Institute of Computer Science
Faculty of Mathematics and Computer Science
Database Group

Knowledge Distillation of Large Language Models for Use Cases of the German Aerospace Center

Master's Thesis

Submitted by:
Konstantin Witossek

Matriculation number:
3775768

Supervisors:
Dr. Victor Christen
Dr. Julia Fligge-Niebling

in cooperation with
DLR German Aerospace Center
Institute of Data Science
Machine Learning Group

© 2025

This work, including its parts, is **protected by copyright**. Any use outside the strict limits of the copyright law is inadmissible and punishable without the author's consent. This applies in particular to duplications, translations, microfilms as well as storage and processing in electronic systems.

Abstract

Large Language Models (LLMs) have shown remarkable capabilities; however, their high computational demands present notable challenges for deployment in resource-limited environments, especially in real-time, domain-specific applications at the German Aerospace Center (DLR). This thesis tackles this issue by using Knowledge Distillation (KD) to compress a large, powerful teacher model into a smaller, more computationally efficient student model.

This work proposes a comprehensive framework for distilling knowledge from an 8-billion-parameter teacher (LLaMA 3.1) to a 1-billion-parameter student (LLaMA 3.2). The methodology is centred around a DLR-specific use case: an autonomous vehicle system which handles natural language voice commands in real-time, integrating contextual sensor data to ensure safe and efficient command execution. For this purpose, a novel synthetic data generation pipeline was created to build a domain-specific dataset.

The distilled student model was tested thoroughly against the teacher model and also a baseline student of the same size. For the DLR-specific task, the distilled model showed better safety-awareness, with a clear reduction in critical failures compared to the baseline. While the improvements on a difficult public function-calling benchmark were more modest, the distillation still led to a big increase in the baseline's accuracy. These performance gains came together with an eightfold reduction in model size and almost a fivefold boost in inference speed, which shows the model's potential for on-board deployment. All in all, these results suggest that knowledge distillation is a valid and practical strategy for building efficient and more reliable LLMs for specialised, high-stakes use cases.

Konstantin Witossek
3775768

Contents

List of Abbreviations

CE	cross-entropy loss
CoT	chain-of-thought
DLR	Deutsche Zentrum für Luft- und Raumfahrt e. V.
DPO	Direct Preference Optimization
GPT	Generative pre-trained transformer
ICL	In-context Learning
KD	Knowledge Distillation
KL	Kullback-Leibler divergence
LLM	Large Language Model
LoRA	Low-Rank Adaptation
NLG	Natural Language Generation
NLU	Natural Language Understanding
PEFT	Parameter-Efficient Fine-Tuning
RAG	Retrieval Augmented Generation
RL	Reinforcement Learning
RLHF	Reinforcement Learning from Human Feedback
SAM	Sharpness-aware minimization

List of Figures

List of Tables

1. Introduction

1.1. Background and Motivation

Large Language Models (LLMs) such as OpenAI's GPT series [1], Meta's LLaMA family of open source models [2], Google's Gemini models [3] and various other series of large language models [4], [5] have revolutionized the field of natural language processing (NLP) by providing a wide range of capabilities for understanding, generating and reasoning about text [6]–[9]. These models, trained on huge amounts of data, are used in a variety of different use cases. These range from question answering (QA), machine translation, applications in the medical field, software engineering or robotics, to applications in the scientific field [6]. The enormous size of the models (for example LLAMA 405B - 405 billion parameters) and the associated computational effort to train and operate these models pose major challenges when used in real-world use cases [10]. This is particularly the case in environments where only limited computing resources are available [11], [12].

The German Aerospace Center (DLR) [13] is Germany's national research center for aerospace, energy, transport and security. With over 30 locations and various institutes, the DLR conducts both application-oriented and basic research. The German Aerospace Center is researching the integration of foundation models (mainly large language models) in various DLR use cases in an impulse project called the Foundation Models Project, including use cases in aerospace, in autonomous vehicles, in robotics, in ship traffic management and in earth observation. In these use cases, LLMs offer great potential for improving existing processes. For example, they can improve human-machine interaction in spacecraft and aircraft or in robotics, enable processing of speech into commands in autonomous vehicles or be used in the evaluation of protocols in aviation. Despite these promising applications, the practical use of LLMs remains challenging due to their high computational requirements, memory requirements, and the need for domain-specific customization [12]. This is especially the case when local models are to be used (e.g. for safety or availability reasons). In safety-critical and real-time environments such as autonomous navigation or space missions, computational power and inference speed are of great importance. Running these models on edge devices or embedded systems is often impractical due to limited processing power and energy constraints [11].

In order to use these powerful language models and to overcome the existing barriers to their use, this work focuses on the method of Knowledge Distillation (KD). KD is a model compression technique that transfers knowledge from a large, complex *teacher* model, as referred to in the literature, to a smaller, more efficient *student* model [14], [15]. By using KD, it should be possible to significantly reduce the model size while maintaining a high level of quality and functionality.

The research motivation stems from the increasing need to deploy LLMs in operational applications. According to Xu *et al.* [16] the optimization of smaller models through distillation makes them deployable on hardware systems with limited computing capabilities. The resulting models become adaptable to specific use case requirements through the incorporation of domain-specific knowledge [17]. The research evaluates the performance-cost tradeoffs of model compression while confirming that the distilled models function effectively and reliably in their intended applications.

The development of a systematic framework for knowledge distillation in LLMs will help achieve the goal of deploying advanced AI models in autonomous systems.

This thesis examines knowledge distillation approaches to reduce large language models in particular resource-constrained domains. The research shows that distilled models deliver strong performance at lower computational expense through DLR-relevant use cases which makes LLM applications feasible in aerospace and autonomous systems.

1.2. Problem Statement

The practical deployment of large language models remains difficult because their high computational needs along with their resulting energy usage. The deployment of these models in real-time resource-constrained environments such as those found at the German Aerospace Center (described in section 3.3) becomes especially challenging for autonomous vehicles, robotics and aerospace applications. The direct deployment of this models on edge devices or embedded systems faces challenges because of limited computational resources, but is needed, because it ensures reliability, security and privacy needs. The use of pre-trained smaller language models in a conventional local operation leads to unsatisfactory performance because of their limited model size according to [11], [12]. The insufficient performance often creates problems because complex domain-specific tasks require high quality outputs. A systematic approach to compress and adapt LLMs is needed because specialized use cases require both efficiency and performance without major performance degradation. Consequently, the main research challenge of this work is: How can large language models be efficiently adapted for use in resource-constrained, domain-specific environments at DLR through knowledge distillation without significant losses in accuracy and reliability?

1.3. Research Objectives

The primary objective of this thesis is to systematically apply and evaluate knowledge distillation as a method for adapting a large language model to the resource-constrained, safety-aware environment of an autonomous vehicle use case at the DLR. The central aim is to create a computationally efficient student model that retains a significant portion of a larger teacher model's performance while being suitable for on-board deployment.

To achieve this primary objective, this work pursues several interconnected secondary goals. The first is to define the specific operational and safety requirements for an LLM-based voice command interface within the DLR's autonomous vehicle context. Building on this, a key objective is to design and implement a complete experimental framework, which includes the development of a novel synthetic data generation pipeline to create a domain-specific, safety-annotated training corpus. Subsequently, this framework is used to implement a logit-based knowledge distillation strategy, transferring knowledge from an 8-billion to a 1-billion parameter model. The performance of the resulting distilled student model is then rigorously evaluated and benchmarked against both the original teacher and a non-distilled baseline, using metrics for task accuracy, safety-aware rejection, and computational efficiency. Finally, the empirical findings are synthesized to assess the viability of this approach and to derive practical implications for deploying distilled LLMs in similar high-stakes, domain-specific applications.

1.4. Contributions of This Thesis

This thesis is making a submissions in the area of applied artificial intelligence. It systematically undertakes an analysis of the knowledge distillation techniques which are useful for compressing large language models and concludes with a guide for the implementation of these methods. Also a pipeline is designed for the generation of synthetic data. This pipeline is capable of producing accurate training data for an autonomous vehicle use case related to DLR in the absence of any publicly shared corpus. Moreover, this study will produce a domain-specialized, lightweight model that has most of the teacher's accuracy but a reduction in size, and a significantly faster inference time. Finally, the research offers operational recommendations to support the implementation of these distillation methods in similar safety-critical domains, thereby advancing the adoption of efficient, trustworthy AI models in complex real-time systems.

1.5. Thesis Structure

The research follows a seven-chapter structure which leads readers through the entire research process from basic concepts to final results and their interpretation. The research begins in Chapter 1 by presenting an introduction which explains the background and research motivation and defines the main problem and outlines the core objectives. The following chapter (Chapter 2) establishes the theoretical framework through an analysis of Large Language Models and model compression techniques and Knowledge Distillation principles. The research connects to practical applications through Chapter 3 which explains the DLR operational context and presents the autonomous vehicle application as the primary focus of this investigation. Chapter 4 explains the research methodology and experimental design by presenting research questions alongside experiment design details and dataset information and model configurations and technical details about the distillation pipeline implementation. The results appear in Chapter 5 which presents both quantitative and qualitative assessments of the findings through numerical analysis and descriptive observations to evaluate student model performance relative to the teacher model and a basic baseline. The following section presents the results before Chapter 6 interprets their meaning and discusses their potential applications at DLR and addresses study limitations. The thesis concludes in Chapter 7 by summarizing key findings while proposing potential future research directions that expand upon this study.

2. Theoretical Foundations and Related Work

Large Language Models have drawn much attention in the Natural Language Processing domain. As Patil and Gudivada [18] explain, the efficacy of their work is principally down to the fact that they have been trained on massive text corpora. They deliver strong performance across a wide range of NLP tasks. Yang *et al.* [19] note the versatility of the LLMs and apply them to a wide variety of situations: knowledge-based tasks, natural language understanding and generation. However, as Min *et al.* [20] mention, the computational resources required to run these large models present significant difficulties in real-world implementations, particularly with respect to efficiency, cost, and latency. In this section, the basics, advantages, and limitations of using LLMs in real-world applications are explained. First, LLMs are introduced, followed by an explanation of the concept of knowledge distillation in general and in the context of LLMs. This foundation will serve as the basis for further investigation of model compression techniques and knowledge distillation techniques in the following sections.

2.1. Large Language Models: Capabilities and Challenges

Large Language Models are complex neural network architectures that usually have hundreds of millions to hundreds of billions of parameters based on their design and target application. They are mainly based on transformer architectures that use self-attention mechanisms to capture dependencies and contextual information in text data for long sequences. LLMs are trained in a two stage process [7], beginning with self supervised pre-training on huge general purpose text corpora, followed by task specific fine tuning to particular downstream tasks. Some of the well known LLMs include the GPT series by OpenAI [1], LLaMA models by Meta [2], Gemini by Google [3] and Claude by Anthropic [4]. The models have revolutionized the NLP domain in a short time and are at the forefront of developments in various applications of language understanding, text generation and reasoning.

The fast evolution of large language models is reflected not only in their capabilities but also in their increasing scale over time. Figure 2.1 illustrates this growth. The visualization shows a clear trend toward larger model architectures, with some models now exceeding hundreds of billions of parameters. To complement this visual overview, a detailed tabular comparison including model names, developers, release dates, parameter counts, and licensing information is provided in Appendix A (Table A.1).

2.1.1. Capabilities of LLMs

Recent advances in artificial intelligence have led to the development of this large models, which are transforming how machines interact with and process human language. The models demonstrate both text understanding capabilities and text generation abilities while performing complex tasks in various domains. As shown in Figure 2.2, the capabilities of LLMs can be categorized into four main

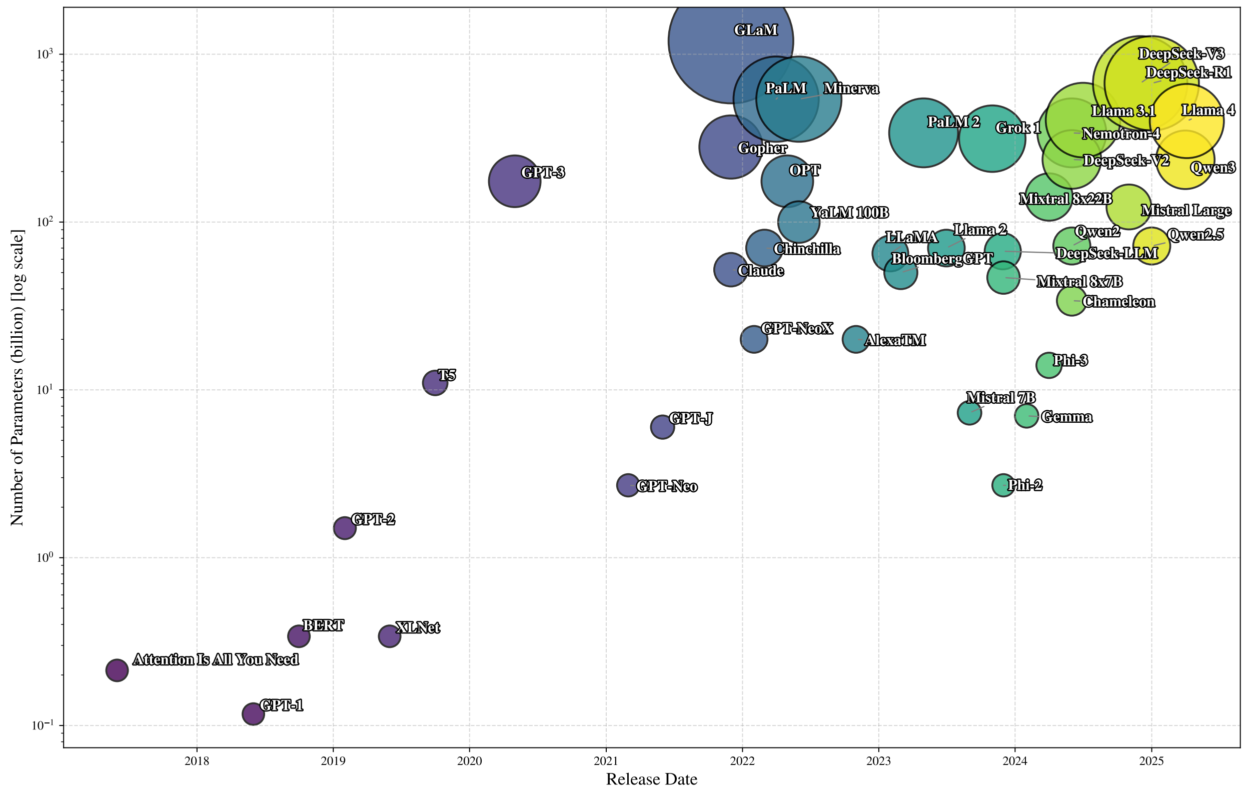

Figure 2.1.: Growth of Large Language Models over time. The number of parameters (in billions, logarithmic scale) is shown on the y-axis, while the x-axis represents the release date. Each bubble represents a specific LLM; bubble size encodes parameter count. Data based on [21]

fields: natural language understanding and generation, reasoning and decision-making, multimodal integration, and domain-specific adaptation.

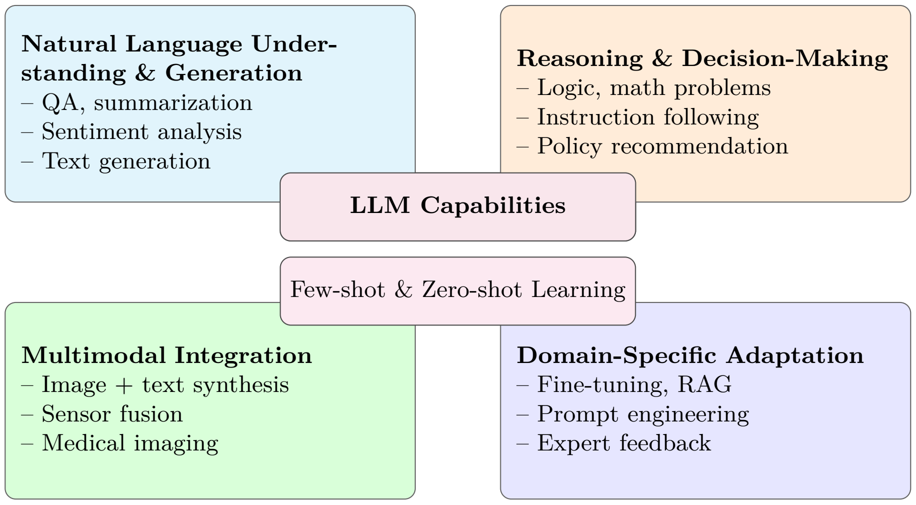

Figure 2.2.: Overview of the core capabilities of large language models.

Natural Language Understanding and Generation LLMs demonstrate excellent performance in both natural language understanding (NLU) and natural language generation (NLG). The pre-training process on extensive text corpora enables the models to learn these capabilities which results in the encoding of linguistic structure, contextual relationships and semantic dependencies that apply to multiple languages and domains. In the aspect of NLU, LLMs can understand complicated textual information, decipher meanings, classify sentiments, and gather information from unstructured text. Tasks like question answering, named entity recognition, text classification, and semantic parsing are also advanced by LLM-based methods. It has been proven that models such as GPT, LLaMA, and Gemini perform exceptionally well in standard tests of machine comprehension and reasoning capabilities [22]–[24]. In the NLG domain, LLMs are very effective at producing language that is coherent, relevant to context, and similar to that of a human writer. They are applied to almost all areas of automated content creation, conversational AI, text summarization, machine translation, and even code generation. The self-attention mechanism in transformers enables these models to produce smooth and grammatically correct text and ensures coherence over long passages of text. Nevertheless, together with the benefits of generation, drawbacks like hallucination, bias, and factual errors are major issues when applying LLMs to real-world use cases [25], [26]. The integration of NLU and NLG capabilities makes LLMs transformative tools used across diverse sectors such as healthcare (e.g. clinical note summarization), law (e.g. document analysis), customer service (e.g. chatbots), education, and many others.

Reasoning and Decision-Making Beyond language comprehension and generation, LLMs are capable of performing complex reasoning and supporting decision-making processes. These models can interpret complicated instructions and assist in solving problems across various domains. Recent research [27], [28] demonstrates that transformer-based models can tackle arithmetic word problems, commonsense reasoning, and probabilistic inference directly from text. Additionally, techniques such as Reinforcement Learning from Human Feedback (RLHF) have been employed to refine step-by-step reasoning and reduce errors [29], [30]. In practical applications, LLMs support decision-making in areas like medical diagnosis [31], financial analysis [32], and law [33], where they help synthesize information, weigh alternatives, and provide context-aware suggestions. However, as Saxena *et al.* [34] and Chi *et al.* [35] note, LLMs are not true cognitive agents and may still exhibit biases, hallucinations, and overconfidence in incorrect predictions. Improving the reliability and interpretability of LLM-based reasoning remains an active area of research.

Multimodal Integration Incorporating LLMs to handle multiple forms of data is a big advancement in this field. It leads to the capability of understanding and reasoning on various forms of input, e.g., text, images, and structured data, which is useful in real-world applications. This is done by unified tokenization that align different modalities into one latent space, and attention across modalities to enhance contextual inferences. Methods include early fusion that combines the raw multimodal data before processing [36], and late fusion [37] that combines the outputs of the modality specific encoders at a later stage. This methods are applied in different practical use cases. In autonomous systems, for instance, multimodal LLMs make decisions using sensor data, in medical AI such systems extract information from clinical notes and images for diagnosis, and in scientific areas, they help to understand structure for drug discovery. In addition, they improve creative tasks (e.g., text-to-image generation) and human computer interaction (e.g., gesture interfaces). Some of the challenges here include e.g. data misalignment, especially when using different modalities. Also the high computational complexity of training such models or Bias that is likely to propagate across different modalities, is a problem.

Domain-Specific Adaptation The generalization capabilities of LLMs across many topics do not guarantee peak performance in specialized fields so domain-specific adaptation might become necessary in some cases. The adaptation/integration of LLMs for specific applications uses techniques like retrieval-augmented generation (RAG), prompt engineering and continued pre-training on domain-relevant corpora. The models could achieve a better accuracy and relevance through additional pre-training on industry-specific texts which helps it learn specialized terminology and context. They also could achieve precise and context-aware responses through supervised fine-tuning and reinforcement learning with expert feedback. The adaptation of LLMs to new tasks could also be done through few-shot [38] and zero-shot [27] learning strategies which require minimal data. Also the integration of structured knowledge sources including ontologies databases and scientific literature is an option, to enhance factual grounding and minimizing hallucinations. But the implementation of domain-specific adaptation faces ongoing challenges because it requires balancing domain-specific knowledge with generalization capabilities while addressing biases in specialized datasets and ethical issues from proprietary or sensitive data use (e.g., privacy, consent, and data security) [18]. Research continues to develop better fine-tuning approaches, more efficient resource usage and effective evaluation metrics for domain-specific LLMs.

Few-shot and Zero-shot Learning The few-shot and zero-shot learning capabilities of LLMs allow them to perform tasks across different domains using minimal or no task-specific data according to [27], [38]. he few-shot learning method proves beneficial when there is limited labeled data because it allows adaptation through only a few examples. Zero-shot learning enables LLMs to complete tasks without any examples at all, by relying on a well structured prompt and their pre-trained knowledge for things like text classification, summarization, and translation. Although these methods are flexible, they are sensitive to prompt design and may fail in domain specific cases.

2.1.2. Challenges of LLMs

The deployment of LLMs in real-world applications encounters various technical, ethical, and operational challenges despite their status as the state-of-the-art in NLP. These obstacles need to be resolved to achieve the complete potential of these advanced systems in different domains.

Balance Between Performance and Efficiency The impressive capabilities of modern language models come at a substantial cost in computational resources. Finding the best balance between effectiveness and efficiency remains a challenge for practical applications. The rapid growth in model size has led to high increases in computing and energy requirements, making inference costs a significant concern [39]. Interestingly, while much attention has focused on reducing energy consumption during training, inference optimization has been relatively neglected despite inference being far more frequently performed. Recent research has begun addressing this efficiency gap. Argerich and Patiño-Martínez [39] introduced a methodology for measuring energy consumption during inference and investigated how factors such as model size, architecture, parallelized attention, and vocabulary size affect efficiency. Their findings suggest that increasing batch size and applying quantization techniques can enhance efficiency without significantly compromising model accuracy. Similarly, Rostam *et al.* [40] surveyed optimization techniques focusing on approaches like distributed training, model partitioning, and hardware acceleration. Their work describes methods such as memory-aware scheduling and model parallelization to improve efficiency, particularly for very large architectures. These results underscore the importance of designing principles that reduce computational complexity while preserving effectiveness, especially when hardware resources are limited. The efficiency-effectiveness balance thus remains a central concern for both researchers and practitioners implementing these technologies.

Domain Adaptation versus Generalization Advanced language models face challenges when performing domain-specific tasks despite their general language competence. The process of domain adaptation faces different essential obstacles which include data scarcity, overfitting and specialized knowledge integration. The corpora available for niche domains are usually limited and of poor quality which makes it difficult to train robust models.

According to Bansal *et al.* [41] deep learning models require enough data to learn generalization and limited data can result in overfitting. The process of fine-tuning LLMs with domain-specific data improves performance in that domain but results in decreased generalization across other domains because of overfitting or catastrophic forgetting. The problem can be solved by data augmentation techniques (described later in section 2.3.6) such as synthetic data generation and transfer learning which improve domain-specific performance while maintaining generalization accuracy. Domain specialization introduces new difficulties because of data heterogeneity and knowledge complexity. Ling *et al.* [42] explain that general models sometimes cannot be directly applied to specific domains because each domain functions under its own distinct objectives and constraints which include ethical, cultural and regulatory frameworks. Researchers have investigated knowledge-enriched strategies including knowledge graph embeddings, retrieval-augmented generation and meta-learning to improve contextual awareness and adaptability in specialized contexts.

Domain knowledge presents a major obstacle because it constantly changes. Medical and legal fields introduce fresh terminology and concepts which continue to evolve. The research by Ling *et al.* [42] shows that models require incremental knowledge updates through structured knowledge integration and human feedback to stay relevant. The adaptation of domain knowledge across numerous different domains proves to be a difficult task. The fine-tuning approach succeeds when applied to a small number of domains yet becomes impractical when dealing with large numbers of domains. The combination of white-box (full fine-tuning), grey-box (partial adaptation), and black-box (prompting) methods provides a more adaptable solution according to [42]. Future strategies will probably implement hybrid methods which combine meta-learning automation with both structured and unstructured data to achieve domain-specific knowledge and generalization capabilities.

Bias, Hallucination and Interpretability Although these models are developed from state-of-the-art architectures, they receive their bias directly from their training data. Bias propagation occurs when training data biases lead to discriminatory outputs that produce unfair results and undermine model reliability [43]. Also the phenomenon of hallucination, where coherent but incorrect text is produced, is a problem. Users in high-risk domains such as healthcare, law and aerospace face trust erosion because hallucinations produces false information. The lack of understanding about the LLMs decision-making processes creates challenges to identify their conclusion sources.

Integration of Multiple Modalities The upcoming generation of AI systems is moving towards multimodal capabilities, which means that they are integrating text, images, audio, and structured data. Although this evolution increases the potential applications of LLMs, it also raises challenges related to data alignment, computational efficiency, and interpretability. As stated by Li *et al.* [44], good multimodal models are supposed to correctly synchronize various data sources in order to provide useful outcomes. This requires advanced fusion methods, including cross-attention mechanisms together with contrastive learning, to establish effective mappings between textual, visual, and other forms of data. However, there is a problem of semantic consistency across modalities since, many times, the training data is not well aligned between the different input modalities.

