Thesis for The Degree of Licentiate of Engineering

Towards Domain-Centric Artificial Intelligence

Bridging General Capabilities and Domain-Specific Context

Arsham Gholamzadeh Khoee

CHALMERS

Division of Computing Science
Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2026

Towards Domain-Centric Artificial Intelligence

Bridging General Capabilities and Domain-Specific Context

Arsham Gholamzadeh Khoee

ISSN 1652-876X

Department of Computer Science & Engineering
Division of Computing Science
Chalmers University of Technology
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2026.

Abstract

Foundation models and Large Language Models (LLMs) have strong general capabilities. They can understand language, reason across different tasks, generate code, and solve problems in areas where they were not specifically trained. This broad capability makes them powerful starting points for real-world AI systems. However, for high-stakes domains, such as automotive software engineering, AI systems must do more than provide plausible answers. They must follow domain rules, respect data structures, handle operational constraints, and produce reasoning that experts can check and trust. This creates a gap between general capability and domain-specific reliability.

This thesis argues for Domain-Centric AI: the design of AI systems that are generalizable across domains, adaptable to target domains, and able to reason reliably within specific operational domains. These levels build on one another. Generalization provides the model-level foundation. Adaptation aligns this foundation with a target domain. Domain-specific system design then enforces the rules, workflows, and constraints needed for reliable use.

The thesis explores this progression through four papers. The first paper surveys meta-learning methods for domain generalization and shows how models can become more robust to unseen domains. The second paper extends this perspective to vision-language models by introducing latent domain prompt learning for domain generalization. The third and fourth papers focus on industrial LLM-based agent systems for automotive software release analytics. They demonstrate how general LLM capabilities can be embedded in multi-agent and pipeline designs to support informed decision-making.

Together, the studies show reliable AI in domain-specific settings can be designed by combining a flexible model core with a constraining system around it. The core model must generalize across unseen domains. The surrounding system must enforce domain logic. By bridging general AI capabilities with structured domain-specific context, Domain-Centric AI can improve robustness, reduce manual effort, and support more reliable decision-making in safety-critical workflows.

Keywords: Foundation Models, Domain Generalization, Prompt Learning, LLM-Based Systems, Domain-Specific Reasoning, Intermediate Representations, Automotive Software Engineering

Acknowledgments

First of all, I would like to express my deepest gratitude to my supervisors, Yinan Yu and Robert Feldt, for providing the foundation for my research and for guiding my growth both professionally and personally. Their encouragement, patience, and thoughtful advice have been invaluable throughout this journey. Special thanks also go to my industrial supervisor, Dhasarathy Parthasarathy, for his insightful guidance and for providing the industry expertise that strengthened my work. I also sincerely thank my examiner, Christian Berger, for his constructive feedback, and Nir Piterman for his valuable guidance.

Beyond my direct supervision, I am grateful to Carl and Alex for ensuring a productive and professional environment. I also thank Mary and John for our spontaneous and insightful discussions. My thanks also go to the administrative staff, especially Clara, for her constant kindness and support behind the scenes.

I would also like to thank my colleagues and friends in the Computing Science division: Ali, Erik, Shuai, Zhixing, Niklas, Katya, Henrik, Robert, Amir, Alasdair, Antonina, Adrian, Piero, and the rest of the division.

I am also grateful to the Iranian community at Chalmers: Mohammad, Mohammad Ali (Saba), Mehrdad, Maryam, Firooz, Samira, Arman, Shirin, Farzaneh (Tamay), Ehsan, Elham, Ashkan, Mehdi, Javad, Kiarash, and Behnaz for their friendship and for making my time here so memorable.

I am deeply grateful to my parents, Saeed and Ramesh, and my sister, Armita, for their unconditional love and support. Finally, I am grateful to my constant companion, Kimia, for her infinite patience and for being by my side throughout this journey.

This work was partially supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

Arsham Gholamzadeh Khoee
Gothenburg, May 2026

List of Publications

Appended publications

This thesis is based on the following publications:

[**Paper I**] **Arsham Gholamzadeh Khoee**, Yinan Yu, Robert Feldt, *Domain Generalization through Meta-Learning: A Survey*
Artificial Intelligence Review 57.10 (2024): 285.

