”. In: *Proceedings of the 40th International Conference on Machine Learning*. 2023 (cit. on p. 18).

[163] Haozhe Xie et al. *CityDreamer: Compositional Generative Model of Unbounded 3D Cities*. June 5, 2024. DOI: 10.48550/arXiv.2309.00610. arXiv: 2309.00610 [cs]. URL: http://arxiv.org/abs/2309.00610. Preprinted (cit. on p. 35).

[164] Min Xu et al. “Unleashing the power of edge-cloud generative AI in mobile networks: A survey of AIGC services”. In: *IEEE Communications Surveys & Tutorials* (2024) (cit. on pp. 5, 37).

[165] Hongyang Yan and Yi Li. “A Survey of Generative AI for Intelligent Transportation Systems”. In: *arXiv preprint arXiv:2312.08248* (2023) (cit. on p. 5).

[166] Chenbin Yang and Huiyi Liu. “Stable Low-Rank CP Decomposition for Compression of Convolutional Neural Networks Based on Sensitivity”. In: *Applied Sciences* 14 (Feb. 2024), p. 1491. DOI: 10.3390/app14041491 (cit. on p. 22).

[167] Guandao Yang et al. “PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South): IEEE, Oct. 2019, pp. 4540–4549. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00464. URL: https://ieeexplore.ieee.org/document/9010395/ (cit. on p. 35).

[168] Lehao Yang et al. “Street Layout Design via Conditional Adversarial Learning”. In: *arXiv preprint arXiv:2305.08186* (2023) (cit. on p. 35).

[169] Minghao Yin et al. “Gohsp: A unified framework of graph and optimization-based heterogeneous structured pruning for vision transformer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 9. June 2023, pp. 10954–10962 (cit. on p. 16).

[170] Eleni Zapridou, Ezio Bartocci, and Panagiotis Katsaros. “Runtime Verification of Autonomous Driving Systems in CARLA”. In: *Runtime Verification*. Ed. by Jyotirmoy Deshmukh and Dejan Ničković. Cham: Springer International Publishing, 2020, pp. 172–183. ISBN: 978-3-030-60508-7. DOI: 10.1007/978-3-030-60508-7_9 (cit. on p. 33).

[171] Dingkun Zhang et al. *LAPTOP-Diff: Layer Pruning and Normalized Distillation for Compressing Diffusion Models*. 2024. arXiv: 2404.11098 (cit. on p. 14).

[172] Mingyang Zhang et al. *LoRAPrune: Pruning Meets Low-Rank Parameter-Efficient Fine-Tuning*. 2024. arXiv: 2305.18403 [cs.LG]. URL: https://arxiv.org/abs/2305.18403 (cit. on p. 14).

[173] Qin Zhang et al. "A survey on deep learning benchmarks: Do we still need new ones?" In: *Benchmarking, Measuring, and Optimizing: First BenchCouncil International Symposium, Bench 2018, Seattle, WA, USA, December 10-13, 2018, Revised Selected Papers 1*. Springer. 2019, pp. 36–49 (cit. on p. 34).

[174] Xiangyu Zhang et al. "Accelerating Very Deep Convolutional Networks for Classification and Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.10 (2016), pp. 1943–1955. DOI: 10.1109/TPAMI.2015.2502579 (cit. on p. 21).

[175] Qibin Zhao et al. "Tensor Ring Decomposition". In: (June 2016) (cit. on p. 23).

[176] Tianchen Zhao et al. *ViDiT-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation*. 2024. arXiv: 2406.02540 [cs.CV] (cit. on p. 19).

[177] Kun Zhou et al. "Don't make your llm an evaluation benchmark cheater". In: *arXiv preprint arXiv:2311.01964* (2023) (cit. on p. 6).

[178] Linqi Zhou, Yilun Du, and Jiajun Wu. "3D Shape Generation and Completion through Point-Voxel Diffusion". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). Montreal, QC, Canada: IEEE, Oct. 2021, pp. 5806–5815. ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.00577. URL: https://ieeexplore.ieee.org/document/9711332/ (cit. on p. 35).