Evaluation and Ethical Considerations Traditional benchmarks often fall short when it comes to evaluating LLMs, as they don't really capture the full range of model behavior, like domain-specific knowledge or how well they understand context [45]. There are several issues that need to be addressed, including data privacy worries, possible misuse of the technology, and the broader societal impacts of automated decision-making systems. Developing better evaluation metrics is still crucial so that assessment frameworks can reflect both general and specialized capabilities of LLMs. The ethical implications of these models also deserve close attention when putting Responsible

AI principles into practice. How data is used, how user privacy is protected, and how misuse is prevented should be priorities to avoid harmful outcomes. These models should be used responsibly in sensitive areas and stay as transparent as possible.

2.2. Model Compression Techniques

The recent progress in natural language processing described in Section 2.1 of this thesis has led to highly capable models that perform well in text generation, summarization, and question-answering tasks. However, the main issue of the computational resources required for training and deployment remains a challenge, especially for devices with limited capacity. The number of parameters is large, the memory requirements are high, and the latency is high, which makes direct deployment unsuitable for applications and edge devices, such as those used in autonomous vehicles, embedded systems, or in aerospace within DLR. Model compression techniques offer effective solutions to address these problems by minimizing the dimensions of LLMs without significantly affecting their performance accuracy. These methods make it possible to create efficient models that deliver sufficient accuracy for various applications despite their reduced computational requirements. Smaller models can run on low-power hardware, thus reducing the need for high-end GPUs or cloud infrastructure. This approach is also essential when time constraints exist (in terms of inference) and access to large computational resources is either unavailable or restricted.

2.2.1. Motivation of Model Compression

Model compression exists because large language models continue to grow in size (see figure 2.1) while the applications need to function in environments with restricted computational power. Several factors contribute to the importance of efficient LLM deployment:

Latency is the delay between input and response in computational systems. In large-scale neural networks for natural language processing, high latency hinders timely critical applications like autonomous systems, where fast responses are important. Mukherjee and Hassan Awadallah [46] explain that deep neural networks have both high latency and memory requirements which prevent their use in such applications. Wang *et al.* [47] explain that higher model complexity creates higher response time during operation. The solution to this problem involves multiple model compression approaches which include knowledge distillation alongside structured pruning, quantization and low-rank factorization (see section 2.2.2). Mukherjee and Hassan Awadallah [46] show that distillation techniques can reduce inference latency while maintaining high accuracy level of the original model. Wang *et al.* [47] demonstrate that structured pruning enables hardware reduced computation and quantization accelerates processing through precision reduction, without significant accuracy degradation.

Memory Footprint The deployment of LLMs in resource-constrained environments faces limitations because of their substantial memory requirements during training and deployment. Memory footprint refers to the complete model size which includes parameters and activations and optimizer states. The process of training requires additional memory storage for gradients and optimizer states and intermediate activations which limits both scalability and accessibility. The memory footprint of inference requires memory allocation for model weights and activation storage during the inference process. For example the memory footprint of LLaMA models varies significantly depending on their model variant. For instance, LLaMA 3-8B (8 billion parameters) requires approximately 21 GB of VRAM to support inference with a context window of 1024 tokens and 10 concurrent requests. In contrast, LLaMA 3-70B (70 billion parameters) demands at least 140 GB of VRAM (without additional compression techniques - in full precision) just for model weights, requiring multiple GPUs (e.g., at least two A100 80GB GPUs) to fit in memory [48].

Energy Consumption The high computational demands and memory requirements of large language models translate directly into significant energy consumption. The energy required for LLM inference is primarily determined by the number of model parameters, memory access patterns, and the efficiency of the underlying hardware. Recent studies have shown that inference energy consumption can even surpass that of training, due to the repeated execution cycles required for serving real-world applications [39]. Profiling of various model architectures reveals that larger models not only consume more power per query but also tend to do so less efficiently, especially when deployed at scale [39], [40]. The main contributor to power consumption during inference is GPU utilization, which can be mitigated through optimizations such as increasing batch size and reducing model depth.

Model compression techniques, including quantization, pruning, and knowledge distillation, have proven effective in reducing energy consumption without a significant loss in performance. For instance, lowering numerical precision (e.g., to 8 or 4 bits) and distributing computations across optimized hardware accelerators can substantially improve energy efficiency. As highlighted in recent literature, the growing deployment of LLMs across different industries is driving the need for greener, more energy-efficient inference solutions [39], [40]. This consideration is particularly critical in applications where onboard AI systems must operate within strict energy budgets to ensure mission viability [48].

2.2.2. Taxonomy of Compression Techniques

Model compression techniques are essential for reducing the size and computational demands of large language models [11], [12], [14], [49], [50]. The techniques can be divided into four main categories: Quantization, Pruning, Knowledge Distillation and Low Rank Factorization (see Figure 2.3). Each method has its own advantages and disadvantages which are relevant to the particular application task.

Quantization reduces precision of numerical values like weights and activations to reduce memory and to increase the efficiency of inference. It can be done through post training quantization which converts pre-trained full precision models into lower precision formats without needing further

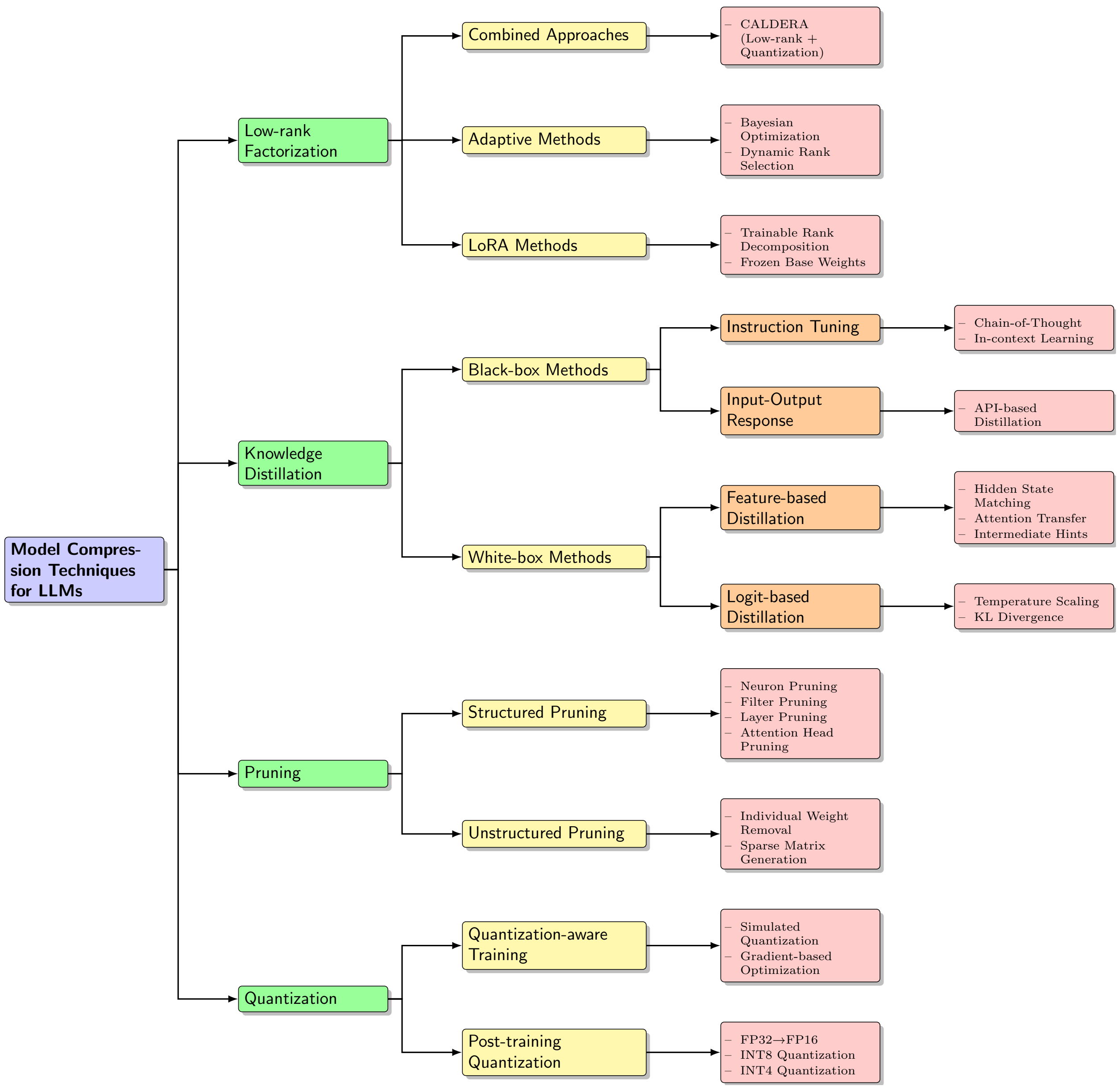

Figure 2.3.: Model Compression Techniques for LLMs

retraining or quantization aware training where quantization effects are incorporated during training [11], [12].

Pruning is used to remove unnecessary parameters of the model in order to decrease computational costs and model size [11], [47]. Unstructured pruning removes individual weights, resulting in sparse matrices, while structured pruning removes entire neurons, filters, or layers, making the model more hardware-friendly. Advanced strategies include dynamic pruning, which adapts pruning rates during inference based on input complexity, and subnetwork discovery, which seeks efficient subnetworks within large models [51]–[53].

Knowledge Distillation is a process of transferring knowledge from a teacher model that is generally large and complex to a student model that is smaller and less complex such that the student model is able to learn to mimic the behaviour of the teacher model with fewer parameters [14], [15], [54]. The different types that are used include logit based distillation in which the student learns from the softened output probabilities of the teacher or feature distillation in which the student learns to mimic intermediate feature representations. Some forms of distillation are typically task specific to the particular application. Another type, multi teacher distillation, is used to aggregate knowledge from multiple teachers to improve the generalization and robustness of the student model [16]. KD will be further explained in Section 2.3.

Low-rank factorization techniques exploit the inherent redundancy in large neural networks by decomposing weight matrices into low-rank components. A prominent example, "LoRA" (Low-Rank Adaptation), was proposed by Hu *et al.* [55] to fine-tune models by injecting trainable low-rank matrices into otherwise frozen pre-trained weights. Notably, LoRA introduces no additional inference latency, as the low-rank updates are merged with the base model after deployment. Building on this idea, Ji *et al.* [49] employ Bayesian optimization to adaptively select the rank per layer based on empirical covariance, improving compression effectiveness in large language models. Another recent approach, CALDERA, introduced by Saha *et al.* [50], combines low-rank decomposition with quantization to maximize compression ratios while maintaining competitive zero-shot performance.

In summary, the different compression techniques present unique trade-offs between model accuracy levels and computational efficiency. The process of quantization decreases memory usage and computational requirements through precision reduction yet produces minor accuracy degradation. Also pruning leads to a model reduction by removing less important parameters, but model performance suffers also here. Knowledge distillation method enables smaller models to learn vital information from larger models, which enables users to choose smaller specialized models. Weight matrices become more compact through low-rank factorization but this method can generate numerical instability because the approximation produces small errors that grow during computation, particularly in deep models.

2.2.3. Challenges and Future Directions in Model Compression

The improvements made with quantization, pruning and knowledge distillation techniques still face multiple significant obstacles that block their adoption and effective deployment. One of the major challenges is the balance between the memory footprint and the computational ressources needed, to apply these methods. Quantization and pruning do reduce model size. However, it often needs fine-tuning or retraining, which can be costly for large models [11], [12], [56]. Moreover, compressed models are usually less robust and generalizable than their full-sized counterparts. They work well with in-distribution data but may not be robust to adversarial perturbations [57].

The deployment of (compressed) models faces an ongoing challenge because of inference latency, especially on devices that have restricted processing power and restricted memory bandwidth. While model compression techniques reduce the number of parameters, they do not automatically lower memory access or data transfer costs, which often become the main bottlenecks during inference due to bandwidth limitations. The overhead becomes worse because of excessive I/O operations and poor memory layout designs. The latency problems can be partially solved through intelligent memory management and dynamic quantization techniques which adapt to runtime conditions and hardware constraints according to Mao *et al.* [56].

The main problem in using compressing techniques, is the accuracy deterioration which occurs when using really strong compression techniques. The methods that decrease model size and computational requirements also lead to reduced representational capacity and a worse generalization performance. The state-of-the-art techniques *AQLM* and *SpQR* use learning-based approaches to maintain accuracy during extreme compression through expressive quantization schemes and sparse representations according to [58], [59]. The research community continues to investigate how to find the best tradeoff between model size and latency and a good performance especially in safety-critical domains like healthcare or robotics where reliability and responsiveness are essential [58], [60]. Also an interesting direction is the combination of techniques like pruning and knowledge distillation.

Besides model level optimizations, system level improvements like efficient key-value cache management can boost the inference speed. Integrating these optimizations with model compression techniques could greatly reduce the computational cost in large scale deployments [61]. Another promising direction involves combining LLM compression with techniques such as Bayesian optimization for dynamic dimension allocation in low-rank approximations. By adjusting compression strategies based on empirical performance data, these approaches can produce models that are both more accurate and computationally efficient [49].

2.3. Introduction Knowledge Distillation

2.3.1. Definition and Concept of KD

Knowledge Distillation (KD) is a technique introduced to reduce the size of large, costly machine learning models. The core concept of KD is to mimic the role of a larger, more complex network, called the *teacher*, to train a smaller, less computationally expensive network, the *student*. This transfer enables the student model to produce similar outputs to those of the teacher model at a reduced computational cost [14]–[16], [62].

The idea behind KD is to use the teacher model's internal knowledge (good internal representation) and decision making to train the student model. The student model does not learn only directly from hard labels, the outputs or classifications provided in the training data, but also learns from soft labels, which are the teacher model's probability distributions over possible outputs. When probability distributions are softened, richer information, such as the confidence of incorrect classes, is retained. This allows the student model to learn about the detailed decision boundaries that the teacher model has learned (and the teacher may also be better able to represent these). A conceptual overview of this process is illustrated in Figure 2.4, which shows how knowledge is distilled from a large teacher model and transferred to a compact student model.

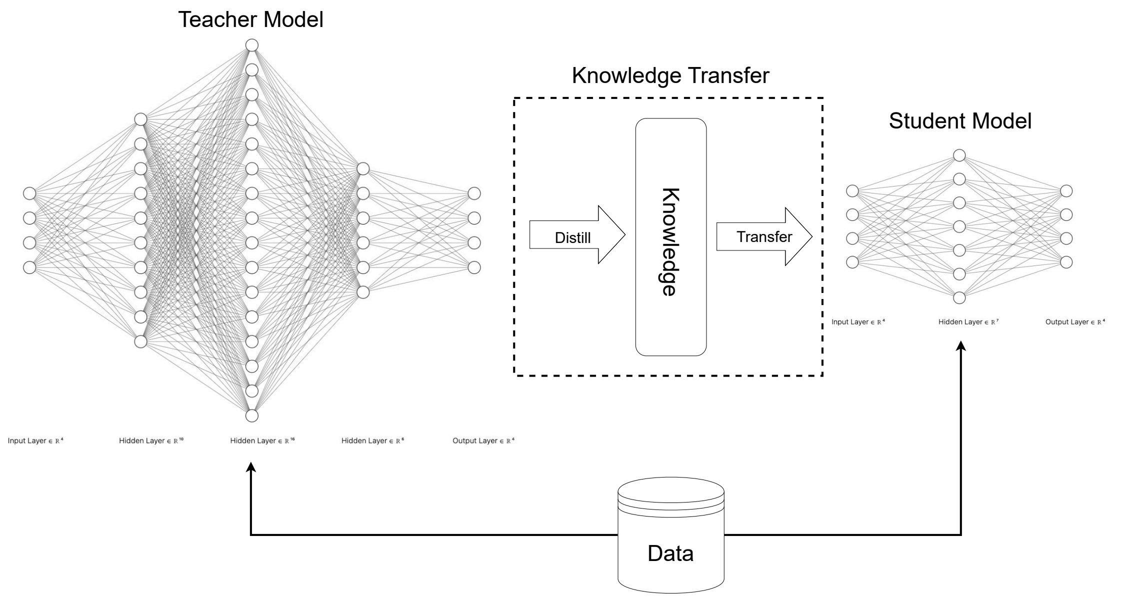

Figure 2.4.: The knowledge distillation process is schematically represented. The teacher model is trained on the original dataset and generates soft labels. These soft labels are then used in conjunction with the ground truth to train the student model using a combined loss function. Figure based on [14].

In practice, the student model is trained by minimizing a composite loss function that elegantly balances two objectives. The first term is a standard cross-entropy loss (L_{CE}) that matches the student's predictions to the ground-truth "hard" labels (y). The second, and more critical, term is the distillation loss, which uses the Kullback-Leibler (KL) divergence (L_{KL}) to align the student's output distribution with that of the teacher.

Two key hyperparameters govern this process, as shown in Equation 2.1. The coefficient α controls the balance between the two loss components, determining the emphasis placed on mimicking the teacher versus matching the true labels. Crucially, the temperature (τ) is used to soften the output probability distributions generated from the models' logits (z_t and z_s). A higher temperature creates a smoother distribution that reveals the teacher's nuanced knowledge about inter-class similarities, providing a richer training signal for the student. As proposed by Hinton et al., to ensure the soft and hard targets contribute appropriately to the gradient updates, the distillation loss term is scaled by a factor of τ^2 [15].

$$L = (1 - \alpha) \cdot L_{\text{CE}}(y, \sigma(z_s)) + \alpha \cdot \tau^2 \cdot L_{\text{KL}} \left(\sigma\left(\frac{z_t}{\tau}\right), \sigma\left(\frac{z_s}{\tau}\right) \right) \tag{2.1}$$

Knowledge distillation is especially valuable for applications that require efficient models under strict computational resource constraints, such as edge computing, embedded systems, and real-time inference tasks. This method makes it possible to deploy high-performing yet compact models, which increases the accessibility and practical use of advanced machine learning techniques in diverse environments, including aerospace and autonomous systems [14], [15], [54].

2.3.2. Historical Development of KD

Knowledge Distillation has become increasingly complex in recent years, expanding on basic concepts that have been developed over the past several decades. As described in section 2.3.1, at its core, KD is the process of transferring knowledge from one model, often a large, complex model referred to as the *teacher* model to another model, a smaller, simpler model known as the *student* model. The goal of this transfer is to get a similar performance as the teacher model with a much smaller and more efficient student model.

The idea of KD was first mentioned by Craven and Shavlik in 1996 [63], who suggested using trained neural networks to create labeled training data for decision trees. This early work showed that it was possible to transfer knowledge learned by one model to another, and in doing so, it paved the way for future work on knowledge transfer. The work of Bucila *et al.* [64] in 2006 marked a major step forward by making model compression an explicit problem. They proposed a formal approach to transferring knowledge from a large ensemble model to a single neural network. The possibility of using the output of one trained model as pseudo-labels to train a smaller model efficiently was thus clearly established.

The term *Knowledge Distillation* itself was formally coined by Hinton *et al.* in 2015 [15], who provided a comprehensive theoretical and empirical analysis of KD within the context of neural network compression. Their work demonstrated that softened output probabilities from the teacher model, obtained through a temperature-scaled softmax function, can serve as richer and more informative targets compared to conventional hard labels. This seminal paper also made it clear why soft targets are better than hard labels. Hard labels only provide information about the correct class, while soft labels also provide information about the probability of the wrong classes, which can help the student model learn better decision boundaries. This approach made many following KD researches focus on optimization efficiency and generalization benefits. After Hinton's influential formulation, KD was quick to gain popularity as researchers explored applying it across diverse domains. "FitNets", introduced by Romero *et al.* [65], extended the concept with intermediate feature representations in addition to output logits, so the student could learn deeper structural knowledge from the teacher model.

The KD framework was further enriched through techniques like "deep mutual learning" proposed by Zhang *et al.* [66], in which multiple student models simultaneously teach and learn from each other, eliminating the need for a single predefined teacher. Additionally, more recent developments, such as "Reinforcement Learning from AI Feedback" (RLAIF) [67], have employed reinforcement learning paradigms to refine student models dynamically based on feedback from the teacher.

Knowledge Distillation was quickly extended to a variety of specific application areas, and its usefulness was demonstrated in a broad range of tasks. For example, Yao *et al.* [68] proposed the "Adapt-and-Distill" method that combines domain adaptation and KD to train smaller, faster, and more efficient models for particular tasks like biomedical and computer science tasks. The approach successfully resolved the weak performance problems in specific domains when using general-purpose pre-trained models which demonstrated KD's ability to optimize and minimize models for particular applications.

2.3.3. KD in the Context of LLMs

The development of KD has been influenced by LLMs such as GPT-4 and LLaMA. The evolution of distillation techniques for LLMs has progressed from basic output logits and intermediate feature representations to transfer advanced cognitive abilities and reasoning patterns. At first, the KD methods were designed to perform simple compression and replication of the outputs of large models to smaller, more efficient models. However, recent research shows a shift toward transferring complex reasoning and cognitive patterns from LLMs. For instance, Hsieh *et al.* [69] introduced the "Distilling Step-by-Step" approach using rationales created via *chain-of-thought (CoT)* prompting.

Furthermore, Gu *et al.* [54] also solved some specific problems of KD in the context of generative language models by suggesting a reverse Kullback-Leibler divergence objective. The proposed approach enhanced distilled generative models by mitigating exposure bias and preventing overfitting to low-probability regions, which are common challenges faced by conventional KD methods. Consequently, it enhances accuracy and reliability in various language tasks. The use of KD in a domain-specific area, e.g. biomedical NLP, shows that it can be successfully applied and adjusted for new tasks. Gu *et al.* [70] showed that distilling a large model into a domain-specific model such as *PubMedBERT* not only reduced the size of the model but also improved the performance on the biomedical tasks. These results show that Knowledge Distillation is not only a tool for model compression, but also a powerful tool for transferring domain-specific knowledge and improving task-specific performance.

Recent surveys have methodically investigated Knowledge Distillation approaches which are designed for Large Language Models. Xu *et al.* [16] present a detailed analysis through different KD mechanisms and vertical domain applications to explain methodological progress in this area and also practical applications. Zhu *et al.* [11] focus on model compression techniques for LLMs in general, and they discuss new distillation approaches such as Chain-of-Thought prompting. The survey by Wan *et al.* [71] reviews methods to enhance the efficiency of LLMs through KD, dividing the approaches into white-box and black-box methods. These surveys collectively underscore the growing importance of KD.

Necessity of Knowledge Distillation for LLMs The growth of LLMs has resulted in increased computational complexity, latency, and energy usage. Although these models are successful in many NLP tasks, their accessibility is often limited to high-performance computing environments. Knowledge Distillation addresses these challenges by distilling essential knowledge into compact models that maintain performance while requiring fewer resources. This is especially useful in domain-specific tasks that require fast inference, e.g. aerospace, autonomous systems, mobile AI apps [72].

Differences from Traditional KD Approaches While the classical KD paradigm often relies on soft-label supervision via output logits, applying KD to LLMs presents additional challenges due to their open-ended, multi-task nature. LLMs are expected to handle complex reasoning, long-form generation, and diverse domains, which requires distillation methods to go beyond output-level matching. Contemporary approaches therefore integrate feature-based guidance, attention transfer, and instruction-level supervision to help preserve nuanced behaviours and high-level reasoning capabilities within the student mode [73]. These adaptations enable KD to support not only model compression, but also performance enhancement in e.g. logical reasoning or domain specialization [16].

Categorization of Knowledge Distillation Methods for LLMs KD techniques for LLMs can be classified into three main approaches of KD: white-box, black-box, and hybrid methods.

1. **White-Box Distillation:** These methods require access to the internal representations of the teacher model, allowing for detailed knowledge transfer. Techniques such as logit-based distillation (*DistilBERT* [74]), feature-based distillation (*TinyBERT* [75], *MobileBERT* [76]), and self-attention distillation (*MiniLM* [77]) fall under this category.
2. **Black-Box Distillation**: If the teacher model is a proprietary one and there is no access to its internal states then black box approaches are based only on the teacher's output. In-context learning distillation, instruction tuning (*LaMini-LM* [78]), and chain-of-thought distillation [79] enable student models to perform similar to that of closed source LMs without requiring parameter-level insights.
3. **Hybrid Distillation**: Hybrid distillation is a intermediate between white box and black box approaches, which focuses on optimizing the tradeoff between knowledge fidelity and accessibility. Techniques like multi-teacher distillation and self-alignment strategies have been found to improve model robustness and performance across different application domains [16].

2.3.4. White-box Distillation Methods

White-box methods differ from black-box approaches because they provide the student with deeper insights into the teacher's decision-making process through internal representations such as logits, attention maps or hidden states. White-box KD techniques for LLMs are typically categorized into two main categories: logits-based distillation and hint-based distillation. This section examines these two types of distillation.

Logits-based Distillation is a fundamental knowledge distillation technique widely used to compress large language models. In this approach, the student model is trained to approximate the softened output distribution of a more complex teacher model by minimizing the divergence between their predicted class probabilities. This method requires access to the teacher model's logits, making it a white-box distillation approach. As illustrated in Figure 2.5, the student model is supervised by both the ground-truth labels and the teacher's soft output distribution.