[**Paper II**] Zhixing Li, **Arsham Gholamzadeh Khoee**, Yinan Yu, *Latent Domain Prompt Learning for Vision-Language Models*
ICASSP 2026 – IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2026.

[**Paper III**] **Arsham Gholamzadeh Khoee**, Yinan Yu, Robert Feldt, Andris Freimanis, Patrick Andersson Rhodin, Dhasarathy Parthasarathy, *GoNoGo: An Efficient LLM-based Multi-Agent System for Streamlining Automotive Software Release Decision-Making*
IFIP International Conference on Testing Software and Systems, Springer Nature Switzerland, 2024.

[**Paper IV**] **Arsham Gholamzadeh Khoee**, Shuai Wang, Robert Feldt, Dhasarathy Parthasarathy, Yinan Yu, *GateLens: A Reasoning-Enhanced LLM Agent for Automotive Software Release Analytics*
Submitted, under review.

Other publications

The following publications were completed during my licentiate studies, or are currently under submission or revision. They are not appended to this thesis because their content either overlaps with the appended publications or falls outside the scope of the thesis.

[**a**] **Arsham Gholamzadeh Khoee**, Yinan Yu, Robert Feldt, *Domain-Invariant Prompt Learning for Vision-Language Models*
ICLR 2025 Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models.

Contents

Part A

Summary and Overview

1

Introduction

The field of Artificial Intelligence (AI) has changed significantly with the emergence of Foundation Models and Large Language Models (LLMs) [48, 49]. These models, built at a large scale with substantial investments, show impressive general capabilities, including language understanding, reasoning, code generation, and zero-shot problem-solving [2]. In commercial settings, they often function as general-purpose assistants that can respond to a wide range of general requests [44].

However, when faced with challenging real-world tasks, these models often fail to meet the requirements of such settings [12, 21]. General models work well in flexible and ambiguous situations but often struggle in specialized, high-stakes areas where rigid operational constraints and strict safety requirements apply, and where model errors can directly impact human safety, project timelines, and financial outcomes [12]. In industries such as automotive software engineering, data is not just general-purpose text; it includes test results, technical specifications, cross-disciplinary dependencies, and strict safety requirements [20]. Consequently, using general-purpose models without adjustments can lead to unreliable outcomes. They may hallucinate, miss important constraints, or fail to follow the logic needed for critical decisions [17, 21].

To make AI useful in practical environments, research should focus on improving general capabilities while also tailoring those capabilities to specific fields. This requires working across a spectrum of domain integration:

Domain-Generalizability The ability of a model to handle out-of-distribution (OOD) data and domain shifts without retraining or prior exposure to the target environment [50]. Methods like meta-learning for domain generalization aim to enhance robustness by learning features useful

across various domains [18]. Foundation models and LLMs operate primarily at this level: they are trained on wide-ranging datasets and can adequately respond to many prompts. While this makes them highly versatile and easy to deploy out of the box, it also limits their depth in specialized tasks.

Domain-Adaptability The ability to transfer knowledge from a source domain to a specific, known target domain [35]. Unlike generalizability, adaptability requires access to some data or information about the target environment. Adaptation often involves fine-tuning a generalized model to improve performance in a specific domain while preserving its general capabilities [11, 18]. This level provides a practical balance between flexibility and specialization, enabling organizations to leverage existing foundation models without necessarily building entirely new systems from scratch.

Domain-Specificity This is the deepest level of integration. AI systems are intentionally designed around the constraints, logic, and workflows of a specific industry. Here, domain knowledge is built into the system structurally, enabling reasoning with the precision required in high-stakes settings [20]. The key distinction at this level is that the system internalizes the domain's rules and priorities, producing outputs that are reliable and suitable for critical use [11]. Figure 1.1 illustrates how these three levels form a cumulative progression, where each builds on the capabilities established by the previous.