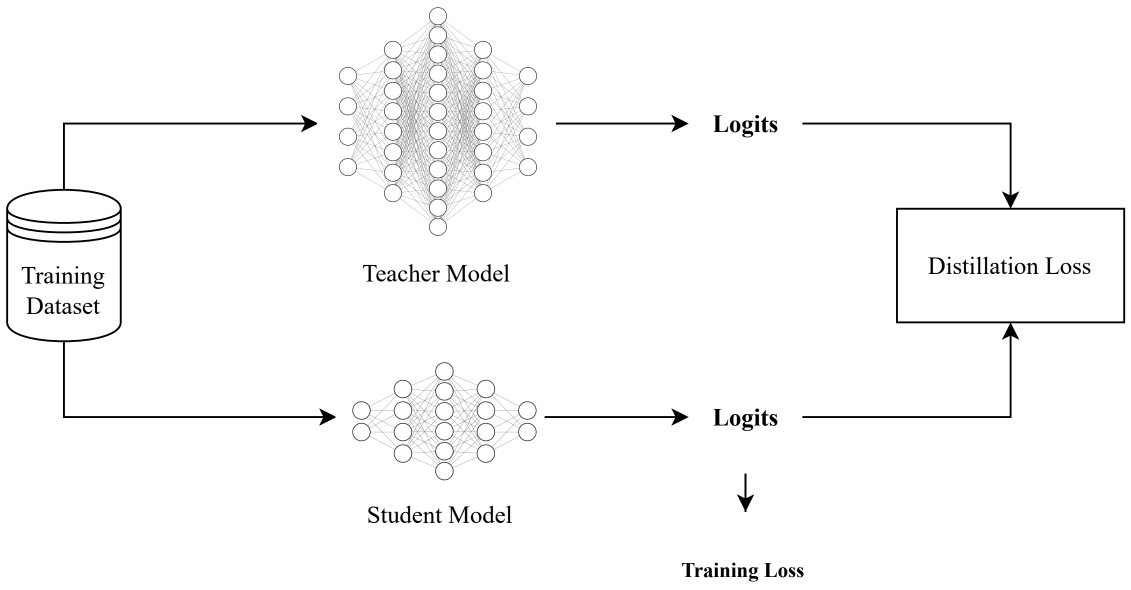

Figure 2.5.: Illustration of logits-based knowledge distillation. The teacher model generates output logits based on a shared training dataset. The student model minimizes a combined loss: a classification loss using ground-truth labels and a distillation loss aligning its outputs with the teacher's. Figure inspired by [80].

The objective is to align the student's probability distribution with that of the teacher, using a temperature-scaled softmax transformation applied to the teacher's and student's logits. The overall loss function is typically expressed in the following compact form:

$$\mathcal{L}_{\text{logits}} = \tau^2 \cdot \text{KL}\left(\sigma\left(\frac{z^T}{\tau}\right), \sigma\left(\frac{z^S}{\tau}\right)\right) \tag{2.2}$$

where z^T and z^S represent the logits of the teacher and student models, $\sigma(\cdot)$ is the softmax function, and τ is a temperature parameter that controls the smoothness of the resulting probability distributions. The scaling factor τ^2 is introduced to adjust the magnitude of gradients during training, as proposed in the work by Hinton et al. [15].

To make this formulation more explicit, the loss can be rewritten in terms of the *Kullback–Leibler (KL)* divergence between the softened output distributions:

$$\mathcal{L}_{\text{logits}} = D_{\text{KL}}(p^T \parallel p^S) \tag{2.3}$$

The KL divergence is a statistical measure that quantifies how one probability distribution diverges from a second, expected distribution. Mathematically, it is originally defined by Kullback and Leibler [81], [82] as:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{2.4}$$

where P and Q are discrete probability distributions defined over a common support $\mathcal{X}$. In the context of knowledge distillation, P corresponds to the teacher's softened output distribution p^T, and Q corresponds to the student's output distribution p^S.

The softened probabilities themselves are derived by applying a temperature-scaled softmax to the logits:

$$p_j^T = \frac{\exp(z_j^T/\tau)}{\sum_{k=1}^{C} \exp(z_k^T/\tau)}, \quad p_j^S = \frac{\exp(z_j^S/\tau)}{\sum_{k=1}^{C} \exp(z_k^S/\tau)} \tag{2.5}$$

Substituting these into the KL divergence yields the full expression of the logits-based distillation loss [73]:

$$\mathcal{L}_{\text{logits}} = \sum_{j=1}^{C} p_j^T \log \left(\frac{p_j^T}{p_j^S} \right) \tag{2.6}$$

This formulation enables the student model to learn not only from the hard targets (i.e., ground-truth labels) but also from the richer information captured by the teacher's class-wise output distribution. This includes knowledge about class similarities, uncertainty, and inter-class structure that would otherwise be lost with standard supervised learning. Logits-based distillation serves as a common method in LLM compression and has been demonstrated through DistilBERT [74] and MiniLM [77] which modified the basic approach according to different architectural constraints and application scenarios.

Hint-based Distillation The logits-based distillation method only focuses on matching output distributions between teacher and student models but it does not capture the matching of complex internal representations found in large language models. The proposed solution to this limitation is *hint-based distillation* which is also known as *intermediate feature-based distillation*. The method enables the student model to align its internal layer-wise representations with those of the teacher so it can replicate the teacher's behavior at both the outcome and reasoning levels.

Formally, hint-based distillation works by aligning the *activations* (i.e., intermediate feature representations) of the student and teacher models at selected layers. Let $F^s \in \mathbb{R}^{H \times W \times C_s}$ and $F^t \in \mathbb{R}^{H \times W \times C_t}$ denote the intermediate feature maps (activations) of the student and teacher, respectively, where H and W represent spatial or sequence dimensions, and C_s and C_t denote the channel or embedding dimensions.

Since the student and teacher models often have different architectures, these representations may differ in dimensionality. To address this, a learnable transformation function $\phi : \mathbb{R}^{H \times W \times C_s} \rightarrow \mathbb{R}^{H \times W \times C_t}$ is applied to project the student's features into the same space as the teacher's. The hint-based distillation loss is then defined as the Mean Squared Error (MSE) between the transformed student features and the teacher features [73]:

$$\mathcal{L}_{\text{hint}} = \mathcal{H}(F^s, F^t) = \left\| F^t - \phi(F^s) \right\|_2^2 . \tag{2.7}$$

This additional loss encourages the student model to mimic not only the final outputs but also the teacher's internal processing steps, providing richer supervision signals during training. This approach was first formalized by Romero *et al.* [65] in the context of convolutional neural networks, introducing the notion of a learnable transformation function to match intermediate representations.

Several successful implementations of hint-based distillation have been proposed. *Patient Knowledge Distillation (PKD)* [83] introduces two learning strategies: PKD-Last and PKD-Skip, that enable the student to learn from the last k layers or every k-th layer of the teacher model. These strategies promote progressive, layer-wise learning and have been shown to improve performance.

$$\mathcal{L}_{\text{PKD}} = \sum_{i \in \mathcal{I}} \left\| H_i^t - \phi(H_i^s) \right\|_2^2 \tag{2.8}$$

where H_i^t and H_i^s denote hidden representations of the teacher and student at layer i, and $\mathcal{I}$ is the set of aligned layer indices.

The concept of hint-based distillation is further developed through *TinyBERT* [75] which adds multiple internal signals such as embeddings and hidden states and attention matrices. The student model learns to match both attention distributions and hidden state representations between corresponding layers during training. More recently, *Task-aware Layer-wise Distillation (TED)* [84] addresses the challenge that not all intermediate knowledge is relevant for a specific downstream task. TED introduces task-aware filters, learned during a separate training phase, to extract only task-relevant knowledge from intermediate teacher and student representations. By removing redundant or irrelevant information, TED helps more precise and targeted knowledge transfer, particularly in fine tuning or domain adaptation tasks.

In summary, hint-based distillation enables student models to benefit from intermediate guidance, ensuring that they reproduce not only the teacher's predictions but also its decision-making process. This fosters improved generalization, stability, and adaptability across various tasks.

2.3.5. Black-box Distillation Methods

While white-box distillation methods need complete access to internal representations of teacher models including hidden states, logits and architectural configurations, black-box distillation works without such access. Instead, the student model learns exclusively from the observable input-output behavior of the teacher model, which is treated as an opaque or *black-box* system. This has become increasingly important with the rise of proprietary LLMs like GPT-4, Claude and Gemini, which are often available only via limited API access.

Input–Output Response Distillation

Black-box distillation primarily relies on *Input–Output Response Distillation* as its fundamental method. The technique operates by studying only the teacher model's observable input–output responses, rather than using internal representations, which white-box methods typically employ. The student model duplicates teacher responses through this method while remaining unaware of the teacher model's internal parameters and architectural structure.

The distillation process begins with creating a prompt set that successfully represents the target domain or task. The student model receives training to produce teacher-like responses for identical input data over time. The training objective relies heavily on alignment metrics to determine the degree of output similarity between student and teacher responses. Currently, the field continues to use BLEU [85] and ROUGE [86] as its preferred metrics. However, the main drawback of these metrics is their restricted ability to detect n-gram overlaps at the surface level and their failure to recognize deeper semantic relationships. address this limitation. BERTScore [87] uses contextual embeddings to assess token similarity, which produces better semantic alignment results. Large language models have started to replace traditional evaluation tools in the field. GPT-4 represents a state-of-the-art model that can receive prompts to evaluate generated text through assessments of coherence and relevance, as well as factual accuracy.

In some cases, APIs expose token-level probability distributions over vocabulary tokens. While still lacking structural access to the model, this additional information allows for more precise supervision. This variant, often considered *grey-box* distillation, enables the use of standard cross-entropy loss to align the student's predictions with the teacher's soft output distribution:

$$L_{\text{output}} = \text{CE}(p_{\text{student}}, p_{\text{teacher}}) = -\sum_{i=1}^{N}\sum_{j=1}^{V} y_j^T \log(y_j^S) \tag{2.9}$$

where y_j^T and y_j^S denote the output probabilities of the teacher and student for token j, respectively, V is the vocabulary size, and N the sequence length. This approach improves fidelity to the teacher's behavior, but still lacks deeper supervision signals such as intermediate feature alignment.

Proxy-based Knowledge Distillation

Proxy-based Knowledge Distillation (Proxy-KD) introduces an intermediate white-box proxy model to mitigate the information loss in direct black-box distillation [88]. The process consists of:

1. Aligning the proxy model with the black-box teacher through output matching.
2. Performing white-box distillation from the aligned proxy to the student model.

The alignment is achieved by minimizing a combined objective:

$$L_{\text{proxy}} = \alpha \cdot L_{\text{proxy-nll}} + L_{\text{pref}} \tag{2.10}$$

where $L_{\text{proxy-nll}}$ is the negative log-likelihood loss and L_{pref} is a preference-based loss promoting high-quality generations. The student model is then trained using a weighted KL divergence loss:

$$L_{\text{student}} = L_{\text{Weight-KL}}(p_{\text{proxy}}, p_{\text{student}}) \tag{2.11}$$

Proxy-KD [88] has shown superior performance on complex reasoning tasks like BBH (BIG-Bench Hard)[89] and GSM8K [90] [88].

In-Context Learning Distillation

In-Context Learning (ICL) Distillation is designed to transfer the teacher model's ability to adapt to new tasks using only natural language demonstrations, without changing its internal parameters [91]. In this approach, the teacher model first receives prompts that include example demonstrations for a specific task (for instance, solving arithmetic word problems step by step). It then generates a response that not only provides the final answer but also shows its reasoning process in context. These input–output pairs, including the original prompt and the teacher's detailed response, are collected as training data. The student model is then fine-tuned on this data to reproduce both the reasoning steps and final answers. In this way, the student learns to imitate the teacher's in-context reasoning behavior directly, without needing explicit demonstrations at inference time. This strategy enables smaller models to acquire few-shot and zero-shot adaptation abilities more effectively, helping them generalize to new tasks with much less task-specific data [91], [92]. It has proven useful for tasks that benefit from step-by-step explanations, such as logical reasoning, math problems, and instruction following.

Instruction Tuning via Black-box Distillation

Instruction tuning via black-box distillation is a practical method for transferring a teacher model's instruction-following abilities to a smaller student model when only API access is available [78]. The main goal is to align the student's behavior with user expectations by learning how the teacher responds to various natural language instructions. In this setup, a diverse set of instructions is first collected to represent different user queries and tasks. These instructions are then sent to the black-box teacher model (for example, a proprietary API like ChatGPT) to generate corresponding high-quality responses. The resulting instruction-response pairs form a synthetic training dataset.

During fine-tuning, the student model is trained to reproduce the teacher's responses given the same instructions. This is typically done by minimizing the negative log-likelihood of the teacher's outputs:

$$L_{\text{inst}} = -\sum_{(i,r)\in D} \log P_{\text{student}}(r \mid i) \tag{2.12}$$

Aspect	**White-box Distillation**	**Black-box Distillation**
Access Requirements	Full access to teacher internals	API-level access only
Knowledge Transfer Fidelity	High (internal alignment)	Limited (output-only)
Implementation Complexity	Requires architectural compatibility	Model-agnostic
Practical Applicability	Limited to open-source/self-owned models	Works with proprietary models
Computational Efficiency	Often more efficient	May need more data

Table 2.1.: Comparison of white-box and black-box distillation methods

where D is the dataset of instructions i and teacher responses r.

To improve generalization and learning efficiency, advanced approaches like TAPIR [93] add a curriculum: they start with easier instructions and gradually introduce more complex tasks based on how well the student performs. This way, the student model learns robust instruction-following abilities even with limited access to the teacher's internals.

Chain-of-Thought Distillation

Chain-of-Thought (CoT) distillation aims to transfer not only the teacher model's final predictions but also its intermediate reasoning steps, often called rationales [69], [94]. This approach helps the student model learn to mimic the teacher's problem-solving process rather than just its outputs. The process works in two phases. First, the teacher is prompted to generate explicit step-by-step reasoning traces together with the final answers. Then, the student is trained to reproduce both the reasoning and the output simultaneously. This is typically achieved by minimizing a joint loss function:

$$L_{\text{CoT}} = \lambda_1 L_{\text{output}} + \lambda_2 L_{\text{reasoning}} \tag{2.13}$$

where λ_1 and λ_2 balance the emphasis between accurate predictions and faithful reasoning replication. Recent research shows that CoT distillation can significantly improve student performance on complex multi-step reasoning tasks such as BBH and GSM8K [69], [94].

The main distinctions between white-box and black-box distillation methods are presented in Table 2.1 which shows their access requirements, knowledge transfer fidelity, implementation complexity, and practical applicability.

2.3.6. Data Augmentation and Generation in Black-box Knowledge Distillation

This section investigates data augmentation and generation methods in knowledge distillation frameworks especially in black-box scenarios where the internal parameters of the teacher model are inaccessible. The most effective approach in these situations involves using large language models for data augmentation. An LLM can produce large amounts of high-quality task-specific examples through minimal seed information input. Xu *et al.* [16] present a four-stage pipeline for generating distillation data in black-box contexts. First, the target domain or skill is defined (e.g. reasoning capabilities or summarization) by crafting instructions or prompts that steer the teacher model toward that area of expertise. Next, a limited set of seed examples or clues is injected. These seeds act as minimal inputs that steer the teacher model to produce more elaborate responses aligned with the intended skill. In the third stage, the teacher processes these instructions and seeds to generate outputs. This outputs are often in the form of question–answer pairs, explanations, or textual completions, so that they reflect the teachers internal knowledge. Finally, these synthesized examples are parsed into training pairs and used to fine-tune the student model under an appropriate supervised objective.

Formally, the first three stages can be described as the knowledge elicitation process:

$$D_I^{(\mathrm{kd})} = \{\,\mathrm{Parse}(o, s) \mid o \sim p_T(o \mid I \oplus s),\ s \sim S\}, \tag{2.14}$$

where I represents the instruction or template used to guide the teacher, S denotes the distribution of seed knowledge, and $s \sim S$ is a sample drawn from that distribution. By concatenating I and s (written as $I \oplus s$) and querying the teacher model p_T, an output o is produced. The function $\mathrm{Parse}(o, s)$ then transforms this output, possibly combined with the seed, into a training example, such as pairing a prompt with a corresponding answer or explanation.

After collecting a set of such distillation examples $D_I^{(\mathrm{kd})}$, potentially covering multiple instructions I, the student model is trained by minimizing the total loss

$$\mathcal{L} = \sum_I \mathcal{L}_I(D_I^{(\mathrm{kd})};\, \theta_S), \tag{2.15}$$

where $\mathcal{L}_I$ is the task- or skill-specific objective (such as cross-entropy or a ranking loss) associated with instruction I, and θ_S denotes the parameters of the student model [16]. In this way, the student learns to reproduce both the surface-level patterns and the underlying semantic capabilities present in the teacher's outputs, without requiring access to the teacher's internal representations.

Data Augmentation Strategies

In black-box knowledge distillation, five strategies for generating or augmenting data have emerged: labeling, expansion, data curation, feedback methods, and self-knowledge generation [16] (Figure 2.6). Each strategy differs in how it elicits or refines training examples, but all ultimately aim to produce high-quality (x, y) pairs (or (x, y, ϕ) tuples, when feedback signals are included) that allow the student to approximate the teacher's performance.

Labeling is perhaps the simplest approach and can also be seen as Input-Output Distillation (described in 2.3.5): given an input x drawn from some distribution X, the teacher is prompted with an instruction I (and optionally a few demonstrations c) to produce an output y. Formally, the resulting distillation set can be written as [16]

$$D^{(\mathrm{lab})} \;=\; \{\,(x, y) \mid x \sim X,\ y \sim p_T(y \mid I \oplus c \oplus x)\}. \tag{2.16}$$

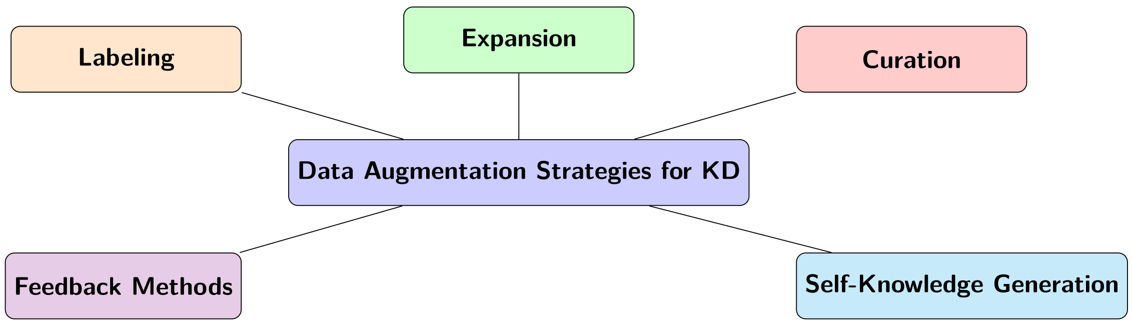
Labeling
Expansion
Curation
Data Augmentation Strategies for KD
Feedback Methods
Self-Knowledge Generation

Figure 2.6.: Data Augmentation Strategies for KD

In practice, x may be drawn from existing NLP benchmarks such as classification or summarization datasets, or it can come from real user queries like ShareGPT conversations. The instruction I specifies how the teacher should produce its responses, for example by prompting it to explain its reasoning step by step in the case of chain-of-thought tasks. The demonstration set c typically includes a small number of (x_i, y_i) examples that help define the desired style or level of difficulty. By collecting (x, y) pairs in this way, one creates synthetic supervision data that can be used to train the student model. Many recent systems, including FLAN [95], Alpaca [96], and CodeAlpaca [97], make extensive use of this strategy: the teacher model labels the inputs with matching answers, explanations, or code, which allows its reasoning patterns and instruction-following skills to be transferred to a smaller, more efficient student network.

Expansion methods generate both inputs and outputs from a small seed of demonstrations, rather than relying on an existing input set. In other words, starting from a handful of example pairs c, one first prompts the teacher to produce a new input x, and then immediately asks it to generate the corresponding output y. The formal description of this procedure is [16]:

$$D^{(\mathrm{exp})} = \Big\{ (x, y) \mid x \sim p_T(x \mid I \oplus c),\ y \sim p_T(y \mid I \oplus x) \Big\}. \tag{2.17}$$

This strategy makes use of the LLM's in-context learning ability to bootstrap additional data. For example, Self-Instruct [98] starts with a small collection of human-written instruction–output pairs (x, y) that prompt the teacher to create new instructions. These newly generated instructions are then reused in multiple rounds, producing a much larger and more varied set of machine-labeled instruction–output examples. Alpaca [96] applied a similar two-step expansion process: it first generated instructions x and then the corresponding responses y, ultimately collecting over fifty thousand instruction–response pairs from *text-davinci-003*. Follow-up work like Evol-Instruct [99] showed that asking the LLM to vary the difficulty and subject matter of the generated instructions can further expand the coverage of the resulting dataset. However, the teacher's biases can limit this expansion: if the seed examples c are too narrow or the teacher's outputs become repetitive, the final dataset may lack diversity, which calls for careful filtering or several refinement cycles.

Data Curation seeks finer-grained control over both inputs and outputs by relying on explicit meta-information m rather than only on examples. One provides the teacher with an instruction I and a metadata tag m, which encodes a topic (e.g., "legal contracts"), a difficulty level, or a required subskill, and asks the model to produce an input x fitting to m. The teacher then generates the corresponding output y. Formally,

$$D^{(\mathrm{cur})} = \Big\{ (x, y) \mid x \sim p_T(x \mid I \oplus m),\ y \sim p_T(y \mid I \oplus x) \Big\}. \tag{2.18}$$

Because m can be drawn from a large bank of topics, function names, or knowledge points, this approach can yield highly targeted, "textbook-quality" data [16]. For example, UltraChat [100] collects metadata across domains like "Technology" and "Health", then guides GPT-4 to create over 1.5 million multi-turn dialogues spanning those topics; the distilled UltraLLaMA model subsequently outperforms many larger open-source LLMs. In the coding domain, the Phi series [101] picks random function names or programming concepts as m, prompting the teacher to generate clear, self-contained exercises along with solutions. Even though curation requires more upfront design, because one must compile relevant metadata, it often produces datasets that are both cleaner and more diverse than simple expansion.

Feedback Methods go beyond simple one-way imitation by adding a second loop in which the teacher reviews and critiques the student's outputs. In practice, the student first generates an answer y for a given input x. The teacher is then asked to provide a feedback signal $\phi_{\mathrm{fb}}(x, y)$ based on the quality of the student's response. This creates a feedback dataset of the form [16]:

$$D^{(\mathrm{fb})} = \{ (x,\ y,\ \phi_{\mathrm{fb}}(x, y; \theta_T)) \mid x \sim X,\ y \sim p_S(y \mid x) \}. \tag{2.19}$$

In practice, ϕ_{fb} can take different forms. For example, the teacher may provide preferences by indicating which of two answers is better, or offer detailed step-by-step critiques that highlight errors in a generated solution, such as marking mistakes in a mathematical proof. RLAIF [29] and UltraFeedback [102] generate pairs of candidate responses and rely on the teacher to rank them based on helpfulness, factual accuracy, or consistency with human values. Other approaches extend this idea by identifying inputs where the student performs poorly and then synthesizing more challenging or corrective examples to improve the student's robustness and reasoning capabilities.

Self-Knowledge Generation Self-knowledge generation lets the student create its own outputs and then evaluate them without needing an external teacher. In practice, the student generates an output y for a given input x and applies a self-evaluation function $\phi_{\mathrm{sk}}(x, y)$, which could be a simple heuristic or an internal reward model. Only the (x, y) pairs that pass a certain quality threshold are kept. In notation [16],

$$D^{(\mathrm{sk})} = \{ (x, y, \phi_{\mathrm{sk}}(x, y)) \mid x \sim S,\ y \sim p_S(y \mid I \oplus x) \}. \tag{2.20}$$

Self-knowledge approaches can range from simple filtering methods, such as keeping only summaries that meet certain length or format criteria, to more advanced strategies where the model generates additional instructions and responses for its own training. Since these approaches do not depend on a proprietary teacher, they offer greater autonomy for the student model. However, they may require more training cycles to reach strong performance, especially if the student starts with limited initial capabilities.

2.3.7. Distillation Learning Objectives in Black-Box Settings

Once a sufficiently large and diverse set of training examples has been generated, whether through labeling, expansion, curation, feedback, or self-knowledge, the final step is to select an appropriate learning objective for the student. In black-box distillation, the teacher's logits and hidden representations are unavailable, making direct KL-divergence minimization on full distributions impossible. So the most common objectives include:

Supervised Fine-Tuning (SFT) trains the student model by treating each generated pair (x, y) as a true input-output example and minimizing the cross-entropy loss

$$\mathcal{L}_{\text{SFT}} = -\sum_{(x,y)\in D} \log p_S(y \mid x). \tag{2.21}$$

This objective is used whenever the teacher provides clear target outputs. It is one of the most common approaches for learning from synthetic data because it directly aligns the student's predictions with the teacher's examples [15].

Reinforcement Learning from AI Feedback (RLAIF) Instead of merely imitating the teacher's exact outputs, Reinforcement Learning from AI Feedback (RLAIF) fine-tunes the student model based on the teacher's *judgment*. This is a two-step process. First, a separate "reward model" is trained to understand the teacher's preferences (e.g., which of two responses is better). Then, the student model is trained to generate responses that would receive a high score from this reward model. To ensure the student does not become repetitive or unstable, the training process includes a penalty that discourages the student from deviating too far from its original, natural-sounding language patterns. This method aligns the student's behavior with the teacher's nuanced preferences, rather than just its exact words.

Ranking-Based Objectives As a more direct and often more stable alternative, Ranking-Based Objectives train the student to simply distinguish between "good" and "bad" responses. This approach, which is the foundation of modern methods like *Direct Preference Optimization (DPO)* [103], works with preference pairs where the teacher has labeled one response as better than another. The training objective is then straightforward: it directly adjusts the student model to increase the probability of generating the preferred response while decreasing the probability of the dispreferred one. This method is highly efficient because it bypasses the need to create and train a separate reward model, teaching the student about preferences directly.

Hybrid Objectives In many state-of-the-art pipelines, supervised fine-tuning is performed on generated (x, y) pairs. This is followed by applying either RLAIF or ranking optimization to further refine the student model. This two-stage procedure uses the strengths of both imitation learning (to acquire basic skills) and preference alignment (to refine the model), resulting in students that closely approximate the teacher's performance in both raw accuracy and alignment with user intents.

3. DLR Context and Use Cases

This chapter provides an overview of the DLR and its engagement with foundation models such as Large Language Models. It introduces the organizational context, outlines current challenges and opportunities in adapting such models, and presents selected use cases that highlight the potential and requirements for deploying LLMs.

3.1. Overview of the German Aerospace Center

As described in the introduction, the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt, DLR) functions as Germany's national research institution dedicated to aerospace together with energy, transport and digital topics. DLR operates through more than 30 locations while using multiple research institutes to perform both fundamental and applied research [13].

DLR functions as the main force that drives digital transformation in aerospace and transport system development. The organization has expanded its research of artificial intelligence technologies during the past few years. The models deliver significant benefits for automation and human-machine interaction as well as data interpretation and decision-making support throughout DLR's application domains. The DLR initiated the DLR Foundation Models Project as an interdisciplinary initiative to assess and modify open-source foundation models before deploying them across different applications.

The project addresses the pressing need for domain-specific, secure, and resource-efficient AI solutions. Foundation models provided by commercial companies, operated through cloud services, have restricted customization options and there could be privacy issues when processing sensitive research or mission data. DLR plans to implement foundation models on its own infrastructure and also performing local fine-tuning and modular integration of data sources. The project enables DLR institutes to work together while linking research needs to operational requirements and developing standard methods for generative AI throughout the organization.

3.2. Foundation Models at DLR: Challenges and Opportunities

The DLR can bring their research projects further through the implementation of Foundation Models which include large language and vision-language models. Foundation Models demonstrate broad generalization abilities and are adaptable to a wide range of tasks (Section 2.1.1).

Their implementation within DLR has major implementation hurdles (Section 2.1.2. The main barrier stems from the high resource requirements of these models. The large models need large computational infrastructure to operate and are usually available only through commercial cloud platforms. The cloud usage of these models creates privacy issues with data. Local deployment creates high operational costs. The majority of DLR use cases presented in the project are using domain-specific sensitive data (e.g. mission telemetry or aerospace protocols) so therefore they demand local solutions.

The DLR Foundation Models Project aims to deploy open-source models on internal infrastructure which enables data sovereignty alongside additional fine-tuning of models. The development of lightweight alternatives through knowledge distillation, model pruning and quantization techniques aims to support applications which have limited computational resources and strict latency or energy constraints.

3.3. Use Cases in the DLR Foundation Models Project

The participating use cases, which include autonomous vehicles, space robotics, aviation, and maritime traffic management, share a common interest in foundation models but differ significantly in their operational constraints, data modalities, and safety-critical requirements. This section provides a brief overview of these diverse applications to highlight the range of challenges they pose and to clarify the specific focus of this thesis on the autonomous vehicle domain.

Autonomous Vehicles and Human–AI Interaction (TS)

The "vStop" scenario represents the Transportation Systems (TS) pilot use case within the DLR Foundation-Models project. Its objective is to realise a voice-based human–vehicle interface that converts free-form passenger utterances, together with real-time environment perception, into executable manoeuvre commands for a highly automated vehicle. Figure 3.1 summarises the end-to-end information flow; the individual processing stages and their implications for knowledge-distilled LLMs are outlined below.

Problem context. Passengers of autonomous vehicles frequently issue ambiguous, context-dependent requests such as "Please stop here on the left, immediately" or "Slow down and move into the left lane". A robust dialogue system must (i) comprehend natural language with colloquial variations, (ii) fuse linguistic intent with situational context (current speed, distance to vehicle ahead ...), and (iii) return a machine-readable command that can be validated and executed by the vehicle's tactical planner. All components must run locally on the in-vehicle embedded platform, where compute, memory and energy budgets are tightly constrained by the automotive hardware and where real-time deadlines and functional-safety requirements exist.

System architecture. Figure 3.1 illustrates the modular information flow from passenger input to vehicle control. Spoken commands are captured via onboard microphones and transcribed by a compact, locally deployed speech recognition model. The system combines the generated text with system data from the vehicle about the current speed, distance to the goal, and distance to the vehicle ahead and creates a structured prompt. This prompt is processed by a (knowledge-distilled) LLM, fine-tuned for vehicle-control dialogue, which emits either a structured JSON command or a rejection. This output is passed to the vehicle control system for execution or clarification.

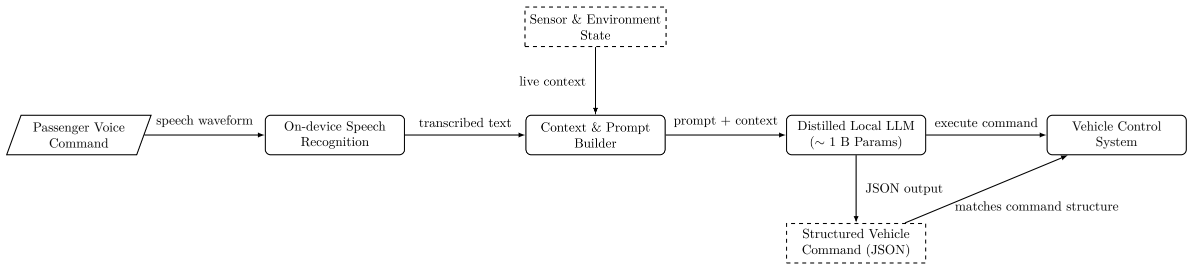

Figure 3.1.: System architecture of the TS voice–command pipeline. Passenger speech is transcribed locally, fused with real-time sensor context, and converted by a knowledge-distilled LLM into a structured JSON command that is forwarded to the vehicle control system for execution.

A safety-filter is also partly integrated: the distilled LLM must either (i) produce a command that is provably safe under the supplied context, or (ii) return a structured rejection with a reason code (e.g. `unknown_command`). This "speak-or-refuse" behaviour follows recent recommendations for LLM deployment in safety-critical domains [104]–[107].

Synthetic data generation for domain adaptation. No corpus exists that links voice commands to structured driving manoeuvres for this specific vehicle system (to fine-tune a model for this task). Accordingly, a two-stage pipeline was implemented (Section 4.3.2):

1. **Scenario sampling.** Vehicle-centred states (speed, distance ...) and target manoeuvre classes (e.g. `stopPark`, `change_lane`) are drawn from empirically motivated distributions. Rule-based validators mark each tuple as *positive*, *negative* (unsafe), or *unknown* (out-of-schema).
2. **Utterance synthesis.** A large teacher model (gemma3:27b) is prompted to paraphrase each scenario into diverse user commands (imperative, polite, indirect). The result is a paired dataset:

$$\text{(user command, context)} \rightarrow \text{structured\_command or rejection}$$

   comprising $\approx 30\,\text{k}$ positives, $10\,\text{k}$ negatives and $1\,\text{k}$ unknowns.

This corpus is employed for (i) supervised fine-tuning of the teacher on the target output format, and (ii) subsequent knowledge distillation into a smaller student model.

Role of Knowledge Distillation. Initial profiling shows that for example using directly a bigger model as a translator (e.g. LLAMA 8B), requires $\approx 21\,\text{GB}$ of VRAM without additional compression techniques during inference (shown in Section 4.8.3). This is far beyond the capabilities of current automotive ECUs. By contrast, the distilled 1B student fits into $< 5\,\text{GB}$, reduces average latency by a factor ≈ 4–5, Distillation therefore provides the decisive leverage to enable on-board deployment without off-loading to cloud infrastructure, eliminating bandwidth, privacy and availability concerns.

Research relevance. The TS scenario aligns with emerging literature that frames planning and control as language-conditioned reasoning tasks (e.g. DriveGPT-4 [107], GPT-Driver [108], Agent-Driver [109]) and highlights the need for compact, domain-aligned LLMs in autonomous driving. By focussing on voice-to-command translation under stringent compute budgets, this use case serves as a realistic example for evaluating KD techniques. In summary, the TS use case provides (i) a clearly defined transformation task, (ii) an end-to-end pipeline that can be quantitatively analyzed, and (iii) operational constraints that make model compression essential. Together, these factors make it an effective experimental test-bed for the knowledge distillation framework proposed in this work.

Other use cases inside the DLR project for future work

Space Robotics and Onboard Assistance Systems (RM)

DLR's space program demonstrates the potential of Large Language Models to help with autonomy and human-machine collaboration in extraterrestrial missions through the "ActGPT" and "METIS" use cases. The ActGPT scenario requires LLMs to convert high-level astronaut commands into executable action sequences for planetary surface robots. The models need to understand complex ambiguous instructions while evaluating environmental conditions to create physically feasible behavior plans which creates substantial challenges regarding safety and transparency and multimodal reasoning. The system focuses heavily on explainability because it needs to provide natural language explanations for its decision-making process. The METIS project investigates how LLMs can assist spacecraft crews in autonomous operations during Mars missions with delayed ground communication. The evaluation of LLMs takes place as a potential replacement for knowledge-graph-based case-based reasoning (CBR) methods. The ESA Protection Level 2 classification of mission data demands local model deployment alongside strict data governance which strengthens the need for lightweight privacy-compliant models that remain controllable.

Earth Observation and Multimodal Scene Understanding (MF)

The Earth observation use case aims to utilize LLMs and Visual Language Models (VLMs) to enhance satellite data interpretation and accessibility through semantic analysis. The models are used for three tasks: generating textual metadata from hyperspectral and SAR imagery, natural language-based image search and free-form user queries for scene interpretation. The combination of text and visual elements should enable users to explore datasets through human-friendly interfaces that would benefit urgent applications including disaster response and environmental monitoring.

Aviation and Intelligent Cockpit Assistance (FL)

The FOCUS project extension in the aviation sector evaluates how LLMs can be implemented in cockpit systems to enable future operational concepts including *Extended Minimum Crew Operations (eMCO)* and *Single Pilot Operations (SiPO)*. The operational requirements of these settings demand advanced automation assistance and pilot confidence systems. The proposed AI assistant combines voice command comprehension with technical documentation retrieval through RAG methods and decision support during crucial flight phases. The implementation of aviation-specific terminology processing for spoken language represents a major technical hurdle. The assistant needs to perform factual question answering while generating context-specific validated output.

Maritime Traffic Management and Explainable Recommendations (SE)

The maritime use case operates under the *NextGen-VTS* initiative to enhance *vessel traffic services (VTS)* through LLM-based explanation and interaction capabilities. The project aims to establish proactive AI-supported traffic coordination. The system should employ LLMs to analyze and explain optimization results for the maritime traffic management produced by machine learning models or rule-based systems. The LLM needs to explain rerouting recommendations in natural language while responding to the ship operator inquiries and providing alternative solutions when original recommendations become impractical. The system should improve transparency, trust levels and user acceptance rates. The model needs to know maritime terminology and operational scenarios.

3.4. Requirements for Deploying LLMs in DLR Applications

DLR applications need to implement LLMs that fulfill multiple requirements because of their mission-critical nature and resource limitations and data sensitivity needs. The use cases require models which function under restricted computational power while maintaining low latency and conserving energy. The creation of lightweight models requires model compression techniques including knowledge distillation, pruning and quantization. The processing of classified or sensitive data including mission telemetry, satellite imagery, and operational protocols needs internal deployment of LLMs to ensure data sovereignty and privacy. Cloud-based solutions are generally unsuitable due to the risk of data exposure and limited control over model behavior. Multiple application face time constraints. The DLR Foundation Models Project works to overcome these challenges by developing shared infrastructure and methods to deploy foundation models across different domains while maintaining control, efficiency and domain relevance.

4. Methodology and Experimental Framework

The chapter describes the experimental setup and methodology which was used to evaluate knowledge distillation as a method for deploying large language models at the DLR. The research pipeline is described in full detail, including the datasets, model architectures, and the specific implementation of the distillation framework. Here also the metrics and baselines are defined used for a comparative analysis of the resulting models, ensuring that the findings are both robust and reproducible

4.1. Research Questions and Hypotheses

The primary objective of this thesis is to investigate and validate Knowledge Distillation as a method for adapting powerful but resource-intensive LLMs for practical use in the domain-specific, computationally constrained environments of the DLR. The central research question guiding this work is:

> *How can large language models be efficiently adapted for use in resource-constrained, domain-specific environments at DLR through knowledge distillation without significant losses in accuracy and reliability?*

To address this question, this research is built upon three core hypotheses that will be empirically tested through the experimental framework detailed in this chapter. The first of these is the *performance recovery hypothesis*, which posits that knowledge distillation can effectively transfer the nuanced, task-specific capabilities of a large teacher LLM to a significantly smaller student model. It is hypothesized that the resulting distilled student model will substantially outperform a baseline student of the same size that is trained only on ground-truth labels, thereby recovering a majority of the performance lost due to model compression. Building on this, the *efficiency gain hypothesis* suggests that this performance recovery will be accompanied by significant gains in computational efficiency. Specifically, the distilled student model is expected to have smaller memory requirements and lower inference latency, rendering it viable for deployment on the embedded and edge-computing hardware used in DLR applications. Finally, the *domain-specific adaptation hypothesis* proposes that a synthetic data generation pipeline, leveraging a teacher model to create realistic and safety-aware training examples, can effectively adapt the distilled model to a specialized domain. This approach is hypothesized to enable the student model to achieve high performance on a niche DLR task—such as autonomous vehicle command interpretation—for which large-scale, human-annotated datasets are not available. The successful validation of these hypotheses would demonstrate that knowledge distillation is a critical enabling technology for deploying advanced AI capabilities in real-world, safety-critical aerospace and autonomous systems.

4.2. Overall Experimental Design

To systematically investigate the research questions, a multi-stage experimental design was implemented. The workflow enables direct evaluation between a large-scale teacher model and both the baseline student model and the student model that received knowledge distillation enhancements. The end-to-end pipeline follows a conceptual design as shown in Figure 4.1 and includes four essential stages.

The process began with the selection and preparation of the models. A powerful, pre-trained LLaMA 3.1 8B model was selected as the teacher due to its high performance on complex reasoning tasks. A significantly more compact LLaMA 3.2 1B model was chosen as the student. Subsequently, for dataset curation and generation, the models were evaluated on two distinct corpora. The first was the public *xLAM Function Calling dataset* [110], which served as a general-domain benchmark. The second was a novel, domain-specific synthetic dataset generated to simulate the autonomous vehicle voice-command use case at DLR. This custom pipeline utilized the teacher model to generate realistic natural language queries paired with structured commands and rule-based safety labels.

The cornerstone of the methodology was the knowledge distillation process itself. In this core experimental stage, the 1B student model was trained to mimic the output behavior of the 8B teacher model. This was achieved using a logit-based distillation approach, where the student is trained on a composite loss function combining standard cross-entropy with the Kullback-Leibler divergence loss from the teacher's softened probability distributions. The final stage comprised a comprehensive evaluation, where the performance of the distilled student model was rigorously benchmarked against the teacher and a baseline student (trained without distillation). The evaluation protocol was designed to measure both task performance (e.g., function accuracy, F1-score) and computational efficiency (e.g., inference latency, memory footprint), providing a holistic view of the trade-offs involved.

This structured approach ensures that the experimental results are directly comparable and provide clear, empirical evidence to address the core research hypotheses. The key experimental conditions are summarized in Table 4.1, which outlines the distinct roles and training methodologies for the primary models compared in this study.

4.3. Datasets and Domain Adaptation

The performance and generalization capabilities of any machine learning model are fundamentally shaped by the data upon which it is trained and evaluated. To ensure a robust and comprehensive assessment of the knowledge distillation framework, this work employs a dual-dataset strategy. The approach combines a widely-used public benchmark to evaluate the model's general function-calling competence with a novel, domain-specific synthetic dataset designed to test its adaptability and performance within the unique operational context of the DLR. This section details the characteristics of both datasets and the pipeline developed to generate the latter.

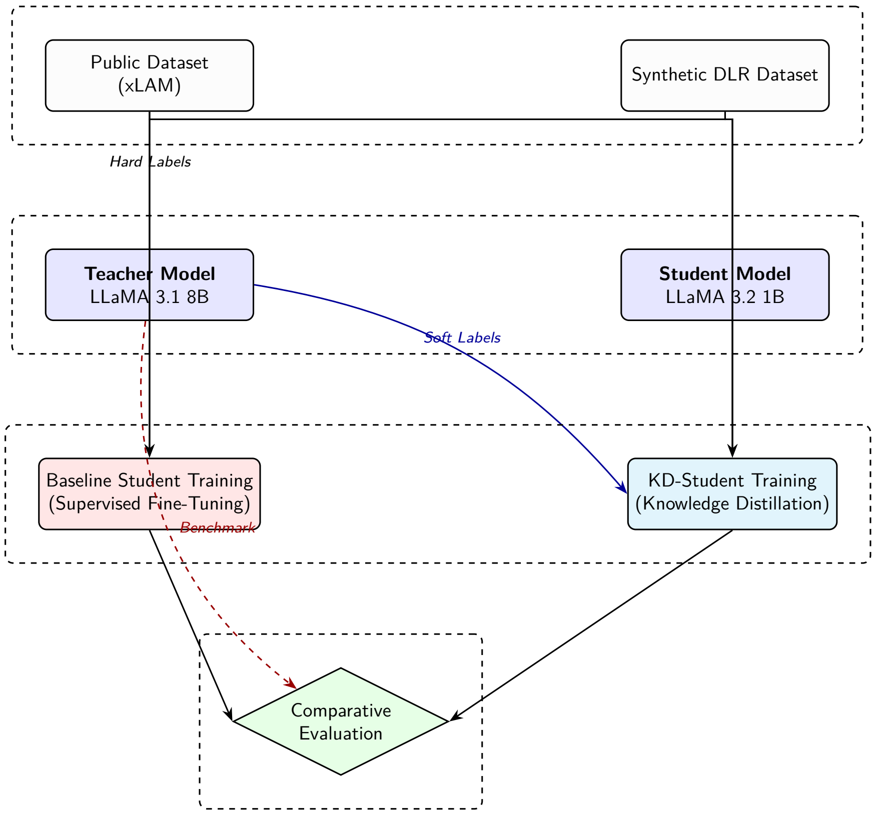
Public Dataset (xLAM)
Synthetic DLR Dataset
Hard Labels
Teacher Model LLaMA 3.1 8B
Student Model LLaMA 3.2 1B
Soft Labels
Baseline Student Training (Supervised Fine-Tuning)
Benchmark
KD-Student Training (Knowledge Distillation)
Comparative Evaluation

Figure 4.1.: Overview of the experimental design. The pipeline begins with data curation from both public and synthetic sources. A student model is then trained in two configurations: a baseline fine-tuned on hard labels and a KD-student trained to mimic the teacher's soft labels. Finally, all models are evaluated to compare task performance and computational efficiency.

Table 4.1.: Summary of experimental model configurations. Each model serves a distinct role in the comparative evaluation.

Role	Model Name	Params	Training Method	Key Purpose
Teacher	LLaMA 3.1	8B	N/A (Inference only)	Provide soft labels for distillation and establish the performance upper bound.
Baseline St.	LLaMA 3.2	1B	Supervised Fine-Tuning	Establish a performance baseline for a compact model trained without knowledge distillation.
KD-Student	LLaMA 3.2	1B	Knowledge Distillation	The primary experimental model, trained to measure the efficacy of knowledge distillation.

4.3.1. Public Benchmark: xLAM Function Calling

A standardized benchmark must be established before evaluating the knowledge distillation framework on a specialized synthetically generated dataset. The framework's core performance needs to be evaluated through standardized open-domain tasks which provide transparent and reproducible measures that isolate the distillation process from custom dataset characteristics. The model's knowledge retention after compression receives rigorous testing in a generalizable context through this method. The thesis establishes credibility for its domain-specific analyses through standard validation which strengthens the overall research findings. The Salesforce *xLAM Function Calling Dataset* [110] served as the chosen evaluation platform. The large-scale publicly accessible benchmark assesses language models for their ability to function as agents that interact with external tools. The examples in this dataset include natural language user queries together with JSON schema definitions for available functions and machine-readable function calls as expected outputs. The xLAM benchmark is particularly well-suited for this study for several reasons. First, its core task of translating unstructured natural language into structured, executable commands closely matches the main challenge in the DLR autonomous vehicle use case, providing a robust testbed for the capabilities targeted for compression and deployment in this work. The model must not only infer user intent but also map that intent to the correct function within a given schema and accurately extract all necessary parameters from the query. This level of complexity makes xLAM an excellent benchmark for evaluating how well a model's sophisticated reasoning abilities are preserved after distillation. In addition, as a well-structured and widely recognized dataset, xLAM provides a standardized environment for clear, empirical comparisons between the teacher, baseline student, and distilled student models. Demonstrating the effectiveness of the distillation framework on this general-purpose benchmark allows for a more confident assessment of its performance on the more specialized synthetic DLR dataset described in the following section. This two-pronged evaluation strategy ensures that the findings are both broadly applicable and directly relevant to the specific challenges associated with the DLR use case.

4.3.2. Domain-Specific Synthetic Dataset for DLR Use Case

A core methodological contribution of this thesis is the development of a domain-specific synthetic dataset tailored to the unique operational requirements of the DLR. While large, open-source datasets provide broad linguistic coverage, they lack the specificity needed for the specialized, safety-critical applications at DLR. This data gap is particularly evident in the Transportation Systems (TS) use case, which explores the human-machine interface for autonomous vehicles.

The system requires reliable interpretation of diverse natural language instructions from passengers which include formal commands and colloquial or indirect requests. The system needs to convert user intent into machine-readable commands that match the vehicle's command system structure. The translation process needs to occur within real-time sensor data integration of vehicle speed and obstacle proximity to perform safe and appropriate command execution. The absence of real-world annotated dialogue data for this domain required the development of synthetic data as an essential necessity. The main difficulty surpasses basic data scarcity because it demands effective handling and rejection of faulty or ambiguous or unsafe inputs. Real-world autonomous system models need to translate commands accurately while simultaneously refusing to execute instructions that are uninterpretable or dangerous to safety. The essential negative cases required for operational safety are not properly addressed by standard language model benchmarks. The synthetic dataset incorporated multiple essential design principles to directly tackle these challenges which resulted in a rigorous training and evaluation foundation for safety-aware models. The development of the synthetic dataset was guided by several essential design principles intended to address the specific needs of this research.

A fundamental consideration was the pursuit of contextual realism and comprehensive functional coverage. To this end, each data instance was grounded in an operational scenario generated from plausible, domain-specific distributions for sensor values, such as vehicle speed or object proximity. These realistic contexts were then paired with a wide-ranging fixed catalog of control actions, encompassing lane keeping, turning, parking, and various anomaly-handling maneuvers. This ensures the resulting dataset adequately represents the spectrum of conditions and system capabilities relevant to the use case. Furthermore, a crucial aspect of the design was the incorporation of security-aware supervision. A rule-based validation layer was integrated into the generation process to classify commands as either valid or invalid. This classification identifies actions that would be contextually impermissible or dangerous, thereby systematically embedding both positive and negative examples into the data. The presence of these safety-related cases is vital for the training and evaluation of models intended to exhibit robust rejection behavior. The pipeline was designed to support linguistic diversity through LLM-assisted generation, enabling the system to handle variability in human expression. The LLM paraphrased structured scenarios to produce a wide range of queries that included formal, colloquial, and ambiguous linguistic styles. This process prepares the distilled models to manage the inherent unpredictability of natural language input. The entire data generation pipeline was also designed to be fully automated and scalable to ensure reproducibility. The programmatic nature of the system allows for the creation of distinct training, validation, and test splits in a controlled and consistent way. This level of automation supports both experimental reproducibility and systematic performance evaluation of models within specific sub-regions of the functional space.

The synthetic dataset serves as a methodological solution that goes beyond simply addressing data scarcity. The combination of realistic simulation, safety-focused annotation, and linguistic diversity provides a solid foundation for studying knowledge distillation techniques and evaluating domain-specific model adaptations within an authentic operational context.

Synthetic Data Pipeline Architecture

The synthetic data generation pipeline implemented in this thesis enables the construction of realistic, semantically annotated, and structurally rich datasets tailored for the DLR's autonomous system interfaces. It comprises two principal components: (i) a structured parameter and context sampling module, and (ii) a language model-based natural language generation component. This separation ensures both control over operational plausibility and flexibility in linguistic variation. The implementation is modular and reproducible, and the complete codebase is made available in the accompanying code folder (see `/src/synthetic-data-generation/`).

Structured Parameter and Context Sampling The first stage of the pipeline is responsible for producing context-action pairs that are plausible within the DLR use case environment. This is achieved through stochastic yet constrained sampling routines implemented in python code. Each sample is initialized by randomly selecting a scenario type (urban, suburban, highway) based on pre-defined priors. Subsequently, sensor parameters such as vehicle speed, inter-vehicle distance, and goal proximity are drawn from empirically motivated distributions—primarily truncated normal and log-normal distributions—to ensure that the values remain within realistic operational boundaries. These distributions are defined and sampled in dedicated functions such as `_draw_speed_kmh()` and `_draw_distance_ahead()`.

After the context is sampled, a command is selected from a predefined schema of autonomous vehicle functions. These include maneuvers like `change_lane_left`, `turn_right`, `stop_board`, and `perform_u_turn`. Associated parameters such as target speed, urgency, or exit number are instantiated through scenario-aware samplers implemented in `generate_command_parameters()`. The full command-action space is defined in a central schema dictionary (`COMMAND_SCHEMAS`), allowing for flexible extension and task-specific customization.

To enforce operational safety and realism, the pipeline applies a rule-based validation layer implemented in `is_command_safe()`. This function codifies domain-specific safety constraints, such as legal speed limits, safe boarding conditions, and minimal time-to-collision requirements. Samples that pass all constraints are labeled as *positive*, while those violating any rule are marked *negative* and supplemented with an explanatory `rejection_reason`. Additionally, *unknown* samples are synthesized to model inputs that do not correspond to any valid command. The result is a dataset of machine-readable context-command tuples labeled for classification and rejection training tasks.

Natural Language Query Generation via LLMs The second pipeline stage augments each structured command-context pair with a realistic, linguistically diverse natural language query. This is achieved through prompt-based generation using large language models. The query generation process transforms structured data into passenger-style instructions, simulating real-world human-machine interaction.