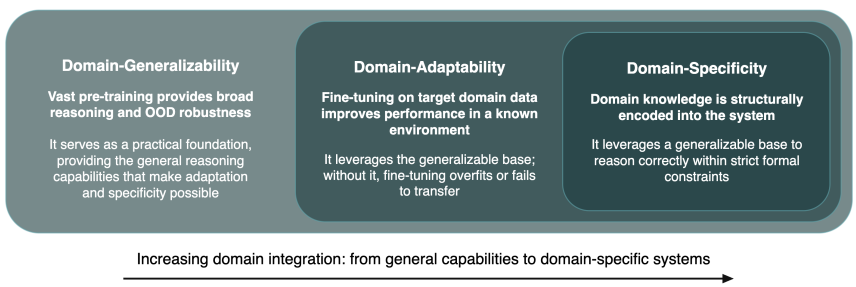


Figure 1.1: The domain integration spectrum underlying Domain-Centric AI, illustrating a progression from broad generalizability to structural domain specificity. Each level builds on the previous, with generalizability serving as a practical foundation for adaptation and structural specialization.

Each level of this spectrum reflects a different commitment to domain knowledge, ranging from broad generalization to targeted specialization. In this thesis, Domain-Centric AI denotes the design of AI systems that are generalizable across domains, easily adaptable, and able to reason reliably within a specific operational domain. Such systems explicitly structure and constrain general model capabilities around the domain's rules, workflows, data structures, and priorities.

These levels should not be seen as separate alternatives. Instead, they often build on one another: broad reasoning, language comprehension, and pattern-recognition capabilities can provide a useful starting point for later adaptation and structural specialization. This thesis is grounded in this view of Domain-Centric AI, arguing that effective AI in high-stakes, domain-specific situations requires core models with robust generalization abilities across unseen domains, while also embedding them in systems where structured, domain-specific logic can be actively enforced.

Achieving this requires AI systems that explicitly structure and constrain model reasoning. For complex engineering tasks, simple prompts are not enough. The model must work within a dedicated system that directs this process. Two main design patterns help with this: Multi-Agent Systems (MAS) [4, 9, 28] and pipeline systems [17], both of which can leverage Intermediate Representations (IRs) to further constrain and ground model reasoning.

MAS distribute tasks among several specialized LLM agents, such as planners, actors, and reflectors. This approach is flexible and works well for open-ended, intent-driven tasks [10, 20]. For instance, in automated mobile GUI testing, agents can set goals, explore interfaces, and act with human-like intention, overcoming the limitations of traditional coverage-based methods [46]. MAS can also support exploratory data analysis by transforming high-level stakeholder questions into detailed executable plans [33]. However, MAS needs careful design. Since agents have some autonomy, their performance can become slower, more unpredictable, and costlier in terms of token usage. In high-stakes situations, a more controlled approach is often necessary [28].

Pipeline systems mainly enforce structural constraints on the model's reasoning, boosting reliability and predictability. This is achieved by introducing fixed processing stages, whether rule-based, learned, or formal, that shape how inputs are transformed into outputs. A common approach is to constrain the model's reasoning through an intermediate representation: a formal, unambiguous structure such as a logical form, abstract syntax tree, or domain-specific query language, that bridges the gap between unstructured

natural language and the task objective [32]. This generally decreases the reliance on extensive few-shot prompting, lowers latency, and steers the model away from unconstrained generation [17]. However, pipeline architectures can be rigid. They expect inputs to fit a specific structure and degrade when faced with inputs that deviate from it; they also struggle with tasks that require dynamic, adaptive reasoning.

The need for Domain-Centric AI is especially evident in multidisciplinary software development in the automotive sector [31, 41]. Building modern vehicles requires turning scattered domain knowledge, like API specifications and Controller Area Network (CAN) signals, into code and test cases ready for deployment. Typically, connecting these different areas is a manual process that requires extensive coordination.

By treating these manual workflows as software engineering challenges, LLMs can be utilized not as open-ended chatbots, but as "skeleton keys" integrated into MAS or pipeline systems to automate complex handoffs [31]. In real-world industrial settings, this domain-centric approach has shown clear value. For instance, in automotive software integration, critical Go/No-Go decisions must be made across subsystem, system, and full-vehicle levels [20]. Traditionally, various stakeholders, including project managers and mechanical engineers, depended on a small, centralized team of analysts to query raw validation data. This reliance created major operational bottlenecks.