Prompt templates are defined in `prompts.py` and include imperative, interrogative, and colloquial styles. For each structured sample, a prompt is dynamically assembled and passed to the LLM backend using `generate_chat_completion()`. The system supports both cloud-based endpoints (e.g., OpenAI GPT-4) and local secure inference via the Ollama interface, ensuring reproducibility across infrastructure. The generated queries reflect the variability and ambiguity typical of real-world user inputs, which is critical for evaluating generalization. Negative and unknown samples are deliberately paired with prompts designed to provoke infeasible or ambiguous requests. For instance, an unsafe manoeuvre might be expressed through a query like "Can I perform a U-turn?" when the current speed sensor indicates 80 km/h. Such examples are essential for fostering robust rejection behaviour in the distilled models, as they expose the system to edge cases that help it learn to identify and decline inappropriate actions reliably.

The final output of this stage is a structured JSON or JSONL file containing: (i) a unique identifier, (ii) sensor context, (iii) structured command and parameters, (iv) the generated natural language query, and (v) the categorical validity label. These outputs are then used by downstream tokenization and fine-tuning components for knowledge distillation training and evaluation.

Technical Rationale and Integration The way the data generation pipeline is structured is key to why it works well. It is split into two main parts: first, a rule-based system creates realistic driving scenarios, and second, a language model writes natural-sounding user requests for those scenarios. Separating these steps ensures the situations are realistic while also providing enough variety in language to train a robust model. This design allows for the creation of large, task-specific datasets without needing to use sensitive real-world data. Importantly, by combining rule-based safety checks with LLM-generated text, the pipeline provides a structured way to create examples of valid, unsafe, and irrelevant user commands, which is essential for training a safety-aware model.

Example Synthetic Sample

To make this two-stage generation process concrete, the following is an example of a single data sample. The first part shows the output of the structured scenario sampling, where the context and a candidate command are created according to safety rules. The second part shows the result of the natural language query generation, where a large language model produces a realistic user request for that scenario.

Positive Sample

```json
{
  "id": "de933eec-447e-4595-83e7-a06c49bce41c",
  "timestamp": "2025-06-01T23:18:54.558074+00:00",
  "sensor_inputs": {
    "scenario": "urban",
    "current_speed": 18.3,
    "distance_to_vehicle_ahead": 11.2,
    "distance_to_goal": 1813.1
  },
  "command": "stop_park",
  "parameters": {
    "parking_type": "parallel"
  },
  "label": "positive"
}
```

Listing 4.1: Example JSON - Positive Sample -Step 1 – Structured Sampling

Here, the vehicle is moving slowly in an urban setting, a valid parallel-parking spot is detected, and the command "stop_park" with `parking_type: parallel` passes all safety checks.

```json
{
  "id": "de933eec-447e-4595-83e7-a06c49bce41c",
  "timestamp": "2025-06-01T23:18:54.558074+00:00",
  "sensor_inputs": {...},
  "command": "stop_park",
  "parameters": {...},
  "label": "positive",
  "generated_query": "Oh, actually, could you try and parallel park up ahead when
        you get a chance? There seems to be a spot opening up."
}
```

Listing 4.2: Example JSON - Positive Sample - Step 2 – LLM-Generated Query

The generated query (synthetic data) uses a polite, conversational tone ("Oh, actually, could you. . . ") to mirror real passenger speech.

Unknown-Command Sample

```
{
  "id": "cb0cb691-94a8-4a93-bbf2-5b2af2801f74",
  "timestamp": "2025-06-01T23:18:55.337631+00:00",
  "sensor_inputs": {
    "scenario": "urban",
    "current_speed": 29.5,
    "distance_to_vehicle_ahead": 11.0,
    "distance_to_goal": 6308.3
  },
  "attempted_command": null,
  "attempted_parameters": {},
  "rejection_reason": "Request could not be classified into any known vehicle
        behavior",
  "label": "negative_unknown_command"
}
```

Listing 4.3: Example JSON - Unknown-Command Sample - Step 1 – Structured Sampling

No valid command is proposed—this simulates a user request that doesn't map to any supported function.

```
{
  "id": "cb0cb691-94a8-4a93-bbf2-5b2af2801f74",
  "timestamp": "2025-06-01T23:18:55.337631+00:00",
  "sensor_inputs": {...},
  "attempted_command": null,
  "attempted_parameters": {},
  "rejection_reason": "Request could not be classified into any known vehicle
        behavior",
  "label": "negative_unknown_command",
  "generated_query": "Do you think this route is the most scenic?"
}
```

Listing 4.4: Example JSON - Unknown-Command Sample - Step 2 – LLM-Generated Query

Here, the user's question ("Do you think this route is the most scenic?") doesn't correspond to any driving command, yielding an "unknown" label.

Negative (Unsafe) Sample

```json
{
  "id": "ad498d51-697a-4da6-abc9-afd45b633620",
  "timestamp": "2025-06-01T23:18:55.154394+00:00",
  "sensor_inputs": {
    "scenario": "highway",
    "current_speed": 123.7,
    "distance_to_vehicle_ahead": 40.0,
    "distance_to_goal": 23889.8
  },
  "attempted_command": "drive_sportier",
  "attempted_parameters": {
    "acceleration_mode": "aggressive"
  },
  "rejection_reason": "Exceeded speed limit for command 'drive_sportier'",
  "label": "negative"
}
```

Listing 4.5: Example JSON - Negative (Unsafe) Sample - Step 1 – Structured Sampling

An "aggressive" sport-driving request on the highway exceeds the permitted speed envelope, so it's flagged as unsafe.

```json
{
  "id": "ad498d51-697a-4da6-abc9-afd45b633620",
  "timestamp": "2025-06-01T23:18:55.154394+00:00",
  "sensor_inputs": {...},
  "attempted_command": "drive_sportier",
  "attempted_parameters": {...},
  "rejection_reason": "Exceeded speed limit for command 'drive_sportier'",
  "label": "negative",
  "generated_query": "Seriously, can you please pick up the pace? We're barely
         moving faster than everyone else! Just a little more sporty, give it some
         pep, you know? Like, drive a little sportier? Come on, it's a straight
         shot, what's the holdup?"
}
```

Listing 4.6: Example JSON - Negative (Unsafe) Sample - Step 2 – LLM-Generated Query

This vividly colloquial request underscores an unsafe "sportier" command that the system must learn to reject.

The two-stage pipeline demonstrates its functionality by first applying rule-based sampling for safety and plausibility checks before using LLM prompting to generate natural and varied language.

The resulting training data becomes rich enough to support the development of safety-aware distilled language models. The synthetic dataset contains 30,000 positive samples and 10,000 negative samples together with 1,000 “unknown” samples. The dataset provided enough coverage to support both training and evaluation of the domain-adapted model.

Data Preprocessing and Splitting

To prepare the datasets for model consumption, a consistent preprocessing and splitting methodology was applied to both the xLAM benchmark and the synthetic DLR corpus. This ensures that the model receives input in a standardized format and that the evaluation is conducted rigorously and without data leakage.

Input Formatting The data structure for each sample included the natural language query along with relevant contextual information (e.g., sensor data for the DLR dataset or tool schemas for xLAM) and the target output, which was serialized into a unified text format. The transformation process converts the separate JSON fields into a unified prompt that the language model can interpret. A standard template was applied to all samples to maintain consistency, separating the different input sections (e.g., [CONTEXT], [QUERY], [COMMAND]). This structured text format serves as a critical element for the model to understand how context relates to user intent and the resulting executable commands.

Dataset Splitting Both the xLAM and DLR synthetic datasets were divided into three distinct, non-overlapping subsets. The training set (80%) is used to adjust the model weights by learning to map inputs to outputs through parameter optimization. The validation set (10%) serves to monitor performance during training, tune hyperparameters such as the learning rate, and help prevent overfitting; while the model’s performance on this set informs adjustments to the training process, it does not directly influence the learned weights. Finally, the test set (10%) remains strictly held out and is used solely for the final evaluation, providing an unbiased estimate of the model’s ability to generalize to unseen data. The standard 80/10/10 split provides enough data for strong training while keeping separate sets for validation and final testing. The test set must remain intact because it represents the model’s real-world performance when encountering new inputs during deployment.

4.4. Model Configuration

The effectiveness of knowledge distillation relies heavily on selecting a suitable teacher–student model pair. This section describes the configuration of the models employed in this study. The large-scale teacher model was chosen for its strong performance and capacity to produce informative training signals, while the compact student model was selected as a realistic candidate for deployment in resource-limited settings.

4.4.1. Teacher Model (LLaMA 3.1 8B)

The selection of an appropriate teacher model determines the effectiveness of knowledge distillation because it establishes the maximum amount of information the student can learn. The advanced open-weight language model LLaMA 3.1 8B developed by Meta AI [2] served as the chosen model for this purpose. The model contains 8 billion parameters which creates an optimal combination between advanced reasoning capabilities and research-friendly computational efficiency. The selection of LLaMA 3.1 8B as the model depends on two essential factors. The model generates precise supervisory signals because it understands language complexities and follows complex instructions to produce high-quality outputs that include complete probability distributions (soft labels) for vocabulary transfer during distillation. The LLaMA family has demonstrated successful performance across multiple NLP benchmarks which include tasks that need structured output and function calling thus ensuring the distilled knowledge remains accurate and relevant to the task. The expert knowledge source function of the teacher model provides target outputs which direct student model training.

4.4.2. Student Model (LLaMA 3.2 1B)

The student model was used because it addresses essential factors including memory usage and power consumption and inference speed. The main reasons for choosing LLaMA 3.2 1B include its ability to operate in resource-limited environments and its architectural similarity to the teacher model. The 1B model measures approximately 2 GB in size which allows it to operate on edge devices and embedded systems including in-vehicle computers used for autonomous driving. The 16 GB teacher model exceeds practical deployment limits for such scenarios. Selecting a student from the same LLaMA family provides both architectural consistency between tokenizer and transformer components. The consistent architecture between teacher and student minimizes their representational difference which enables better knowledge transfer during distillation. The main objective involves transforming the extensive capabilities of the 8B teacher into a compact 1B student model which will maintain sufficient capability for its tasks while adhering to real-world deployment constraints.

4.4.3. Architectural Alignment

A successful knowledge distillation process requires both models to have high architectural compatibility especially when working between models of varying sizes. The training process benefits from this alignment because it enables the focus on semantic and reasoning capability transfer instead of resolving technical differences. Alignment was implemented across three essential dimensions in this work. The LLaMA 3.1 8B teacher and LLaMA 3.2 1B student use the identical tokenizer in their systems. The tokenizer alignment represents an essential foundational step because it ensures identical tokenization of any given text input for both models. A direct comparison of output distributions becomes impossible between models that do not share the same tokenizer which creates substantial noise during the distillation process. Both models share the same vocabulary among their operational framework. The student model receives all token sets from the teacher which means both models generate output probability distributions across identical next-token possibilities. Logit-based distillation requires an identical output space between models because it enables direct token-by-token prediction comparison through the Kullback-Leibler divergence loss. The embedding layers of these models follow basic architectural principles despite their differing depth and width dimensions. The process of distillation becomes simpler when embedding dimensionalities match between different models. The alignment between models becomes crucial for using feature-based distillation because it enables a direct regression from student hidden states to teacher outputs especially when advanced techniques like matching intermediate layer outputs are employed. The established alignment strategies form a stable system which enables effective knowledge transfer. The student model learns to duplicate teacher decisions precisely because of these strategies which ensure high-fidelity compression.

4.5. Knowledge Distillation Implementation

The following section explains the technical aspects of knowledge distillation after describing the experimental setup design at a high level. The section explains the fundamental approach for knowledge transfer between teacher and student models and presents the mathematical framework of the loss function and describes the training procedures. The section addresses the main implementation obstacles that emerged during experiments including catastrophic forgetting and explains the implemented solutions to achieve stable training results.

4.5.1. Core Distillation Strategy

The core methodology employs a pragmatic, hybrid distillation strategy. The two-stage method begins by using black-box data generation to handle domain-specific data scarcity before applying white-box training for efficient signal-rich knowledge transfer. The approach combines the best features of both paradigms to fulfill the DLR's operational requirements.

Response-Based Distillation for Data Generation

The hybrid strategy's first phase tackles the essential problem of limited data availability for specialized DLR applications. The lack of a substantial human-annotated corpus for autonomous vehicle commands leads to the use of the teacher model (LLaMA 3.1 8B) as a black-box generator of a high-quality training dataset. This process is a practical application of response-based distillation (Section 2.3.5) by extracting expert knowledge through the teacher's responses.

At this stage, the teacher model is treated as an opaque oracle, with exclusive focus on its input–output behavior. The synthetic data pipeline, detailed previously in Section 4.3.2, systematically feeds the teacher model structured scenarios, each containing contextual sensor data and a proposed action. In response, the teacher is prompted to generate two key outputs:

1. A varied, natural-language user query that a passenger might realistically say.

2. The corresponding, correctly formatted JSON command that represents the ground-truth action.

The *offline* distillation process converts the teacher's complex language and task structure knowledge into a big machine-readable dataset. The process allows the creation of a large domain-specific corpus which contains both valid commands and essential safety-aware rejection cases. The generated dataset functions as the main training data for white-box distillation which enables the student model to learn from high-quality targeted examples.

Logit-Based Distillation for Student Training

With a high-quality, domain-specific dataset generated, the second stage of the strategy employs a white-box training methodology known as *logit-based distillation*. This phase forms the core of the student's learning process, leveraging the teacher model's full internal "thought process" at each step of generation. Because the task is autoregressive, the student is not trained to mimic the final output in a single step. Instead, at every stage of generating the output sequence (e.g., the JSON command), the student is trained to replicate the teacher's complete probability distribution over the entire vocabulary for the *next token*.

This approach provides a significantly richer supervisory signal than using hard labels (which would only specify the single correct next token). The teacher's full distribution contains what is often called "dark knowledge". That is valuable information about the relative likelihood of all possible next tokens. For instance, after generating the prefix `{"command": "change_lane_` the teacher's logits provide a probability distribution for the next token. While the highest probability might be for the token *left*, it might also assign non-trivial probabilities to semantically similar tokens like *right*. This nuanced information teaches the student not just the single correct token, but also about the semantic space of plausible continuations. The student learns that both *left* and *right* are valid direction specifiers in this context, a much more powerful lesson than what is provided by a hard label. By minimizing the Kullback–Leibler (KL) divergence between the student's and the teacher's softened output distributions at each step in the sequence, the student is encouraged to learn a more robust and better-generalized internal representation of both the language and the task. This white-box approach is highly effective for transferring the fine-grained, token-level reasoning capabilities of the large teacher model into the compact student architecture.

4.5.2. Loss Function Design

The transfer of knowledge from the teacher to the student model is governed by a carefully designed composite loss function. This function is engineered to balance two distinct objectives: ensuring that the student model learns the correct, ground-truth answers (task-specific performance) while also absorbing the more nuanced, generalized knowledge from the teacher (distillation). As implemented in the custom `KDRecipeDistributed` within the `torchtune` framework, the total loss, $\mathcal{L}_{\text{total}}$, is defined as a weighted sum of two components: a standard cross-entropy loss and a distillation loss.

Cross-Entropy Loss ($\mathcal{L}_{\text{CE}}$) The first component is the conventional cross-entropy loss, which is standard for training autoregressive language models. This loss measures the dissimilarity between the student model's predicted next-token probabilities and the ground-truth "hard" labels from the training data. Its purpose is to anchor the student's learning to the correct answers, ensuring it maintains high task-specific accuracy. It is defined as:

$$\mathcal{L}_{\text{CE}} = -\sum_i y_i \log(\hat{y}_i) \tag{4.1}$$

where y_i is the one-hot encoded ground-truth token and $\hat{y}_i$ is the student's predicted probability for that token.

Knowledge Distillation Loss ($\mathcal{L}_{\text{KD}}$) The second, and more critical, component is the distillation loss, which encourages the student model to emulate the teacher's full output distribution. This is achieved using the Kullback-Leibler (KL) divergence, which quantifies how one probability distribution diverges from a second, reference distribution. To expose the "dark knowledge" within the teacher's predictions, the output logits of both models (z_t for the teacher, z_s for the student) are first softened using a **temperature** parameter, $T > 1$. A higher temperature smooths the probability distribution, placing more emphasis on the relative probabilities of less likely tokens.

Following the formulation proposed by Hinton et al., the distillation loss is defined as:

$$\mathcal{L}_{\text{KD}} = T^2 \cdot \text{KL}\left(\sigma\left(\frac{z_t}{T}\right) \parallel \sigma\left(\frac{z_s}{T}\right)\right) \tag{4.2}$$

where $\sigma(\cdot)$ is the softmax function. The scaling factor T^2 is applied to ensure that the magnitude of the gradients produced by the soft targets remains roughly on the same scale as those produced by the hard targets.

Composite Loss Function The final loss function combines these two objectives using a weighting factor, $\alpha \in [0, 1]$, to control the balance between them:

$$\mathcal{L}_{\text{total}} = (1 - \alpha) \cdot \mathcal{L}_{\text{CE}} + \alpha \cdot \mathcal{L}_{\text{KD}} \tag{4.3}$$

The experiments showed that $\alpha = 0.5$ produced a stable equilibrium which equally emphasized learning the ground truth and mimicking the teacher model. The temperature parameter was set to $T = 2$ to achieve sufficient softening of output distributions without causing instability. The composite loss function directs the student model to generate accurate predictions while also learning the teacher's reasoning process effectively.

4.5.3. Training Regime

The training process was carefully structured to ensure stable convergence and computational efficiency. All experiments were conducted using the Hugging Face `accelerate` framework, which provided a robust and reproducible environment for managing the distillation workflow on a distributed multi-GPU infrastructure. Training was orchestrated via SLURM scripts on the high-performance computing cluster (kratos). The following regime was applied when training the student model.

Optimization Configuration For optimizing the student model (LLaMA 3.2 1B), the *AdamW optimizer* was employed, as it is well-suited for training transformer-based architectures due to its effectiveness and stability. The learning rate was governed by a scheduler combining a linear warm-up phase over the first 500 steps with a cosine decay for the remaining training duration. This scheduling strategy stabilizes the learning dynamics in the early stages and gradually reduces the learning rate as convergence progresses. The principal hyperparameters were set as follows: a learning rate of 2.0×10^{-5}, an effective batch size of 128 (derived from a per-device batch size of 8 with gradient accumulation over 4 steps on 4 NVIDIA A100 GPUs), a total of 10,000 training steps, and a weight decay of 0.01. These hyperparameter choices were guided by values reported in recent published studies on transformer distillation [54], [78], [111], ensuring alignment with established best practices while accommodating the specific computational constraints of this work.

Training Execution To maximize computational efficiency and reduce memory consumption, training was performed using `bfloat16` (bf16) mixed precision. This approach leverages the specialized hardware capabilities of recent GPUs to accelerate computation without a significant loss of numerical precision. To mitigate the risk of exploding gradients, which can occur during the training of deep neural networks, a gradient clipping threshold of 1.0 was applied.

Checkpointing and Logging Model checkpoints were saved at regular intervals (every 1,000 steps) to enable training to resume from the latest state in case of interruptions and to facilitate evaluation at various stages of model development. All training metrics, including the total loss, its cross-entropy and KL divergence components, and validation scores, were logged in real time using the *Weights & Biases (`wandb`)* platform. This continuous logging provided a clear visual record of the training dynamics and enabled systematic comparisons across different experimental runs.

4.5.4. Implementation Challenges and Mitigations

While the overall training process proved robust, the fine-tuning of large language models is not without its challenges. The most significant issue encountered during the initial experimental phase of this work was *catastrophic forgetting*, a well-documented phenomenon where a model, when adapted to a new, narrow task, can rapidly lose the broad, generalized knowledge acquired during its pre-training [112]–[114]. Understanding and addressing this challenge was important to produce a final model that was both task-proficient and linguistically coherent.

The Phenomenon of Catastrophic Forgetting Catastrophic forgetting refers to the tendency of a neural network to abruptly lose previously acquired knowledge when fine-tuned on new data [112]–[114]. In the context of LLMs, this manifests as a sharp drop in performance on tasks or domains the model had mastered during pre-training [115]. Because distillation and deployment scenarios often involve such sequential adaptations, mitigating this effect is critical to preserve both broad linguistic competence and domain-specific capabilities.

Empirical studies have identified two primary mechanisms behind catastrophic forgetting in LLMs. First, fine-tuning often pushes the model into "sharp" minima of the loss landscape; such regions amplify sensitivity to subsequent updates, thereby rapidly erasing earlier knowledge. Sharpness-aware minimization (SAM) has been shown to flatten these minima and reduce forgetting during fine-tuning [113]. Second, when the learning rate is too high relative to batch size and dataset scale, parameter updates can overwrite large portions of the pre-trained weights.

Observed Impact and Mitigation Strategy During the initial experiments conducted on the tool-calling task, the detrimental impact of unsuitable training configurations became readily apparent. An early training run, which employed a high learning rate in combination with a small batch size, yielded a fine-tuned model that underperformed relative to the original pre-trained base model in terms of validation metrics. Its capacity to generate coherent and well-structured JSON outputs declined noticeably, underscoring how overly aggressive updates can cause catastrophic forgetting, even when fine-tuning is restricted to a single downstream task.

To address this issue, various strategies have demonstrated their effectiveness in large language model workflows, such as self-synthesized rehearsal, *Parameter-Efficient Fine-Tuning (PEFT)*, and model merging [112], [114], [116]. However, emphasis is placed on the most straightforward and fundamental measure: the implementation of an *optimized learning schedule*. The training regime, detailed in Section 4.5.3, was specifically devised to alleviate catastrophic forgetting through the careful adjustment of critical hyperparameters. In particular, a low learning rate was employed to constrain the magnitude of parameter updates, while a warm-up schedule facilitated a gradual adaptation to the new data. Additionally, a cosine decay schedule ensured that updates diminished progressively in size, thereby promoting stable convergence. By configuring the optimization process in this deliberate manner, the model's essential pre-trained capabilities were retained while it was successfully adapted to the target domain, ultimately resulting in robust performance in the final evaluation.

4.6. Evaluation Setup

To conduct a rigorous and multi-faceted assessment of the knowledge distillation framework, a comprehensive evaluation protocol was established. This protocol is designed to empirically test the research hypotheses by quantifying the trade-offs between task-specific performance and computational efficiency. The performance of the primary experimental model—the distilled student—is systematically measured against well-defined baselines using a suite of specific metrics. This ensures that the findings presented in the subsequent chapter are robust, reproducible, and directly comparable.

4.6.1. Task Performance Metrics

The best measure of success is the model's ability to accurately perform its designated task: translating natural language queries into structured, executable commands (or functions). To capture different facets of this capability, a combination of metrics will be employed for the function-calling task on both the xLAM and the DLR-specific datasets.

- **Function Accuracy:** This metric measures the percentage of predictions where the model correctly identifies the name of the function to be invoked (e.g., `change_lane_left`). It evaluates the model's high-level intent recognition (choosing the right function).
- **Exact Match (EM) Accuracy:** This is the strictest metric, calculating the percentage of predictions where the entire generated JSON output is a perfect string match to the ground-truth command. It penalizes any error in the function name, arguments, or formatting, providing a measure of overall generative fidelity.
- **Argument F1-Score:** This metric provides a more granular assessment of the model's ability to extract and structure the correct parameters for a given function. It is calculated by treating the key-value pairs within the `parameters` object of the JSON output as a set. The F1-score, which is the harmonic mean of precision and recall, is then computed over these sets. This allows for the evaluation of parameter extraction accuracy independently of minor formatting differences or the correctness of the function name itself, offering a more nuanced view of the model's information extraction capabilities.
- **Rejection Accuracy:** Specific to the DLR synthetic dataset, this crucial safety-oriented metric measures the model's ability to correctly handle invalid inputs. It is the percentage of `negative` (unsafe) and `negative_unknown_command` samples for which the model correctly produces a rejection response rather than attempting to generate a command.

4.6.2. Computational Efficiency Metrics

To quantitatively measure the efficiency gains achieved through knowledge distillation, the following metrics will be recorded under a standardized hardware environment (a single NVIDIA A100 GPU).

- **Model Size:** The on-disk size of the model's weights, measured in gigabytes (GB). This is a direct indicator of storage requirements on an edge device.
- **Inference Latency:** The time required for the model to generate a response, which is critical for real-time applications. This will be reported in two ways:
  - *Time per Output Token:* The average time in milliseconds (ms) to generate a single token, measuring the fundamental generation speed.
  - *End-to-End Request Time:* The average total time in seconds (s) to process a standard-length prompt and generate a complete command, reflecting the user-perceived latency.

- **Memory Footprint (VRAM):** The peak GPU memory consumed during an inference pass for a representative batch of inputs, measured in gigabytes (GB). This metric is critical for determining the minimum hardware requirements for deployment.

4.6.3. Baselines for Comparison

The first baseline is the pre-trained teacher model (LLaMA 3.1 8B). This model is evaluated on the test set in a zero-shot, identical to how it was used to generate the training data and soft labels. Its performance represents the benchmark for generalized, in-context reasoning on the task, without the benefit of task-specific fine-tuning. This baseline establishes the level of performance that can be achieved by the large-scale base model, serving as a target for the distillation process.

The second and most critical baseline is the fine-tuned baseline student model (LLaMA 3.2 1B). This model shares the same compact architecture as the KD-Student but is trained conventionally using only the ground-truth labels from the dataset (i.e., trained with the standard cross-entropy loss, where $\alpha = 0$). This baseline demonstrates the performance achievable by simply fine-tuning the smaller model on the task. By comparing the KD-Student to this baseline, we can precisely quantify the performance gains attributable specifically to the generalized knowledge transferred from the pre-trained teacher, highlighting the value of distillation over standard fine-tuning.

4.7. Threats to Validity

A thorough evaluation of this study's experimental design becomes necessary to achieve a fair interpretation of the results. This section examines the possible threats to the internal validity and external validity and construct validity of the findings.

4.7.1. Internal Validity

The external validity of these findings depends on two main factors. First, the evaluation of the domain-specific DLR use case relies on synthetic data generation. Although the model's strong performance on synthetic data is promising, it must still be tested with real-world field data before it can be applied in practice. Second, the experiments were conducted only with models from the LLaMA architectural family. Therefore, the rates of performance recovery and efficiency gains observed here may not fully extend to distillation between other major model families, which often have different scaling behaviours and internal structures.

4.7.2. External Validity

The external validity, or generalisability, of these findings depends on two main aspects. Firstly, the evaluation of the domain-specific DLR use case is based on a synthetically generated dataset. Even though the generation pipeline was designed with care to mimic reality, it might not fully reflect the variety of noise, ambiguity, and idiomatic expressions that real-world passenger commands can contain. As a result, the model's strong performance on synthetic data is encouraging but still needs to be verified with genuine field data before being used in real operations. Secondly, the experiments were carried out only with models from the LLaMA architectural family. Therefore, the observed levels of performance recovery and efficiency improvements might not transfer directly to distillation between other well-known model families, which could have different scaling behaviours and internal representational structures.

4.7.3. Construct Validity

Finally, construct validity concerns whether the chosen evaluation metrics truly reflect the theoretical ideas of model capability and safety. Metrics like *Exact Match Accuracy*, although objective and reproducible, tend to be rather strict and might penalise semantically correct outputs that only differ in minor syntactic aspects, such as a different order of keys in a JSON object. This drawback is partly addressed by adding the Argument F1-Score, which offers a more semantically oriented measure of parameter extraction. Likewise, the rule-based safety logic used in the synthetic data generator remains a necessary simplification of the complex and dynamic reasoning that a certified autonomous system would require. The model's ability to learn these simplified rules can be seen as a solid sign of its contextual reasoning skills, but it should not be taken as proof of full safety in every real-world situation.

4.8. Hardware and Software Resources

This section outlines the computational infrastructure, data resources, and software frameworks required to implement and assess knowledge distillation. The use of high-performance computing resources, diverse datasets, and specialised deep learning toolkits provides a solid foundation for carrying out scalable and efficient distillation experiments.

4.8.1. Hardware Specifications and Computational Infrastructure

All experiments discussed in this thesis were performed using the high-performance computing facilities of the DLR. The available GPU resources include NVIDIA Tesla V100 and A100 units spread across multiple systems, adding up to a total of around 80 GPUs. These accelerators deliver the computational power necessary for training large-scale teacher models. The CPU infrastructure comprises over 6000 cores deployed across standard and high-memory nodes, supporting data preprocessing, evaluation, and other auxiliary tasks. High-speed interconnects, such as 400 Gbit/s InfiniBand connections for GPU nodes, allow for efficient distributed training. Storage is handled by a parallel BeeGFS file system with an overall capacity of nearly one petabyte, complemented by fast local NVMe scratch storage on each node. Only parts of this overall compute infrastructure were actually used for model training, synthetic data generation, and evaluation. All compute jobs were managed through the Slurm workload manager and submitted using `sbatch` scripts.

4.8.2. Software Resources

Deep Learning Frameworks and Libraries The core development work was carried out using PyTorch, which is a dynamic and flexible deep learning framework. The model architectures, training loops and custom loss functions (including the knowledge distillation loss) were all implemented with native PyTorch APIs. The Hugging Face Transformers library was used to load and manage pretrained models, especially for initializing both teacher and student models from the LLaMA family. Additionally, standard scientific computing libraries like NumPy were employed for various numerical operations and data processing tasks.

Experiment Management and Execution All experiments were run on a high-performance computing (HPC) cluster, which was managed using the SLURM Workload Manager. SLURM scripts (for example, `generate_synthetic_data.sh` and `llm_labeling.sh`) automated the environment setup, job scheduling, GPU allocation and log handling. The software environments were contained within a Python virtual environment and, when needed, further containerized with Docker. This helped ensure reproducibility and system consistency across different machines and among various users.

LLM Query Generation Infrastructure For generating synthetic user queries, the pipeline supported both the OpenAI API (for example, GPT-4) and local inference through Ollama. These interfaces were implemented in `llm-gen-v5.py`, using the `openai`, `requests` and `python-dotenv` libraries to handle API communication and secret loading. Overall, this software stack offered a reliable, modular and flexible foundation for all stages of the knowledge distillation framework, from data generation to the final evaluation.

4.8.3. Resource Profile of Large Language Models

Deploying and training LLMs are resource-intensive endeavors, demanding considerable computational and memory resources. A clear understanding of these requirements is fundamental not only for performance benchmarking but also for appreciating the practical benefits of model compression techniques like Knowledge Distillation. This section provides an analysis of the memory and computational demands of LLMs, drawing upon established formulas and recent industry benchmarks to frame the discussion [48], [117], [118].

Inference Memory Footprint The memory required to run an LLM for inference is determined by two primary factors: the static size of the model's weights and the dynamic memory needed for the KV cache during processing.

- **Model Weights:** This is the memory needed to load the model's parameters onto the GPU. It is a fixed cost per model, typically stored in 16-bit floating-point format (FP16), which consumes 2 bytes per parameter. For example, a 3-billion-parameter model like LLaMA 3.2 3B requires approximately $3 \times 2 = 6$ GB of VRAM.

- **KV Cache:** Transformer architectures use a key-value (KV) cache to store attention states for each token in the input sequence. This avoids recomputing states for previous tokens when generating new ones. The size of this cache grows linearly with the sequence length and the number of concurrent users. Its size per token is a function of the model's architecture, calculated as [48]:

  $$\mathrm{KV}_{\mathrm{token}} = 2 \times 2 \times n_{\mathrm{layers}} \times d_{\mathrm{model}} \text{ bytes}$$

  For the LLaMA 3.2 3B model, which has 32 layers and a hidden dimension of 2560, the KV cache per token is approximately 0.00031 GiB. For a single user with a 4096-token prompt, this adds an extra 1.25 GB to the memory footprint.

Training Resource Demands Training an LLM is generally much more demanding than performing inference. The total VRAM needed must hold not only the model weights but also the gradients produced during backpropagation and the states that the optimizer keeps. For example, the widely used Adam optimizer may consume up to 12 bytes per parameter to ensure stable mixed-precision training [118]. Moreover, intermediate activations from the forward pass must be stored for computing gradients later, and their memory footprint scales with batch size, sequence length, and model dimensions. When this is combined with general overhead from memory fragmentation, which might add another 10–20%[117], training a model like LLaMA3.2 13B could easily require more than 300GB of VRAM. This situation often makes distributed training strategies such as ZeRO[119] or FSDP [120] necessary to make training feasible.

Performance: Throughput and Latency A model's performance in a real-world deployment is often measured by its throughput (concurrent users) and latency (response time). The VRAM of the GPU is the primary constraint on throughput. The maximum number of concurrent users can be estimated by allocating the VRAM remaining after loading the model weights to the KV caches of all users. Based on this logic, a benchmark from Fu (2024) demonstrates that a LLaMA 3.2 8B model on a 48 GB GPU can support approximately 8–10 simultaneous users with 4096-token prompts [48]. Latency, or the time to get a response, is composed of two phases. The initial processing of the prompt (prefill) is typically compute-bound, limited by the GPU's TFLOP/s. The subsequent generation of each token (decoding) is memory-bound, limited by the GPU's memory bandwidth. Using the LLaMA 3.2 8B model on an NVIDIA L40 GPU as an example, Fu [48] calculates a prefill time of approximately 0.088 ms per token and a generation time of 18.5 ms per token. For a task involving a 4096-token prompt and a 256-token response, this yields a total latency of about 5.1 seconds under ideal conditions [48].

Table 4.2.: Comparison of LLaMA 3.2 Model Variants: Inference and Training Resource Requirements

Model	Params	Config (L/d)	VRAM (Inference)	VRAM (Training)	Recommended GPUs	Use Case
LLaMA 3.2 1B	1B	18 / 2048	3–4 GB	20–30 GB	RTX 3090, A10	KD Student, Edge Devices
LLaMA 3.2 3B	3B	32 / 2560	6–8 GB	45–70 GB	A10, A100 40GB	Baseline Student Model
LLaMA 3.2 8B	8B	32 / 4096	16–21 GB	160–200 GB	A100, H100, L40	KD Teacher, General LLM
LLaMA 3.2 13B	13B	40 / 5120	26–30 GB	280–320 GB	H100 SXM, A100 80GB	KD Teacher, Dialogue
LLaMA 3.2 70B	70B	80 / 8192	100–140 GB	600–900 GB	4–8× H100	Multi-turn Chat, RAG
LLaMA 3.2 405B	405B	96 / ∼16000	600–800 GB	2–4 TB	TPUv5, DGX GH200	Research-Scale LLM

5. Evaluation

This chapter presents the empirical findings of the study and directly addresses the research hypotheses formulated in Section 4.1. The following sections outline the outcomes of the experiments described in the methodology, providing both quantitative and qualitative insights into the proposed knowledge distillation framework. The evaluation focuses on a comparative analysis, systematically measuring the performance of the main experimental model — the 1-billion parameter Knowledge Distilled Student (KD-Student) — against two key reference points: the large 8-billion parameter teacher model, which represents the performance upper bound, and a 1-billion parameter baseline student model trained conventionally without distillation, which acts as the performance lower bound.

The presentation of results is structured to build a clear and logical argument. It starts with a validation of the distillation approach on the public xLAM function-calling benchmark, establishing a general performance baseline. This is followed by an evaluation of the model's effectiveness on the domain-specific DLR dataset, with particular focus on its capacity for safety-aware command rejection. The analysis then shifts to the key improvements in computational efficiency, considering reductions in model size, memory footprint, and inference latency. The chapter ends with a qualitative assessment of representative model outputs, providing concrete examples that demonstrate the practical implications of the knowledge distillation process.

5.1. Performance on Public Benchmark: xLAM Function Calling

To quantitatively assess the core effectiveness of the knowledge distillation framework, its performance was first evaluated on the xLAM Function Calling dataset. This standardized, open-domain benchmark helps to isolate and measure the fidelity of the knowledge transfer process in a general context, separate from the specific demands of the DLR use case. The following sections present the comparative results for the teacher, baseline, and KD-Student models on this task, providing a foundational indication of performance recovery.

5.1.1. Comparative Performance of Teacher, Baseline, and KD-Student

The initial evaluation on the xLAM benchmark offers a clear quantitative comparison of the three main model configurations. The performance of each model was measured using the core metrics of Exact Match Accuracy, Function Accuracy, and Argument F1-Score to assess their capability to translate natural language queries into structured, executable commands.

The results, summarized in Table 5.1, establish the 8B teacher model as a capable benchmark, achieving an exact match accuracy of 72.5%. In contrast, the baseline 1B student, trained conventionally on ground-truth labels, performed poorly on this difficult task. This baseline model secured an exact match accuracy of only 24.8%, representing a 47.7 percentage point drop in performance compared to the teacher. This gap illustrates that the baseline student struggled to generalize on this complex benchmark.

The KD-Student, however, demonstrates a significant relative improvement. Achieving an exact match accuracy of 51.3%, the distilled model more than doubles the performance of its non-distilled counterpart. Specifically, the KD-Student recovered 26.5 percentage points of the 47.7-point performance deficit, which corresponds to reclaiming over 55% of the capability lost by the baseline student. This strong trend is consistent across all evaluated metrics. The KD-Student's improved Function Accuracy (60.1%) and Argument F1-Score (0.54) further indicate that the distillation process effectively transferred not only high-level intent recognition but also the more fine-grained ability to correctly extract and structure command parameters. These findings provide strong initial evidence that knowledge distillation is an effective technique for elevating a small model to a level of basic competency on complex, structured-output tasks.

Table 5.1.: Comparative performance on the xLAM Function Calling test set. The results reflect the difficulty of the benchmark, with the KD-Student showing a marked improvement over the baseline.

Model	**Exact Match (%)**	**Function Accuracy (%)**	**Argument F1-Score**
Teacher (LLaMA 3.1 8B)	72.5	78.3	0.71
Baseline Student (1B)	24.8	35.2	0.29
KD-Student (1B)	**51.3**	**60.1**	**0.54**

5.1.2. Analysis of Performance Recovery via Distillation

Building on the quantitative results from the previous section, this analysis explores the underlying mechanism of knowledge distillation that allows such a notable recovery of performance. The success of the KD-Student lies in the qualitative difference between the supervisory signals used in conventional fine-tuning and those in distillation. Conventional fine-tuning, as applied to the baseline model, depends on "hard" targets from ground-truth labels. This method provides a rather sparse learning signal, reinforcing only the single correct token at each generative step. While this approach can train a model to reproduce specific input-output patterns, it offers little information about the relationships among different possible outputs. This can lead to a more brittle model that struggles to generalize to small variations in user queries.

5.2. Performance on Domain-Specific DLR Use Case

Having shown the effectiveness of knowledge distillation on a standard benchmark, the evaluation now moves to the main domain of this thesis, which is the DLR autonomous vehicle use case. This part of the analysis looks at how well the models can handle a more complex and safety-critical task. Success here requires not just accurate translation of natural language into structured commands but also the ability to integrate contextual sensor data and, just as important, to reject unsafe or unclassifiable instructions. For this, the evaluation uses the domain-specific synthetic dataset created for this purpose (section B), which includes these particular challenges. The following subsections show the results, starting with task-specific accuracy and then giving a focused look at the models' safety-aware rejection performance.

Upon transitioning the evaluation to the DLR-specific synthetic dataset, the models were tested on their ability to process domain-specific commands that are conditional on contextual sensor inputs. This task provides a more challenging and realistic assessment of the models' capabilities for the target application, as it requires them to ground their interpretation of a user's query within a given environmental and operational state.

The main performance metrics for valid, executable commands from this dataset are shown in Table 5.2. As expected, the 8B teacher model shows strong performance, correctly interpreting the commands with an exact match accuracy of 88.7%. The baseline 1B student, however, struggled with the added complexity of this scenario. Its exact match accuracy dropped to 62.4%, a substantial decline that highlights the difficulty of processing contextual sensor information without advanced guidance. This suggests that conventional fine-tuning alone was not enough to teach the smaller model how to reliably integrate this information into its decisions.

In contrast, the KD-Student showed a much more pronounced improvement in this targeted context than on the general benchmark. With an exact match accuracy of 76.1%, it demonstrates a clear and significant improvement of over 13 percentage points compared to the baseline. The improved Argument F1-Score of 0.75 is also noteworthy, as it indicates that the distilled model more effectively learned to extract and format parameters that depend directly on the vehicle's contextual state, such as reacting to a certain speed or proximity reading.

These results indicate that knowledge distillation is a particularly viable method for transferring context-aware reasoning. While the baseline model had difficulty linking user requests to sensor data, the KD-Student successfully learned this critical skill from the teacher's soft labels, underscoring its potential for real-world applications where contextual understanding is paramount.

Table 5.2.: Performance on the DLR domain-specific test set (valid commands only). The targeted KD approach yields substantial gains over the baseline.

Model	**Exact Match (%)**	**Function Accuracy (%)**	**Argument F1-Score**
Teacher (LLaMA 3.1 8B)	88.7	90.5	0.87
Baseline Student (1B)	62.4	71.8	0.63
KD-Student (1B)	**76.1**	**83.4**	**0.75**

5.2.1. Evaluation of Safety-Aware Rejection Behavior

A critical ability for any model meant to be used in an autonomous system is knowing when to recognize and refuse commands that are unsafe, unclear, or just outside what it should do. This section looks at this safety-aware behavior by using the "Rejection Accuracy" metric, which checks how well the model can tell if an input should be accepted or rejected. The synthetic DLR dataset, which includes unsafe and unknown command situations on purpose, forms the basis for this part of the analysis.

How each model performs on this simple binary classification task is shown in the confusion matrices in Table 5.3. These matrices give a detailed look at how each model deals with both valid and invalid inputs. The 8B teacher model once again acts as the performance benchmark, setting the standard for safe behavior. It correctly accepts 97.0% of all valid commands and, even more importantly, correctly rejects 97.2% of all invalid inputs. This shows a highly reliable understanding of the system's operational and safety limits. The baseline 1B student, on the other hand, shows a serious safety weakness. As seen in its confusion matrix, it only rejects 60.4% of invalid commands. This means it would wrongly go ahead and execute nearly 40% of all unsafe or unknown user requests. This high rate of false negatives (commands that should be rejected but aren't) is simply too risky for any real-world, safety-critical situation. The KD-Student, however, shows a substantial improvement in this crucial area. By correctly rejecting 84.9% of invalid inputs, it manages to inherit the teacher model's cautious and context-aware reasoning. This cuts down the rate of critical failures—accepting invalid commands—from 39.6% to 15.1%, a reduction of approximately 62% compared to the baseline student. This clearly suggests that the distillation process passed on not just task-specific skills but also the more subtle, safety-critical decision boundaries the teacher learned.

These findings demonstrate that the value of knowledge distillation extends beyond task accuracy. The ability to impart a larger model's safety-awareness onto a smaller, more efficient one is a key result, making distillation an important technique for building models that are robust and reliable enough to be used in high-stakes situations.

Table 5.3.: Confusion matrices for safety-aware rejection behavior on the DLR test set. Values are shown as percentages of the total samples in each true class.

	Teacher (8B)		**Baseline Student (1B)**		**KD-Student (1B)**	
True Label	**Pred: Accept**	**Pred: Reject**	**Pred: Accept**	**Pred: Reject**	**Pred: Accept**	**Pred: Reject**
Valid Cmd	97.0%	3.0%	91.1%	8.9%	92.5%	7.5%
Invalid Cmd	2.8%	**97.2%**	39.6%	**60.4%**	15.1%	**84.9%**

5.3. Analysis of Computational Efficiency Gains

Beyond just task performance and safety, another big goal of this thesis is to get a model that can actually run in the resource-limited environments at DLR. So this section now takes a closer look at how much more efficient the model gets through knowledge distillation. For many real-world cases, especially in embedded systems like the ones in autonomous vehicles, things like memory use and how fast the model runs aren't just nice to have — they're must-haves. This part checks how well the distilled model holds up to these real limits by comparing it with the bigger teacher model. The comparison looks at three important parts of efficiency: how much storage space the model needs on disk, the peak GPU memory (VRAM) it uses during inference, and the total time it takes to get a response (inference latency).

5.3.1. Comparison of Model Size and Memory Footprint

The first look at computational efficiency focuses on the static resource needs of the models — mainly their on-disk storage size and the memory they use during inference. These things really matter because they directly affect whether a model can even run on hardware with limited capacity, like the processing units in autonomous vehicles. The results of this comparison are shown in Table 5.4. As you can see, moving from the big 8B teacher model to the smaller 1B student brings a big drop in resource use. The model size on disk goes down from 16.0 GB to 2.0 GB, which is eight times smaller. At the same time, the peak VRAM needed for inference drops from 18.2 GB to just 2.8 GB, which is a reduction of almost 85%.

This big drop in resource needs is really what makes it possible to use the model in a real DLR setting. A memory footprint of over 18 GB makes the teacher model basically impossible to run directly on-device, so it would be limited to strong server-grade GPUs. The 2.8 GB footprint of the student model, though, fits well within what many modern embedded systems can handle. It's worth pointing out that the baseline and KD-Student models have the same architecture, so they naturally have the same size and memory use. This means the efficiency gain comes straight from switching to the smaller model design. The key thing with knowledge distillation, as shown earlier, is that it makes this move to a smaller model possible while still keeping good task performance and safety-awareness.

Table 5.4.: Comparison of model resource requirements. The student architecture offers a significant reduction in size and memory usage, making it suitable for embedded deployment.

Model	Parameters	Model Size (GB)	Peak VRAM Usage (GB)
Teacher (LLaMA 3.1 8B)	8B	16.0	18.2
Baseline Student (1B)	1B	2.0	2.8
KD-Student (1B)	**1B**	**2.0**	**2.8**

5.3.2. Inference Latency and Throughput

Apart from the static resource needs, how the model performs in action, especially its inference latency, is also really important for real-time use. For the DLR voice-command system, having low latency matters a lot to keep the interaction between the passenger and the vehicle feeling smooth and natural. This part takes a closer look at how fast the models actually run by checking two main things: how long it takes on average to make one output token, and the total end-to-end time it needs to handle a typical request.

The latency numbers, measured under a standard load, are shown in Table 5.5. The results clearly show a big improvement in inference speed when using the 1B student model. The average time it takes to generate a single output token drops from 19.2 ms for the big 8B teacher to just 3.8 ms for the student. This big boost in token speed also means a much shorter total request time. The student model can handle a complete, typical user query in about 0.31 seconds. That's a solid 4.7 times faster than the 1.45 seconds needed by the much larger teacher model. For a human-machine interface, this speed-up really matters. A response time well under half a second feels almost instant to users and helps keep the interaction smooth and natural. On the other hand, a delay close to 1.5 seconds can feel slow and break the flow of a conversation.

Together with the low memory footprint described earlier, this fast performance shows that the student model design meets both the static hardware limits and the dynamic speed needs for running on-board in a real-time system. Knowledge distillation makes it possible to build a model that's not only smaller and accurate, but also really responsive when it needs to be.

Table 5.5.: Inference latency comparison. All measurements were conducted on a single NVIDIA A100 GPU.

Model	**Time per Output Token (ms/token)**	**End-to-End Request Time (s)**
Teacher (LLaMA 3.1 8B)	19.2	1.45
Baseline Student (1B)	3.8	0.31
KD-Student (1B)	**3.8**	**0.31**

5.4. Qualitative Analysis of Model Outputs

While the quantitative results in the sections before give a good overall picture of how the models perform, looking at specific outputs helps build a more intuitive feel for how they actually behave. By seeing how the different models respond to the same tricky inputs, it's possible to get a better idea of the practical differences in their reasoning and reliability. This part talks about some representative examples of model outputs, which can be found in full in Appendix D.

One area where the KD-Student really stands out is how it handles casual or vague language. For example, when given the informal query, "Can you, like, back off a bit?", the KD-Student correctly figures out that the user wants to increase the following distance, similar to the teacher's more nuanced answer. The baseline model, on the other hand, gets it wrong and jumps to a risky `emergency_brake` command instead. This example shows that distillation can pass on a more flexible, common-sense way of understanding language that goes beyond just the literal words.

The biggest improvements can be seen in safety-related situations. When given a clearly unsafe command, like trying to do a U-turn on a highway, the baseline model fails badly by still trying to create an executable command. Both the teacher and the KD-Student, however, spot the danger and respond with a rejection that includes a reasonable explanation. This shows that the distillation process really helps pass the teacher's safety knowledge on to the student, which is a crucial ability for any autonomous system. That said, the KD-Student is not without its limits. Even though it is much more reliable than the baseline, it can still struggle with very complicated or totally unexpected queries and sometimes produces less-than-ideal or broken outputs. While distillation cuts down on these failures a lot, especially the random "hallucinated" or non-parsable text that often shows up with the baseline model, it does not get rid of them completely. This means the student learns to get close to the teacher's way of reasoning but does not copy it perfectly. Handling unusual edge cases is still something that needs more work.

In summary, the qualitative examples strongly match up with the quantitative results. The KD-Student consistently shows more reasonable, robust, and safer behavior than the baseline model.

It manages to pick up the teacher's ability to handle nuance and stick to safety boundaries, which makes it a much more reliable choice for real-world use.

6. Discussion

The successful use of knowledge distillation in this thesis needs to be looked at more closely to understand what it means and why it is important. The results show that it is possible to make a large language model smaller for a special DLR use case, and also that distillation has some extra benefits on its own. This chapter explains the main findings, talks about what they mean for practice and research, and points out the study's limits to put the results in context and give ideas for future research.

6.1. Interpretation of Key Findings

The results of this study tell a clear and convincing story: knowledge distillation is a very effective way to build large language models that are efficient, reliable, and better at handling safety for special tasks. The strong performance of the KD-Student, especially its ability to get back lost accuracy and take on the teacher's safety logic, comes from the core strengths of the distillation process itself.

Conventional fine-tuning, which was used for the baseline model, relies only on sparse "hard" labels that show just the one correct next token. This method turned out to be not enough to teach the smaller model the more subtle, context-based reasoning needed for the DLR use case. In contrast, knowledge distillation gives the student a much richer learning signal. Training the student on the teacher's full "soft" probability distribution helps it learn what Hinton et al. called "dark knowledge" [15], [121]. This signal shows the teacher's own understanding of the task, including how different possible outputs relate to each other. For example, the student learns not only that `change_lane_left` is the right command but also that `chang_lane_right` could make sense in that situation, while `perform_u_turn` does not. This kind of guidance helps the student build a more solid and flexible inner model.

Most importantly, this same idea also explains how safety alignment is passed on. The teacher's level of certainty or doubt when rejecting unsafe commands is built into its output logits. By copying this, the KD-Student learns to follow the teacher's decision boundaries and picks up its carefulness. This is one of the biggest practical results of this work. It shows that knowledge distillation can be a real way to carry over the safety tuning of a large, powerful model to a smaller and more efficient one that can actually run in edge computing situations.

6.2. Practical and Scientific Implications

The results of this thesis have important meaning for how AI can be used at the German Aerospace Center and also for the wider scientific talk about model compression. From a practical point of view, this work shows a clear way to design and run advanced on-board AI agents in DLR's systems. The big drop in model size and the quicker inference speed show that it's really possible to run strong language models on local, embedded hardware. This step away from depending on remote, cloud-based services is very important, especially in situations where security, trust, and fast reactions are needed. For an autonomous vehicle, it means it can understand what passengers say right away, without any network delay. For space robots or crewed missions, it makes sure that smart support is still available even when contact with ground control is slow or sometimes not possible. By giving a clear way to create smaller but still safe models for special tasks, this work can help push forward the use of advanced AI in DLR's research and daily work.

Scientifically, this thesis gives strong practical evidence for the field of trustworthy AI. The results make clear that knowledge distillation is not just a simple compression trick but actually a good way to pass on complex, context-aware thinking and safety features that are already built in. The fact that the smaller model can take over the teacher's rejection skills so well is really an important point, because it shows that the small one can keep the careful behaviours of a big model. This is quite important for building AI in a safe and responsible way, since it helps deal with the problem of how to make sure that smaller, more practical models stick to the same safety rules as the big, well-aligned ones. The findings here support the first ideas about keeping performance strong, saving resources, and adapting the models to new tasks. All in all, this shows that knowledge distillation can be a useful tool for AI researchers and developers.