By employing pipeline architectures with formal IRs (like Relational Algebra) to analyze complex software release data, the time required for these Go/No-Go decisions has been reduced by over 80% [17]. Importantly, Relational Algebra provides an explicit, interpretable reasoning plan that domain experts can inspect and verify the model's logic before accepting the output. This transparency makes errors diagnosable, builds trust among stakeholders, and shifts human effort from data preparation to strategic decision-making. By constraining LLMs to operate strictly within domain-specific rules, these systems deliver accurate, actionable insights that accelerate development without compromising safety or quality [17].

2

Background

This chapter establishes the conceptual and technical background for Domain-Centric AI. It is organized into four parts. Section 2.1 introduces the classical machine learning view of distributional shifts and generalization, and then examines how the emergence of foundation models reframes these challenges from data-level mismatch toward deductive reliability. Section 2.2 reviews foundation models and the mechanisms used to elicit their capabilities, including prompt learning in Vision-Language Models (VLMs) and structured reasoning in LLMs. Section 2.3 discusses LLM-based system architectures, including multi-agent and pipeline orchestration patterns, and the use of intermediate representations as a structuring mechanism. Finally, Section 2.4 introduces the automotive software release setting, where safety-sensitive validation workflows and Go/No-Go decisions provide the industrial context for the domain-centric systems developed in this thesis.

2.1 Preliminaries

2.1.1 Distributional Shifts and Generalization

Traditional machine learning relies on the empirical risk minimization (ERM) framework. This framework assumes that training and testing data are independent and identically distributed (i.i.d.). In this setting, a source domain $\mathcal{S}$ refers to the data distribution from which the training data are drawn, while a target domain $\mathcal{T}$ refers to the data distribution the model encounters at deployment. If the source and target domains are associated with joint distributions $p(X, Y)$ and $q(X, Y)$, respectively, where X and Y denote the input features and labels, respectively, the i.i.d. assumption implies that these distributions are identical: $p(X, Y) = q(X, Y)$ [18].

In real-world situations, this assumption often fails. Changes in data collection methods, environmental conditions, or underlying system dynamics introduce a discrepancy between the source and target domains, known as a domain shift or distributional shift. When a model encounters data from $q(X, Y)$ that is different from its training distribution, it faces Out-of-Distribution (OOD) data. Traditional deep neural networks may perform well on i.i.d. data but struggle with OOD data, leading to significant performance degradation.

To tackle this issue, researchers primarily focus on two main areas:

- **Domain Adaptation (DA):** This involves methods that try to align the source and target distributions, assuming at least some data from the target domain is available during training [35].
- **Domain Generalization (DG):** This is a more difficult challenge where the model must learn to generalize to completely new target domains without prior exposure to their data [50].

For example, in DA, a model trained on data from one hospital is adapted using data from a new hospital, where the imaging devices may have different characteristics. In DG, by contrast, a model trained on data from several hospitals must generalize to an entirely new hospital, without access to any of its data during training. These two settings, therefore, require different evaluation protocols since they have different goals. In DA, the goal is to close the performance gap between the source and target domains, so success is measured directly by the model's performance on that specific target domain [7]. The source domain serves primarily as a training resource, and any performance improvement there is largely irrelevant if the model fails on the target. In DG, however, no single target domain is known in advance. Therefore, evaluation is conducted across held-out domains that the model has not seen during training. Performance is often averaged as the accuracy across all unseen test domains [18]. Additionally, it is common to report how well the model performs in the worst-case scenario. A model might have good average performance but is not truly generalizable if it fails on one specific domain.

To achieve these goals, different methodological techniques are employed. A common approach for DA is parameter adaptation through fine-tuning [40]. Here, a pre-trained model's backbone remains unchanged, while some layers or adapter modules are updated on limited target data [22]. This allows the model to learn domain-specific patterns without losing its general abilities. Another approach is domain-invariant feature alignment, where meth-

ods like adversarial training [7] or kernel methods [8] aim to reduce the statistical gap between source and target feature distributions.

For DG, where no target domain data is available, domain-invariant representation learning is an important strategy, but it often combines with other methods. An effective and widely studied technique is Meta-Learning, or "learning to learn." By simulating domain shifts during training, usually by splitting source domains into episodic meta-train and meta-test sets, the model learns to extract transferable features instead of memorizing domain-specific patterns [18, 26]. This helps develop a more robust representation capable of handling unseen OOD environments.

2.1.2 From Distributional Shifts to Deductive Bottlenecks

The emergence of Foundation Models and LLMs has greatly changed the landscape of generalization. They are trained on internet-scale datasets containing vast amounts of human knowledge, natural language, syntax, and programming languages [5]. These models operate at a scale that challenges the classical notion of distributional shift as the main explanation for failure in specialized domains.

Due to the volume and diversity of their training data, foundation models have achieved a level of generalization that traditional models could not approach. For a text-based foundation model, it is unlikely that the vocabulary or basic structure of a new industrial domain will be entirely unseen. The model has likely encountered many domain-specific terms during pre-training. Therefore, failures in specialized domains are often not due to standard distributional shift in the input space X.

This does not mean that distributional shift is no longer relevant. Rather, for LLMs, it is often a less central limitation than it was for classical models. This raises a more fundamental question: if LLMs are not primarily constrained by unseen vocabulary or traditional OOD data, why do they still struggle in high-stakes, specialized settings like automotive software engineering?

For these models, the primary bottleneck is no longer data recognition but deductive closure [3, 13, 14]. Deductive closure refers to a system's ability to derive only those conclusions that follow from a given set of premises and rules, without introducing contradictions or unsupported inferences. While LLMs excel at identifying patterns and can generate highly plausible text based on probabilistic next-token prediction, they lack a built-in mechanism for maintaining strict logical consistency across multiple steps of domain-specific reasoning [38].

In specialized industrial settings, data analysis follows strict operational

rules, large relational schemas, and complex interdependencies [54]. When an LLM is asked to analyze this data, it is being asked to reason under strict operational rules and formal constraints that its probabilistic architecture is not designed to enforce. Without changes to the model's structure, its probabilistic nature can cause it to break deductive closure. It may hallucinate relationships, overlook implicit constraints, or fail to maintain the logical consistency required for critical safety decisions.

Therefore, the challenge in building effective AI for high-stakes environments is no longer just teaching models to recognize unseen data. Instead, it is about developing systems that leverage the model's generalized knowledge while constraining its reasoning within a closed logical framework, where each step follows from defined domain rules and no conclusions are drawn outside those rules. This alignment of general capability with domain-specific logic lies at the center of Domain-Centric AI.

2.2 Foundation Models and Eliciting Capabilities

As noted, foundation models possess extensive general knowledge, but this knowledge must be elicited and directed toward the requirements of a given task and modality. Two classes of mechanisms are particularly relevant for bridging general foundation-model capabilities with domain-aware requirements. The first is prompt learning [16, 23, 24], which steers a model toward a specific task by optimizing the input context, while keeping the model's weights fixed [36]. This mechanism is primarily discussed here in relation to VLMs and multimodal generalization. The second is structured reasoning [6, 15], which guides the logical deduction process of LLMs by making intermediate reasoning steps explicit, thereby supporting more reliable multi-step problem solving.

2.2.1 Vision-Language Models and Prompt Learning

VLMs [47], such as CLIP (Contrastive Language–Image Pre-training), have bridged the gap between computer vision and natural language processing. CLIP uses a dual-encoder architecture, consisting of a vision encoder and a text encoder that are trained jointly on a massive dataset of image-text pairs using a contrastive objective: matching pairs are pulled together in a shared embedding space while non-matching pairs are pushed apart. This paradigm enables remarkable zero-shot classification abilities. An image can be classified by finding the text description, such as "A photo of a dog," that has the closest embedding to the image embedding [52].

To steer VLMs toward specific downstream tasks, researchers use prompting. Traditionally, this involved hard prompts—manually crafted, discrete text templates such as "A photo of a [CLASS]" prepended to the text encoder's input [52]. However, designing optimal hard prompts is a rigid and time-consuming process that often yields suboptimal performance. To address this, prompt learning, such as CoOp (Context Optimization) [52], was introduced. Rather than using fixed discrete words, soft prompts consist of continuous, learnable vectors added to the token embedding sequence of the text encoder's input and optimized end-to-end via gradient descent for a specific downstream task, while the VLM's weights remain frozen.