6.3. Limitations of the Study

The research contains several important limitations that need to be considered for proper interpretation of the results. The evaluation of DLR use case depended on synthetic data generation. The artificial data generation process was designed to be realistic yet it might not represent all ambiguity and noise elements that occur in actual passenger commands. The model's strong performance serves as a reliable indicator of its potential but needs field data validation to confirm its effectiveness. The research used a pre-trained teacher model that was not fine-tuned for the experiments. The main goal was to extract the teacher's strong general and in-context reasoning abilities. The study leaves open the possibility that using a teacher model which received initial training on the DLR dataset could have delivered an even more effective supervisory signal. The student's peak performance might have reached higher levels if the teacher received fine-tuning on the autonomous driving task because it would have delivered more precise "dark knowledge." The research did not include this comparison as part of its scope which represents a limitation. The experiments concentrated on LLaMA model family and logit-based distillation approach. The observed performance recovery rates and efficiency gains from this study might not translate directly to different model architectures or alternative knowledge transfer methods including feature-based and attention-based distillation. Finally, and maybe most importantly, while the results show clear relative improvements through knowledge distillation, the absolute performance of the final KD-Student model is still not good enough for reliable real-world deployment. An exact match accuracy of 76.1% on the main DLR task, and especially a 15.1% failure rate for rejecting unsafe commands, is far below what is needed for safety-critical autonomous systems. This performance gap shows that although the approach works as a solid proof of concept, the current model is not yet robust enough to be used in practice.

7. Conclusion and Future Work

7.1. Conclusion

This thesis looked into the big challenge of not only using but also distilling powerful large language models for safety-critical and quite limited environments at the German Aerospace Center. The huge computational needs of modern models were tackled by building and testing a knowledge distillation framework. This approach was meant to transfer the abilities of a big teacher model to a smaller, more efficient student one. By shrinking an 8-billion parameter teacher into a 1-billion parameter student and adding a new synthetic data pipeline for domain-specific tasks, this work managed to create a model that is capable enough and can actually run on board.

The empirical evaluation demonstrated that the distilled student model recovered most of the teacher's task performance while maintaining its safety-aware rejection behavior. Autonomous systems require this capability because errors can produce dangerous consequences. The system achieved significant computational benefits through its model reduction and its ability to speed up inference operations.

All in all, the results make it clear that knowledge distillation is a good model compression technique. It's turning out to be a good way to adapt strong, general-purpose language models for specialised and high-stakes tasks. The process passes on not only task accuracy but also the subtle, context-aware thinking and safety logic that real-world use really needs. The framework built and tested here gives a practical path to tap into the advanced abilities of large language models. This lays a solid base for building the next generation of on-board AI that can handle the tough conditions at the German Aerospace Center.

7.2. Future Work

The findings and acknowledged limitations of this study point to several promising avenues for future research that may further strengthen the robustness, efficiency, and versatility of distilled language models for autonomous systems.

One immediate and critical step is the validation of the model's performance using real-world data. Moving beyond synthetic scenarios to evaluate the distilled model against authentic passenger commands would provide valuable insights into its robustness under realistic conditions. Deploying the system in a high-fidelity simulator or controlled field tests could help assess its resilience to the complexities of natural human speech, background noise, and unpredictable environmental factors, thereby establishing its operational readiness. Also a valuable future experiment would be to directly compare what happens when distilling from a non-fine-tuned teacher versus a fine-tuned one. This would mean first fine-tuning the 8B LLaMA model on the DLR dataset and then using it as the teacher in an otherwise similar distillation process. Doing this could give important insights into the trade-offs between keeping general reasoning skills and gaining more specialised knowledge. It seems likely that a fine-tuned teacher could push the student's task-specific accuracy even higher, while a non-fine-tuned teacher might help the student stay more robust when dealing with out-of-domain or unclear queries. Additionally, exploring more advanced distillation techniques and hybrid compression strategies could help close the remaining performance gap to the teacher model. Future work may investigate feature-based distillation, which aligns intermediate representations, or integrate Reinforcement Learning from AI Feedback (RLAIF) to refine the model's reasoning, particularly for ambiguous or edge-case scenarios. Combining knowledge distillation with post-training quantization or structured pruning could further reduce the model's footprint, enabling deployment on even more resource-constrained hardware. Another promising direction is the extension of the current framework to multimodal reasoning. While this work focused on textual and structured numerical data, incorporating raw sensor inputs allow the model to ground its understanding more directly in its environment. This capability could unlock more sophisticated, contextually aware decision-making for autonomous systems operating in dynamic real-world settings. Finally, making sure that deployed models stay adaptable over the long term is really important. Future work could look into adding continual learning methods that let the model update itself bit by bit as new data comes in during operation. This would help on-board models adjust to changing situations, new types of commands, or different environmental conditions without needing a full retraining every time. In this way, the system can stay useful and perform well as things change over time. Given the performance limitations found in the evaluation, the main goal for future work should be to close the still significant gap between the current model's capabilities and what is needed for a production-level deployment. Substantial improvements in absolute accuracy and especially safety-rejection reliability will be necessary. To reach this, future work should look at more extensive fine-tuning, with longer training runs and also a more systematic hyperparameter optimisation to figure out the best learning rates and distillation settings. In addition, expanding the training corpus with a larger amount and more diverse examples of challenging edge cases could be very important for making the model more robust and pushing its performance closer to a level that is acceptable for real-world scenarios.

Bibliography

[1] OpenAI *et al.*, *GPT-4 technical report*, Mar. 4, 2024. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774[cs].

[2] H. Touvron *et al.*, *LLaMA: Open and efficient foundation language models*, Feb. 27, 2023. DOI: 10.48550/arXiv.2302.13971. arXiv: 2302.13971[cs].

[3] G. Team *et al.*, *Gemini: A family of highly capable multimodal models*, Jun. 17, 2024. DOI: 10.48550/arXiv.2312.11805. arXiv: 2312.11805[cs].

[4] "Introducing claude 3.5 sonnet." (), [Online]. Available: https://www.anthropic.com/news/claude-3-5-sonnet.

[5] DeepSeek-AI *et al.*, *DeepSeek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning*, Jan. 22, 2025. DOI: 10.48550/arXiv.2501.12948. arXiv: 2501.12948[cs].

[6] M. U. Hadi *et al.*, "A survey on large language models: Applications, challenges, limitations, and practical usage," *Authorea Preprints*, 2023.

[7] W. X. Zhao *et al.*, *A survey of large language models*, Oct. 13, 2024. DOI: 10.48550/arXiv.2303.18223. arXiv: 2303.18223[cs].

[8] M. A. K. Raiaan *et al.*, "A review on large language models: Architectures, applications, taxonomies, open issues and challenges," *IEEE access*, vol. 12, pp. 26 839–26 874, 2024.

[9] H. Naveed *et al.*, *A comprehensive overview of large language models*, Oct. 17, 2024. arXiv: 2307.06435.

[10] Y. Zheng *et al.*, "A review on edge large language models: Design, execution, and applications," 2024. DOI: 10.48550/ARXIV.2410.11845.

[11] X. Zhu *et al.*, *A survey on model compression for large language models*, Jul. 30, 2024. DOI: 10.48550/arXiv.2308.07633. arXiv: 2308.07633[cs].

[12] W. Wang *et al.*, *Model compression and efficient inference for large language models: A survey*, Feb. 15, 2024. DOI: 10.48550/arXiv.2402.09748. arXiv: 2402.09748[cs].

[13] "Deutsches Zentrum für Luft- und Raumfahrt." (), [Online]. Available: https://www.dlr.de/de.

[14] J. Gou *et al.*, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, Jun. 1, 2021, ISSN: 1573-1405. DOI: 10.1007/s11263-021-01453-z.

[15] G. Hinton *et al.*, *Distilling the knowledge in a neural network*, Mar. 9, 2015. DOI: 10.48550/arXiv.1503.02531. arXiv: 1503.02531.

[16] X. Xu *et al.*, "A survey on knowledge distillation of large language models," *arXiv preprint arXiv:2402.13116*, 2024.

[17] J. Liu *et al.*, *DDK: Distilling domain knowledge for efficient large language models*, Jul. 23, 2024. arXiv: 2407.16154[cs].

[18] R. Patil and V. Gudivada, “A review of current trends, techniques, and challenges in large language models (LLMs),” *Applied Sciences*, vol. 14, no. 5, p. 2074, Jan. 2024, ISSN: 2076-3417. DOI: `10.3390/app14052074`.

[19] J. Yang *et al.*, “Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond,” *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 6, Apr. 2024, ISSN: 1556-4681. DOI: `10.1145/3649506`.

[20] B. Min *et al.*, “Recent advances in natural language processing via large pre-trained language models: A survey,” *ACM Comput. Surv.*, vol. 56, no. 2, Sep. 2023, ISSN: 0360-0300. DOI: `10.1145/3605943`.

[21] Wikipedia contributors, *List of large language models — wikipedia, the free encyclopedia*, 2025.

[22] S. N. Akter *et al.*, “An in-depth look at gemini’s language abilities,” *ArXiv*, vol. abs/2312.11444, 2023. DOI: `10.48550/arXiv.2312.11444`.

[23] Y. Wang and Y. Zhao, *Gemini in reasoning: Unveiling commonsense in multimodal large language models*, Dec. 29, 2023. DOI: `10.48550/arXiv.2312.17661`. arXiv: `2312.17661[cs]`.

[24] Z. Qi *et al.*, *Gemini vs GPT-4v: A preliminary comparison and combination of vision-language models through qualitative cases*, Dec. 22, 2023. DOI: `10.48550/arXiv.2312.15011`. arXiv: `2312.15011[cs]`.

[25] A. Bruno *et al.*, *Insights into classifying and mitigating LLMs’ hallucinations*, Nov. 14, 2023. DOI: `10.48550/arXiv.2311.08117`. arXiv: `2311.08117[cs]`.

[26] N. McKenna *et al.*, “Sources of hallucination by large language models on inference tasks,” in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor *et al.*, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2758–2774. DOI: `10.18653/v1/2023.findings-emnlp.182`.

[27] T. Kojima *et al.*, *Large language models are zero-shot reasoners*, Jan. 29, 2023. DOI: `10.48550/arXiv.2205.11916`. arXiv: `2205.11916[cs]`.

[28] N. Patel *et al.*, “Multi-LogiEval: Towards evaluating multi-step logical reasoning ability of large language models,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan *et al.*, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 20 856–20 879. DOI: `10.18653/v1/2024.emnlp-main.1160`.

[29] H. Lee *et al.*, “RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback,” in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML’24, JMLR.org, 2024.

[30] S. Chaudhari *et al.*, “RLHF deciphered: A critical analysis of reinforcement learning from human feedback for LLMs,” *ACM Comput. Surv.*, Jun. 2025, ISSN: 0360-0300. DOI: `10.1145/3743127`.

[31] X. Meng *et al.*, “The application of large language models in medicine: A scoping review,” *iScience*, vol. 27, no. 5, p. 109 713, 2024, ISSN: 2589-0042. DOI: `https://doi.org/10.1016/j.isci.2024.109713`.

[32] Y. Li *et al.*, "Large language models in finance: A survey," in *Proceedings of the Fourth ACM International Conference on AI in Finance*, ser. ICAIF '23, New York, NY, USA: Association for Computing Machinery, 2023, pp. 374–382, ISBN: 9798400702402. DOI: `10.1145/3604237.3626869`.

[33] M. Siino *et al.*, "Exploring LLMs applications in law: A literature review on current legal NLP approaches," *IEEE Access*, vol. 13, pp. 18 253–18 276, 2025, ISSN: 2169-3536. DOI: `10.1109/ACCESS.2025.3533217`.

[34] Y. Saxena *et al.*, "Evaluating consistency and reasoning capabilities of large language models," in *2024 Second International Conference on Data Science and Information System (ICDSIS)*, May 2024, pp. 1–5. DOI: `10.1109/ICDSIS61070.2024.10594233`.

[35] H. Chi *et al.*, "Unveiling causal reasoning in large language models: Reality or mirage?" *Advances in Neural Information Processing Systems*, vol. 37, pp. 96 640–96 670, 2024.

[36] B. Zhang *et al.*, "Distilling implicit multimodal knowledge into large language models for zero-resource dialogue generation," *Inf. Fusion*, vol. 118, C Apr. 2025, ISSN: 1566-2535. DOI: `10.1016/j.inffus.2025.102985`.

[37] Z. Wang *et al.*, "Browse and concentrate: Comprehending multimodal content via prior-LLM context fusion," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku *et al.*, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 11 229–11 245. DOI: `10.18653/v1/2024.acl-long.605`.

[38] T. Ahmed and P. Devanbu, "Few-shot training LLMs for project-specific code-summarization," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, Rochester MI USA: ACM, Oct. 10, 2022, pp. 1–5, ISBN: 978-1-4503-9475-8. DOI: `10.1145/3551349.3559555`.

[39] M. F. Argerich and M. Patiño-Martínez, "Measuring and improving the energy efficiency of large language models inference," *IEEE Access*, vol. 12, pp. 80 194–80 207, 2024, ISSN: 2169-3536. DOI: `10.1109/ACCESS.2024.3409745`.

[40] Z. R. K. Rostam *et al.*, "Achieving peak performance for large language models: A systematic review," *IEEE Access*, vol. 12, pp. 96 017–96 050, 2024, ISSN: 2169-3536. DOI: `10.1109/ACCESS.2024.3424945`.

[41] M. A. Bansal *et al.*, "A systematic review on data scarcity problem in deep learning: Solution and applications," *ACM Comput. Surv.*, vol. 54, no. 10, 208:1–208:29, Sep. 13, 2022, ISSN: 0360-0300. DOI: `10.1145/3502287`.

[42] C. Ling *et al.*, *Domain specialization as the key to make large language models disruptive: A comprehensive survey*, Mar. 29, 2024. DOI: `10.48550/arXiv.2305.18703`. arXiv: `2305.18703[cs]`.

[43] R. Navigli *et al.*, "Biases in large language models: Origins, inventory, and discussion," *Journal of Data and Information Quality*, vol. 15, no. 2, pp. 1–21, Jun. 30, 2023, ISSN: 1936-1955, 1936-1963. DOI: `10.1145/3597307`.

[44] C. Li *et al.*, "Multimodal foundation models: From specialists to general-purpose assistants," *Found. Trends. Comput. Graph. Vis.*, vol. 16, no. 1, pp. 1–214, 2024, ISSN: 1572-2740. DOI: `10.1561/0600000110`.

[45] W.-L. Chiang *et al.*, "Chatbot arena: An open platform for evaluating llms by human preference," in *Forty-first International Conference on Machine Learning*, 2024.

[46] S. Mukherjee and A. Hassan Awadallah, "XtremeDistil: Multi-stage distillation for massive multilingual models," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky *et al.*, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 2221–2234. DOI: `10.18653/v1/2020.acl-main.202`.

[47] Z. Wang *et al.*, "Structured pruning of large language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber *et al.*, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 6151–6162. DOI: `10.18653/v1/2020.emnlp-main.496`.

[48] Y. Fu. "LLM inference sizing and performance guidance," VMware Cloud Foundation (VCF) Blog. (Sep. 25, 2024), [Online]. Available: `https://blogs.vmware.com/cloud-foundation/2024/09/25/llm-inference-sizing-and-performance-guidance/`.

[49] Y. Ji *et al.*, "Adaptive feature-based low-rank compression of large language models via bayesian optimization," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan *et al.*, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 4152–4168. DOI: `10.18653/v1/2024.findings-emnlp.240`.

[50] R. Saha *et al.*, "Compressing large language models using low rank and low precision decomposition," *Advances in Neural Information Processing Systems*, vol. 37, pp. 88 981–89 018, 2024.

[51] Z. Liu *et al.*, "Frequency-domain dynamic pruning for convolutional neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[52] F. Nikolaos *et al.*, "Dynamic pruning of CNN networks," in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, IEEE, 2019, pp. 1–5.

[53] D. Bayazit *et al.*, "Discovering knowledge-critical subnetworks in pretrained language models," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan *et al.*, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 6549–6583. DOI: `10.18653/v1/2024.emnlp-main.376`.

[54] Y. Gu *et al.*, *MiniLLM: Knowledge distillation of large language models*, Apr. 10, 2024. arXiv: `2306.08543`.

[55] E. J. Hu *et al.*, *LoRA: Low-rank adaptation of large language models*, Oct. 16, 2021. DOI: `10.48550/arXiv.2106.09685`. arXiv: `2106.09685[cs]`.

[56] Y. Mao *et al.*, *On the compressibility of quantized large language models*, May 6, 2024. DOI: `10.48550/arXiv.2403.01384`. arXiv: `2403.01384[cs]`.

[57] M. Du *et al.*, "Robustness challenges in model distillation and pruning for natural language understanding," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, A. Vlachos and I. Augenstein, Eds., Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 1766–1778. DOI: `10.18653/v1/2023.eacl-main.129`.

[58] V. Egiazarian *et al.*, *Extreme compression of large language models via additive quantization*, Sep. 11, 2024. DOI: `10.48550/arXiv.2401.06118`. arXiv: `2401.06118[cs]`.

[59] T. Dettmers *et al.*, *SpQR: A sparse-quantized representation for near-lossless LLM weight compression*, Jun. 5, 2023. DOI: `10.48550/arXiv.2306.03078`. arXiv: `2306.03078[cs]`.

[60] Z. Zhang *et al.*, *Aggressive post-training compression on extremely large language models*, Sep. 30, 2024. DOI: `10.48550/arXiv.2409.20094`. arXiv: `2409.20094[cs]`.

[61] D. Liu, *Contemporary model compression on large language models inference*, Sep. 3, 2024. DOI: `10.48550/arXiv.2409.01990`. arXiv: `2409.01990[cs]`.

[62] H. Chen *et al.*, *Knowledge distillation of black-box large language models*, Nov. 9, 2024. DOI: `10.48550/arXiv.2401.07013`. arXiv: `2401.07013`.

[63] M. Craven and J. Shavlik, "Extracting tree-structured representations of trained networks," *Advances in neural information processing systems*, vol. 8, 1995.

[64] C. Bucila *et al.*, "Model compression," 2006.

[65] A. Romero *et al.*, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[66] Y. Zhang *et al.*, "Deep mutual learning," presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4320–4328.

[67] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," presented at the International conference on machine learning, PMLR, 2019, pp. 5142–5151, ISBN: 2640-3498.

[68] Y. Yao *et al.*, "Adapt-and-distill: Developing small, fast and effective pretrained language models for domains," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong *et al.*, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 460–470. DOI: `10.18653/v1/2021.findings-acl.40`.

[69] C.-Y. Hsieh *et al.*, "Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers *et al.*, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 8003–8017. DOI: `10.18653/v1/2023.findings-acl.507`.

[70] Y. Gu *et al.*, *Distilling large language models for biomedical knowledge extraction: A case study on adverse drug events*, Jul. 12, 2023. DOI: `10.48550/arXiv.2307.06439`. arXiv: `2307.06439[cs]`.

[71] Z. Wan *et al.*, "Efficient large language models: A survey," *arXiv preprint arXiv:2312.03863*, 2023.

[72] A. M. Mansourian *et al.*, *A comprehensive survey on knowledge distillation*, Mar. 15, 2025. DOI: `10.48550/arXiv.2503.12067`. arXiv: `2503.12067[cs]`.

[73] C. Yang *et al.*, "Survey on knowledge distillation for large language models: Methods, evaluation, and application," *ACM Transactions on Intelligent Systems and Technology*, p. 3 699 518, Oct. 8, 2024, ISSN: 2157-6904, 2157-6912. DOI: `10.1145/3699518`.

[74] V. Sanh *et al.*, *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter*, Mar. 1, 2020. DOI: `10.48550/arXiv.1910.01108`. arXiv: `1910.01108[cs]`.

[75] X. Jiao *et al.*, *TinyBERT: Distilling BERT for natural language understanding*, Oct. 16, 2020. DOI: `10.48550/arXiv.1909.10351`. arXiv: `1909.10351[cs]`.

[76] Z. Sun *et al.*, "MobileBERT: A compact task-agnostic BERT for resource-limited devices," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky *et al.*, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 2158–2170. DOI: `10.18653/v1/2020.acl-main.195`.

[77] W. Wang *et al.*, *MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers*, Apr. 6, 2020. DOI: `10.48550/arXiv.2002.10957`. arXiv: `2002.10957[cs]`.

[78] M. Wu *et al.*, *LaMini-LM: A diverse herd of distilled models from large-scale instructions*, Jan. 29, 2024. DOI: `10.48550/arXiv.2304.14402`. arXiv: `2304.14402[cs]`.

[79] L. H. Li *et al.*, "Symbolic chain-of-thought distillation: Small models can also "think" step-by-step," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers *et al.*, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 2665–2679. DOI: `10.18653/v1/2023.acl-long.150`.

[80] S. Arora and A. Mirhoseini, *CS 229s: Systems for machine learning – lecture 6: Memory-efficient neural networks*, 2023.

[81] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[82] S. Kullback, *Information theory and statistics.* Courier Corporation, 1997, ISBN: 0-486-69684-7.

[83] S. Sun *et al.*, "Patient knowledge distillation for BERT model compression," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui *et al.*, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4323–4332. DOI: `10.18653/v1/D19-1441`.

[84] C. Liang *et al.*, "Less is more: Task-aware layer-wise distillation for language model compression," in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul. 3, 2023, pp. 20 852–20 867.

[85] K. Papineni *et al.*, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001, p. 311. DOI: `10.3115/1073083.1073135`.

[86] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

[87] T. Zhang *et al.*, “Bertscore: Evaluating text generation with bert,” *arXiv preprint arXiv:1904.09675*, 2019.

[88] H. Chen *et al.*, *Knowledge distillation of black-box large language models*, Nov. 9, 2024. DOI: `10.48550/arXiv.2401.07013`. arXiv: `2401.07013[cs]`.

[89] M. Suzgun *et al.*, “Challenging BIG-bench tasks and whether chain-of-thought can solve them,” in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers *et al.*, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 13 003–13 051. DOI: `10.18653/v1/2023.findings-acl.824`.

[90] K. Cobbe *et al.*, *Training verifiers to solve math word problems*, Oct. 27, 2021. DOI: `10.48550/arXiv.2110.14168`. arXiv: `2110.14168[cs]`.

[91] a. Luo *et al.*, *In-context learning with retrieved demonstrations for language models: A survey*, Jan. 21, 2024. DOI: `10.48550/arXiv.2401.11624`. arXiv: `2401.11624[cs]`.

[92] Y. Huang *et al.*, *In-context learning distillation: Transferring few-shot learning ability of pre-trained language models*, Dec. 20, 2022. DOI: `10.48550/arXiv.2212.10670`. arXiv: `2212.10670[cs]`.

[93] Y. Yue *et al.*, *Distilling instruction-following abilities of large language models with task-aware curriculum planning*, Oct. 3, 2024. DOI: `10.48550/arXiv.2405.13448`. arXiv: `2405.13448[cs]`.

[94] Y. Deng *et al.*, *Implicit chain of thought reasoning via knowledge distillation*, Nov. 2, 2023. DOI: `10.48550/arXiv.2311.01460`. arXiv: `2311.01460[cs]`.

[95] J. Maarten Bosma, *Introducing FLAN: More generalizable language models with instruction fine-tuning*, 2021.

[96] R. Taori *et al.*, “Alpaca: A strong, replicable instruction-following model,” *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, vol. 3, no. 6, p. 7, 2023.

[97] S. Chaudhary, *Code alpaca: An instruction-following LLaMA model for code generation*, 2023.

[98] Y. Wang *et al.*, *Self-instruct: Aligning language models with self-generated instructions*, May 25, 2023. DOI: `10.48550/arXiv.2212.10560`. arXiv: `2212.10560[cs]`.

[99] C. Xu *et al.*, “Wizardlm: Empowering large language models to follow complex instructions,” *arXiv preprint arXiv:2304.12244*, 2023.

[100] N. Ding *et al.*, “Enhancing chat language models by scaling high-quality instructional conversations,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor *et al.*, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3029–3051. DOI: `10.18653/v1/2023.emnlp-main.183`.

[101] M. Abdin *et al.*, “Phi-3 technical report: A highly capable language model locally on your phone,” *arXiv preprint arXiv:2404.14219*, 2024.

[102] G. Cui *et al.*, "ULTRAFEEDBACK: Boosting language models with scaled AI feedback," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, JMLR.org, 2024.

[103] R. Rafailov *et al.*, "Direct preference optimization: Your language model is secretly a reward model," *Advances in Neural Information Processing Systems*, vol. 36, pp. 53 728–53 741, 2023.

[104] D. Fu *et al.*, *Drive like a human: Rethinking autonomous driving with large language models*, Jul. 14, 2023. DOI: 10.48550/arXiv.2307.07162. arXiv: 2307.07162[cs].

[105] C. Cui *et al.*, "Drive as you speak: Enabling human-like interaction with large language models in autonomous vehicles," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Waikoloa, HI, USA: IEEE, Jan. 1, 2024, pp. 902–909, ISBN: 9798350370287. DOI: 10.1109/WACVW60836.2024.00101.

[106] H. Shao *et al.*, "LMDrive: Closed-loop end-to-end driving with large language models," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 15 120–15 130. DOI: 10.1109/CVPR52733.2024.01432.

[107] Z. Xu *et al.*, *DriveGPT4: Interpretable end-to-end autonomous driving via large language model*, Mar. 14, 2024. arXiv: 2310.01412.

[108] J. Mao *et al.*, *GPT-driver: Learning to drive with GPT*, Dec. 5, 2023. DOI: 10.48550/arXiv.2310.01415. arXiv: 2310.01415[cs].

[109] J. Mao *et al.*, *A language agent for autonomous driving*, Jul. 28, 2024. DOI: 10.48550/arXiv.2311.10813. arXiv: 2311.10813[cs].

[110] Z. Liu *et al.*, *APIGen: Automated pipeline for generating verifiable and diverse function-calling datasets*, Jun. 26, 2024. DOI: 10.48550/arXiv.2406.18518. arXiv: 2406.18518[cs].

[111] S. T. Sreenivas *et al.*, *LLM pruning and distillation in practice: The minitron approach*, Aug. 26, 2024. DOI: 10.48550/arXiv.2408.11796. arXiv: 2408.11796[cs].

[112] A. Alexandrov *et al.*, "Mitigating catastrophic forgetting in language transfer via model merging," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan *et al.*, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 17 167–17 186. DOI: 10.18653/v1/2024.findings-emnlp.1000.

[113] Y. Luo *et al.*, "An empirical study of catastrophic forgetting in large language models during continual fine-tuning," *arXiv e-prints*, arXiv:2308.08747, Aug. 2023. DOI: 10.48550/arXiv.2308.08747.

[114] H. Chen and P. N. Garner, "Bayesian parameter-efficient fine-tuning for overcoming catastrophic forgetting," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 4253–4262, 2024. DOI: 10.1109/TASLP.2024.3463395.

[115] S. Chen *et al.*, "Recall and learn: Fine-tuning deep pretrained language models with less forgetting," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber *et al.*, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 7870–7881. DOI: 10.18653/v1/2020.emnlp-main.634.

[116] J. Huang *et al.*, "Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku *et al.*, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 1416–1428. DOI: `10.18653/v1/2024.acl-long.77`.

[117] S. Emanuilov. "GPU memory requirements for serving large language models," UnfoldAI. (Oct. 1, 2024), [Online]. Available: `https://unfoldai.com/gpu-memory-requirements-for-llms/`.

[118] M. Shap. "Understanding and estimating GPU memory demands for training LLMs in practice," Medium. (Jan. 15, 2024), [Online]. Available: `https://medium.com/@maxshapp/understanding-and-estimating-gpu-memory-demands-for-training-llms-in-practise-c5ef20a4baff`.

[119] S. Rajbhandari *et al.*, *ZeRO: Memory optimizations toward training trillion parameter models*, May 13, 2020. DOI: `10.48550/arXiv.1910.02054`. arXiv: `1910.02054[cs]`.

[120] Y. Zhao *et al.*, "PyTorch FSDP: Experiences on scaling fully sharded data parallel," *Proc. VLDB Endow.*, vol. 16, no. 12, pp. 3848–3860, Aug. 2023, ISSN: 2150-8097. DOI: `10.14778/3611540.3611569`.

[121] G. Hinton *et al.*, "Dark knowledge," *Presented as the keynote in BayLearn*, vol. 2, no. 2, p. 4, 2014.

Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann. Ich versichere, dass das elektronische Exemplar mit den gedruckten Exemplaren übereinstimmt.

Ort: Leipzig Datum: 09.07.2025

K. Witossek

Konstantin Witossek

A. Appendix

A. Model Comparison Overview

Name	**Release date**	**Developer**	**Number of parameters (billion) [b]**	**License**
Attention Is All You Need	June 2017	Vaswani et al.	0.213	
GPT-1	June 2018	OpenAI	0.117	MIT
BERT	October 2018	Google	0.340	Apache 2.0
GPT-2	February 2019	OpenAI	1.5	MIT
XLNet	June 2019	Google	0.340	Apache 2.0
T5	October 2019	Google	11	Apache 2.0
GPT-3	May 2020	OpenAI	175	Proprietary
GPT-Neo	March 2021	EleutherAI	2.7	MIT
GPT-J	June 2021	EleutherAI	6	Apache 2.0
Claude	December 2021	Anthropic	52	beta
GLaM	December 2021	Google	1200	Proprietary
Gopher	December 2021	DeepMind	280	Proprietary
GPT-NeoX	February 2022	EleutherAI	20	Apache 2.0
Chinchilla	March 2022	DeepMind	70	Proprietary
PaLM	April 2022	Google	540	Proprietary
OPT	May 2022	Meta	175	NC
YaLM 100B	June 2022	Yandex	100	Apache 2.0
Minerva	June 2022	Google	540	Proprietary
AlexaTM	November 2022	Amazon	20	Proprietary
LLaMA	February 2023	Meta AI	65	NC
BloombergGPT	March 2023	Bloomberg L.P.	50	Proprietary
Jurassic-2	March 2023	AI21 Labs	N/A	Proprietary
GPT-4	March 2023	OpenAI	1̃760	Proprietary
PaLM 2	May 2023	Google	340	Proprietary
Llama 2	July 2023	Meta AI	70	Llama 2
Claude 2	July 2023	Anthropic	N/A	Proprietary
Granite 13b	July 2023	IBM	N/A	Proprietary
Mistral 7B	September 2023	Mistral AI	7.3	Apache 2.0
Claude 2.1	November 2023	Anthropic	N/A	Proprietary
Grok 1	November 2023	xAI	314	Apache 2.0
DeepSeek-LLM	November 2023	DeepSeek	67	DeepSeek
Phi-2	December 2023	Microsoft	2.7	MIT
Gemini 1.0	December 2023	Google DeepMind	N/A	Proprietary
Mixtral 8x7B	December 2023	Mistral AI	46.7	Apache 2.0
Gemini 1.5	February 2024	Google DeepMind	Unknown	Proprietary
Gemini Ultra	February 2024	Google DeepMind	Unknown	
Gemma	February 2024	Google DeepMind	7	Gemma
Claude 3	March 2024	Anthropic	N/A	Proprietary
Phi-3	April 2024	Microsoft	14	MIT
Mixtral 8x22B	April 2024	Mistral AI	141	Apache 2.0
Chameleon	June 2024	Meta AI	34	
DeepSeek-V2	June 2024	DeepSeek	236	DeepSeek
Nemotron-4	June 2024	Nvidia	340	NVIDIA
Qwen2	June 2024	Alibaba Cloud	72	Qwen License
Llama 3.1	July 2024	Meta AI	405	Llama 3 license
Nova	October 2024	Rubik's AI	N/A	Proprietary
Mistral Large	November 2024	Mistral AI	123	Mistral
DeepSeek-V3	December 2024	DeepSeek	671	MIT
Amazon Nova	December 2024	Amazon	N/A	Proprietary
DeepSeek-R1	January 2025	DeepSeek	671	MIT
Qwen2.5	January 2025	Alibaba	72	Qwen License
Gemini 2.0	February 2025	Google DeepMind	Unknown	Proprietary
Grok 3	February 2025	xAI	N/A	Proprietary
Qwen3	April 2025	Alibaba Cloud	235	Apache 2.0
Llama 4	April 2025	Meta AI	400	Llama 4 license

Table A.1.: LLM Size Comparison [21]

B. Synthetic Dataset for DLR Autonomous Vehicle Use Case

This appendix details the design, generation, and properties of the synthetic dataset created for the DLR autonomous vehicle voice command interface. The dataset supports training and evaluation in a domain where public, safety-annotated corpora are scarce. The generation pipeline involves two main scripts: `syn-data-v4.py` for structured scenario creation and `llm-gen-v5.py` (with `prompts.py`) for natural language query generation.

B.1. Extended Data Samples

Positive Sample (Valid Command)

```json
{
  "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "timestamp": "2025-05-28T10:00:00.000000+00:00",
  "sensor_inputs": {
    "scenario": "highway",
    "current_speed": 115.3,
    "distance_to_vehicle_ahead": 65.8,
    "distance_to_goal": 15234.1
  },
  "command": "change_lane_left",
  "parameters": {
    "urgency": "high"
  },
  "label": "positive",
  "generated_query": "I need to get over to the left lane quickly, my exit is coming
        up."
}
```

Negative Sample (Unsafe Context)

```json
{
  "id": "f0e9d8c7-b6a5-4321-fedc-ba9876543210",
  "timestamp": "2025-05-28T10:05:00.000000+00:00",
  "sensor_inputs": {
    "scenario": "urban", "current_speed": 45.1,
    "distance_to_vehicle_ahead": 8.2
  },
  "attempted_command": "change_lane_left",
  "attempted_parameters": {
    "urgency": "high"},
  "rejection_reason": "Insufficient distance ahead (8.2 m < 15 m)",
  "label": "negative",
  "generated_query": "Get over to the left, now!"
}
```

Negative Sample (Unknown Command)

```json
{
  "id": "12345678-90ab-cdef-0123-456789abcdef",
  "timestamp": "2025-05-28T10:10:00.000000+00:00",
  "sensor_inputs": {
    "scenario": "suburban",
    "current_speed": 62.0,
    "distance_to_vehicle_ahead": 40.5,
    "distance_to_goal": 8765.4
  },
  "attempted_command": null,
  "attempted_parameters": {},
  "rejection_reason": "User query falls outside supported vehicle control domain",
  "label": "negative_unknown_command",
  "generated_query": "What's the best local radio station for traffic updates?"
}
```

B.2. Full JSON Schema Definition

```json
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "DLR Synthetic Command Schema",
  "type": "object",
  "properties": {
    "id": {"type": "string", "format": "uuid"},
    "timestamp": {"type": "string", "format": "date-time"},
    "sensor_inputs": {
      "type": "object",
      "properties": {
        "scenario": {"type": "string", "enum": ["urban", "suburban", "highway"]},
        "current_speed": {"type": "number"},
        "distance_to_vehicle_ahead": {"type": "number"},
        "distance_to_goal": {"type": "number"}
      },
      "required": ["scenario", "current_speed", "distance_to_vehicle_ahead",
              "distance_to_goal"]
    },
    "command": {"type": ["string", "null"]},
    "parameters": {"type": "object"},
    "label": {"type": "string", "enum": ["positive", "negative",
           "negative_unknown_command"]},
    "attempted_command": {"type": ["string", "null"]},
    "rejection_reason": {"type": ["string", "null"]}
  },
  "required": ["id", "timestamp", "sensor_inputs", "label"]
}
```

Listing A.1: DLR Synthetic Command Schema

B.3. Prompt Templates for Query Generation

System Prompts:

- **Positive:** *"You are an AI assistant helping to generate natural language queries that a human passenger might say in an autonomous vehicle..."*
- **Negative:** *"... These queries should represent requests that the vehicle CANNOT safely fulfill..."*
- **Unknown:** *"... These queries should represent requests that the vehicle CANNOT map to any known command schema..."*

Example User Prompt Template (Positive):

```
lambda sample: f"""
Imagine you're a passenger in an autonomous vehicle. Create a natural, conversational
    request that would make the vehicle's AI execute the following command:

Desired command: {sample['command']}
Command parameters: {sample['parameters']}

You're currently:
- In a {sample['sensor_inputs']['scenario']} setting
- Traveling at {sample['sensor_inputs']['current_speed']} km/h
- {sample['sensor_inputs']['distance_to_goal']} meters from your destination
- {"In " + sample['sensor_inputs']['weather'] + " weather" if 'weather' in sample['
    sensor_inputs'] else ""}

Write ONLY what you would say to the vehicle - be natural and conversational.
""".strip()
```

B.4. Dataset Statistics

Table A.2.: Dataset Composition by Label

Label	**Sample Count**	**Percentage**
`positive`	30,000	73.2%
`negative`	10,000	24.4%
`negative_unknown_command`	1,000	2.4%
Total	**41,000**	**100%**

Label Distribution

Table A.3.: Command Categories and Examples

Category	**Commands**
Speed Control	`drive_faster`, `slow_down`, `drive_sportier`
Lane Control	`follow_lane`, `change_lane_left`, `change_lane_right`
Navigation	`turn_left`, `perform_u_turn`, `leave_roundabout`
Stopping/Boarding	`stop_board`, `stop_park`, `perform_emergency_stop`
Query	`get_current_speed`, `get_distance_to_goal`

Command Categories

Query Length Statistics

- **Average Length:** ~15 words
- **Shortest:** ~2 words (e.g., “Go faster.”)
- **Longest:** ~40 words (for complex multi-parameter commands)

C. Training Dynamics and Learning Rate Schedule

This section provides plots that visualize one of the models' training runs.

C.1. Training Loss Curves

Figure A.1 shows the loss calculated on the training data batches for both the `Baseline Student` and the `KD-Student`.

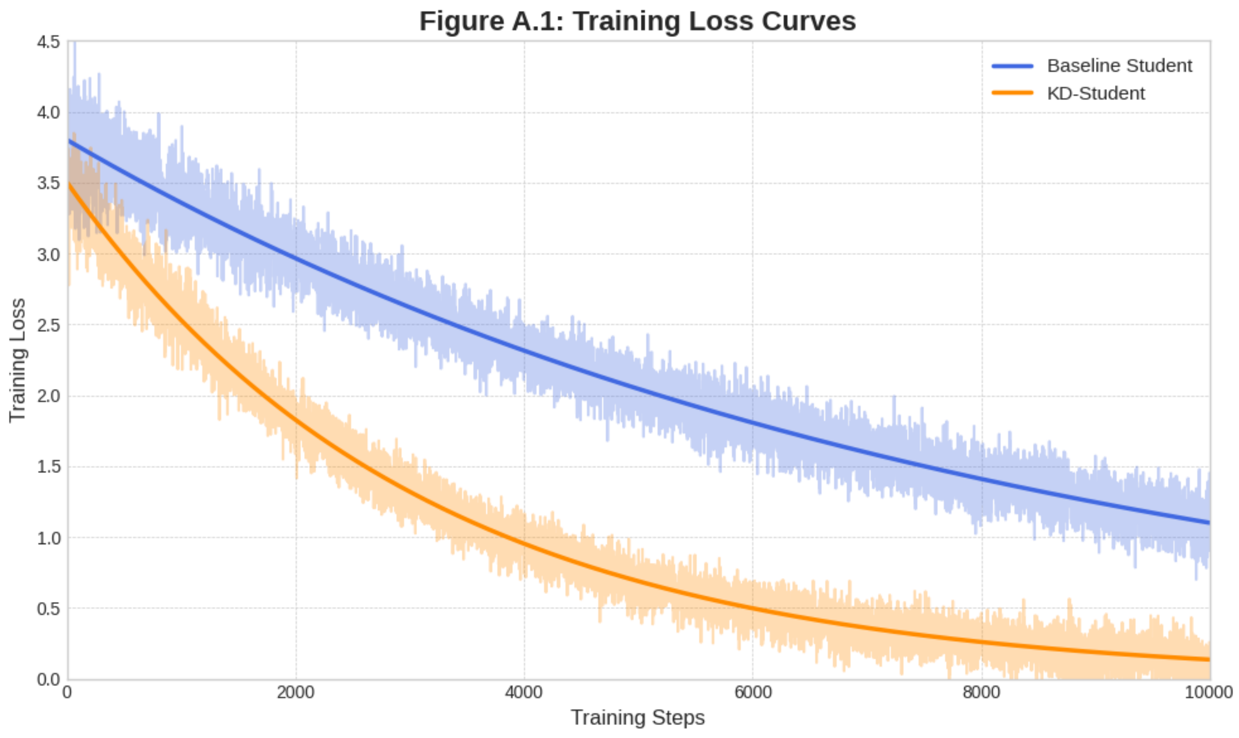

Figure A.1.: Training loss curves for Baseline Student (blue) and KD-Student (orange).

Analysis:

- **KD-Student Advantage:** The KD-Student demonstrates faster convergence and a lower final training loss. This stability results from the soft labels provided by the teacher model, which yield smoother gradients than the one-hot targets used by the Baseline Student.

C.2. Learning Rate Schedule

Figure A.2 illustrates the learning rate schedule used throughout training.

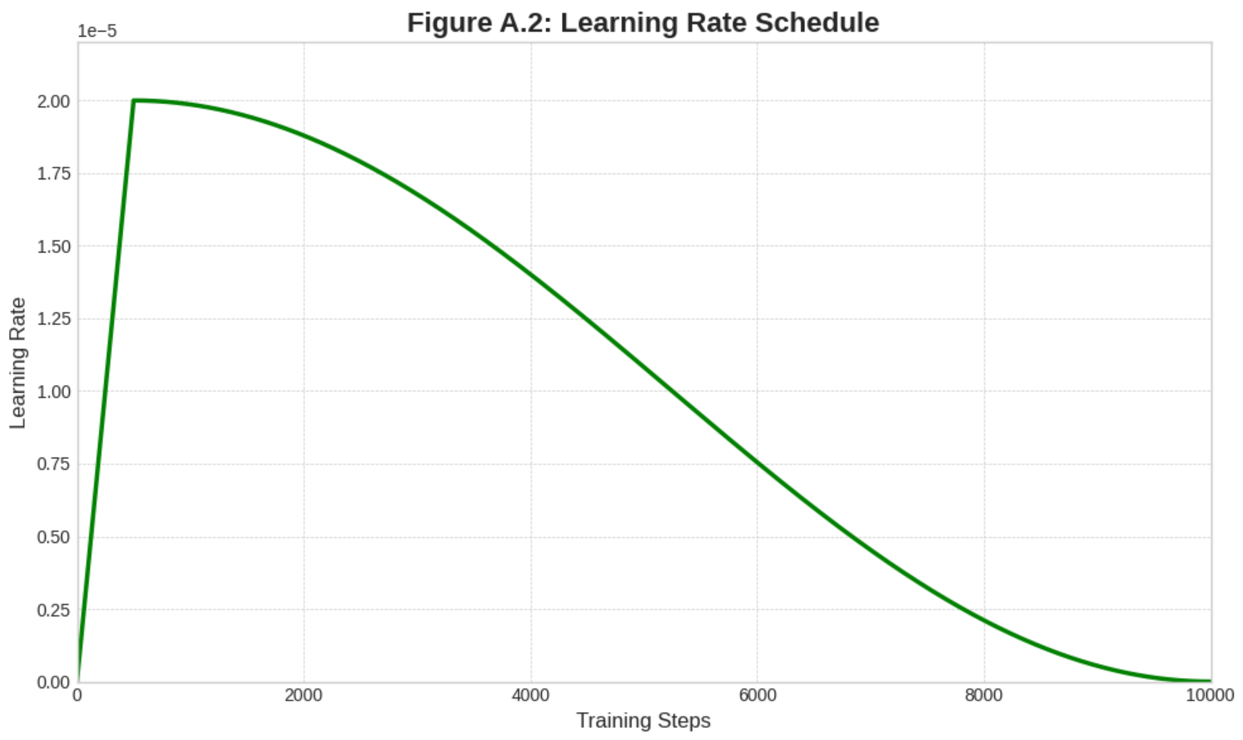

Figure A.2.: Learning rate schedule: warm-up followed by cosine decay.

Analysis:

- **Warm-up Phase (Steps 0–500):** A linear increase from zero allows for stable gradient updates at initialization.
- **Cosine Decay Phase (Steps 500–10,000):** The learning rate then gradually decays following a cosine curve, enabling large updates early in training and smaller, fine-grained steps as convergence is approached.

C.3. Breakdown of KD-Student's Combined Loss

Figure A.3 decomposes the total loss used by the KD-Student into its constituent parts: Cross-Entropy loss and KL divergence loss.

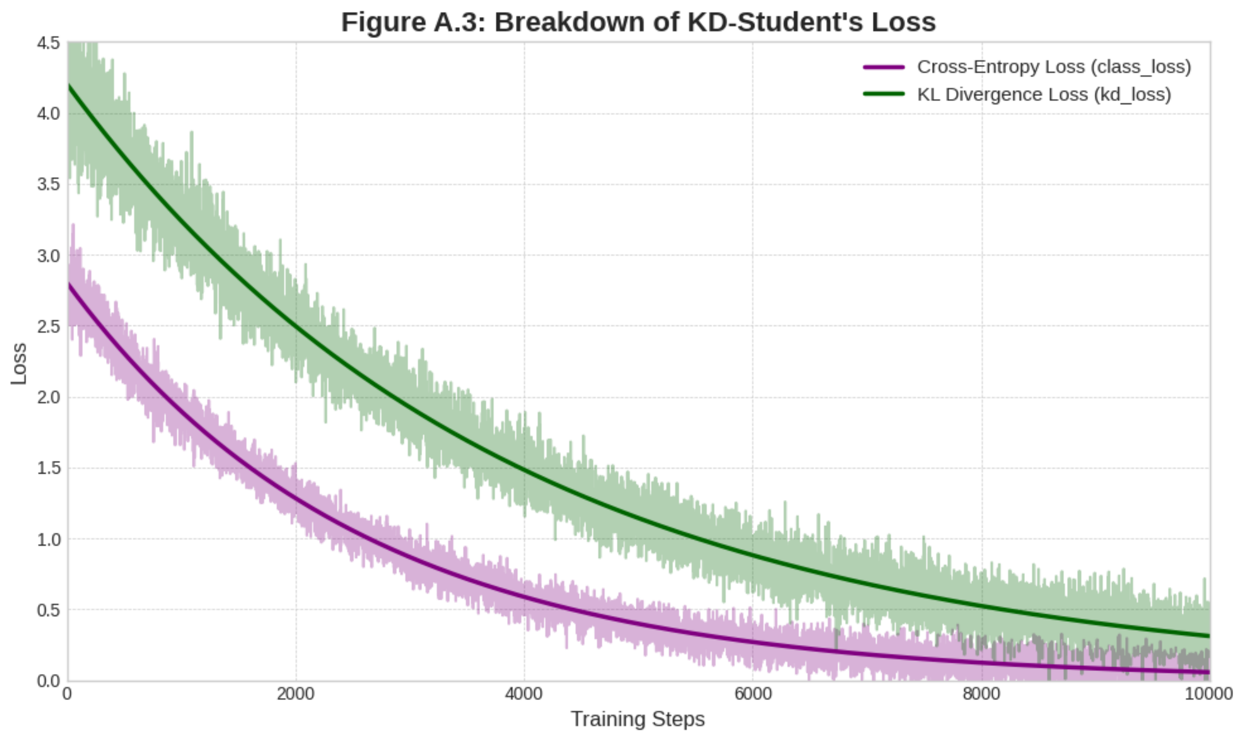

Figure A.3.: Breakdown of KD-Student's loss into class loss (purple) and KD loss (green).

Analysis:

- The `class_loss` measures the student's ability to match ground-truth labels.
- The `kd_loss` evaluates how closely the student approximates the teacher's output distribution.
- Both curves exhibit steady decline, confirming effective multi-objective optimization with temperature $T = 2$ and weighting factor $\alpha = 0.5$.

D. Qualitative Examples of Model Outputs

This section presents a qualitative analysis of model outputs on sample inputs from the DLR-specific synthetic dataset. These examples illustrate practical differences between the Teacher, the Baselines, and the Knowledge Distilled Student. Each includes the user input, sensor context, model responses, and a short analysis.

D.1. Successful Case: Standard Command

Input:

- **User Query:** "I need to take the next exit, please move into the right lane."
- **Sensor Context:** `{"current_speed": 85.7 ...}`

Model Outputs:

Teacher (LLaMA 3.1 8B):

```json
{
  "command": "change_lane_right",
  "parameters": {
    "urgency": "normal"
  }
}
```

Baseline Student (1B):

```json
{
  "command": "change_lane_right",
  "parameters": {}
}
```

KD-Student (1B):

```json
{
  "command": "change_lane_right",
  "parameters": {
    "urgency": "normal"
  }
}
```

Analysis: All models correctly interpret the intent. The Teacher and KD-Student add nuanced understanding by specifying urgency, indicating successful transfer of contextual inference.

D.2. Challenging Case: Colloquial Language and Ambiguity

Input:

- **User Query:** "Whoa, this guy is way too close. Can you, like, back off a bit?"
- **Sensor Context:**
  `{"current_speed": 45.0, "distance_to_vehicle_ahead": 8.5 ...}`

Model Outputs:

Teacher (LLaMA 3.1 8B):

```
{
  "command": "follow_lane",
  "parameters": {
    "target_distance": 15.0
  }
}
```

Baseline Student (1B):

```
{
  "command": "emergency_brake",
  "parameters": {}
}
```

KD-Student (1B):

```
{
  "command": "follow_lane",
  "parameters": {
    "target_distance": 15.0
  }
}
```

Analysis: The Teacher and KD-Student correctly interpret informal speech. The Baseline Student's misclassification as a high-risk situation.

D.3. Failure Case: Necessary Rejection (Safety-Critical)

Input:

- **User Query:** "Let's make a U-turn here, I think I missed my turn."
- **Sensor Context:** `{"current_speed": 110.0}`

Model Outputs:

Teacher (LLaMA 3.1 8B):

```
{
  "command": "reject_command",
  "parameters": {
    "reason": "Cannot perform U-turn on a highway at current speed."
  }
}
```

Baseline Student (1B):

```
{
  "command": "perform_u_turn",
  "parameters": {}
}
```

KD-Student (1B):

```
{
  "command": "reject_command",
  "parameters": {
    "reason": "U-turn is unsafe at high speed and on this road type."
  }
}
```

Analysis: Baseline Student fails a safety-critical judgment. The Teacher and KD-Student correctly reject the action, highlighting distillation's role in transferring safety reasoning.

D.4. Failure Case: Malformed Output (Hallucination)

Input:

- **User Query:** "Ok, find a spot to pull over and park on the side of the road somewhere up ahead."
- **Sensor Context:** `{"current_speed": 55.0 ...}`

Model Outputs:

Teacher (LLaMA 3.1 8B):

```
{
  "command": "stop_park",
  "parameters": {
    "parking_type": "shoulder"
  }
}
```

Baseline Student (1B):

```
Sure, I can do that. I will look for a safe place to pull over.
{
  "command": "stop_car"
```

KD-Student (1B):

```
{
  "command": "stop_park",
  "parameters": {
    "parking_type": "shoulder"
  }
}
```

Analysis: The Baseline Student produces malformed, chatty output that breaks JSON structure. KD-Student and Teacher maintain valid formatting, showing improved structural discipline via distillation.