While soft prompts significantly improve model performance on specific downstream tasks, they introduce a critical vulnerability: they are highly susceptible to overfitting the source training distribution. Because the learned vectors are optimized to maximize performance on the source training data, they risk absorbing domain-specific patterns that do not transfer to OOD target domains, even though the underlying VLM encoder remains generally capable.

Therefore, the challenge in the DG setting is not teaching the VLM new visual features but rather finding ways to leverage its existing generalizable knowledge without allowing the learned steering mechanism (the prompt) to compromise its inherent robustness to distribution shift [19].

2.2.2 Large Language Models and Structured Reasoning

For complex multi-step reasoning tasks, a complementary class of prompting techniques has emerged. As LLMs scale and demonstrate abilities to generate code, analyze textual data, and solve multi-step problems, the focus of prompting extends beyond tuning representations toward actively eliciting and guiding logical deduction [30, 43].

To unlock these reasoning capabilities, researchers use advanced prompting strategies designed to make intermediate reasoning explicit. The most well-known of these is Chain-of-Thought (CoT) prompting [43]. Instead of asking the model to provide a final answer directly, CoT encourages the model to produce a series of intermediate reasoning steps. By decomposing a complex problem into smaller sequential steps, CoT improves performance on tasks involving arithmetic, commonsense reasoning, and symbolic reasoning. This approach is often combined with Self-Consistency (SC) [42], where multiple reasoning paths are sampled and the most common answer is selected as the final output.

Despite its success in general benchmarks, CoT has important limitations in rigid, domain-specific logic. It relies on probabilistic text generation rather

than strict logical grounding. Since the reasoning steps are presented in natural language, which is often flexible and ambiguous, the model can generate unreliable reasoning chains. This can happen when the reasoning steps do not accurately represent the model's actual inference, a phenomenon known as unfaithful reasoning, or when the intermediate steps are logically flawed and do not lead to the intended conclusion. In either case, the model may give answers that sound plausible but are not correct or are based on hallucinated assumptions.

When applied to domains requiring strict formal logic, such as mathematics, complex structured data analysis, or compliance, the ambiguity of natural language reasoning becomes a fundamental limitation [13]. There is no guarantee that a probabilistically generated sequence will follow strict domain-specific rules, and natural language does not provide a way to verify this. As a result, the AI research community has proposed various methods to ground the reasoning process itself [3].

Tree of Thoughts [45] breaks down reasoning into clear decision trees, allowing the model to explore various reasoning paths, evaluate steps, and backtrack when necessary. Algorithm of Thoughts [34] guides the model through reasoning pathways using in-context examples, achieving structured exploration with fewer queries. Chain of Code [25] extends reasoning by generating and executing code as verifiable intermediate steps, effectively grounding language reasoning in formal computation. Logic of Thought [30] augments input contexts with propositional logic representations, ensuring information completeness and enabling more rigorous formal reasoning. These approaches represent a shift from purely linguistic reasoning toward hybrid frameworks where the structure and correctness of the reasoning process can be enforced rather than implicitly assumed.

2.3 LLM-Based System Architectures

As LLMs evolved from pure text generators into models capable of reasoning and generating executable code, they have increasingly been embedded into broader software architectures for solving complex, real-world tasks. These architectures can be understood through two complementary dimensions: orchestration patterns, which define how multiple LLM calls are organized and coordinated, and structuring mechanisms, which constrain the form and interpretation of individual LLM outputs. Two well-established orchestration patterns are multi-agent systems (MAS) and pipeline systems. A key structuring mechanism relevant to both is the use of Intermediate Representations (IRs).

2.3.1 Orchestration Patterns

2.3.1.1 Multi-Agent Systems

Instead of relying on a single, monolithic prompt, MAS tackle complex tasks by dividing responsibilities among multiple specialized LLM agents [4]. Each agent has defined roles, tools, and memory. They work together or compete through structured communication to break down and solve tasks that a single model call might not handle reliably. A typical MAS setup for software engineering or data analysis might include a Planner (to break down the user's request), a Coder or Actor (to write and run scripts based on the plan), and a Reflector or Critic (to evaluate the output, check for execution errors, and suggest corrections).

The main advantage of MAS is its flexibility and ability to solve open-ended problems. Since agents have some autonomy and can engage in iterative, self-correcting dialogues, MAS is highly effective for exploratory tasks, like navigating unseen user interfaces or conducting open-ended data exploration. The iterative reflection loop enables the system to recover from mistakes that a single-pass model would likely overlook.

However, the autonomy that makes MAS flexible can also introduce significant drawbacks. Since agents communicate in natural language, they can create cascading hallucinations where an incorrect assumption by the Planner is simply accepted and compounded by the Coder, causing errors to propagate and amplify across the entire system. Additionally, the iterative loop of coding, failing, reflecting, and rewriting is resource-intensive, slow, and consumes large amounts of tokens. In environments where speed, traceability, and strict adherence to domain rules are required, the unpredictable nature of MAS can be a limitation.

2.3.1.2 Pipeline Systems

In contrast to the dynamic, conversational nature of MAS, Pipeline Systems enforce structural constraints on information flow through predefined processing stages. In a pipeline architecture, the flow of information is strictly controlled, usually as a Directed Acyclic Graph (DAG). The LLM is invoked only at specific nodes to perform clearly defined transformations. The output of each stage becomes the structured input for the next, with each stage carrying out one specific transformation, together achieving a result that a single stage could not deliver alone. Notably, while pipelines and MAS are often presented as opposing paradigms, they are not mutually exclusive; a MAS orchestrator may delegate well-scoped subtasks to an internal pipeline,

combining the flexibility of agentic control flow with the reliability of sequential processing.

Pipeline architectures can improve reliability, transparency, and computational efficiency by constraining information flow through predefined stages. However, this control comes at the cost of rigidity: pipelines typically assume that inputs conform to an expected structure and may degrade when faced with inputs that deviate from it.

2.3.2 Intermediate Representations

Both orchestration patterns can benefit from mechanisms that constrain the form of individual LLM outputs. In Domain-Centric AI, one particularly important mechanism is the use of IRs [37].

Instead of prompting the LLM to map directly from a vague natural language query to an executable output, IR-based approaches require the model to first convert the query into a formal, unambiguous representation, such as an Abstract Syntax Tree (AST), a logical form, or Relational Algebra (RA).

The IR serves as a bridge between unstructured natural language and the final executable task. Governed by mathematical or formal syntax, it reduces the ambiguity of natural language. Once the LLM successfully converts the user's intent into this formal representation, the subsequent code generation becomes a significantly more constrained task—either a deterministic compilation when executed by a formal engine, or a substantially narrowed probabilistic translation when delegated to a second LLM call. By grounding the LLM's reasoning in a meaningful IR, both pipeline systems and MAS agents can significantly alleviate the deductive bottleneck. They reduce the need for iterative error correction, lower reliance on extensive few-shot prompting, and limit the space in which hallucinations can occur, making them highly suitable for rigorous, high-stakes domains.

Thus, IRs are advantageous for Domain-Centric AI because they establish formal logical representations that incorporate domain-specific constraints and separate the task into two distinct cognitive loads:

1. **Semantic Parsing / Reasoning:** The LLM translates the natural language query into the formal IR.
2. **Compilation / Translation:** The IR is either executed directly by a deterministic engine or translated into executable code (e.g., Python or SQL).

This separation is central to Domain-Centric AI: it allows general LLM capabilities to be used while keeping the critical reasoning process inspectable,

controllable, and aligned with domain-specific rules.

2.4 The Industrial Setting: Automotive Sector

The need for Domain-Centric AI is especially clear in the automotive industry, where the transition to the software-defined vehicle has created a deeply interconnected challenge: software development has become increasingly complex, and the volume of test logs, validation data, and system signals is so large that managing the development process now depends on managing the data it produces. Addressing the data analytics bottleneck is therefore not secondary, but essential to maintaining control over the development process itself.

Modern vehicles rely on millions of lines of code to operate and coordinate physical hardware. This generates vast amounts of specialized engineering data that are difficult to interpret with general-purpose AI models. Unlike ordinary text, this data is structured, technical, and tightly tied to system behavior, test conditions, and safety requirements. Models designed for open-ended conversation cannot reliably reason about this kind of data or function within the strict workflows used for automotive validation.

This limitation becomes critical in a safety-sensitive environment where tolerance for error is extremely low. In such settings, a hallucination or incorrect inference is not just a technical error; it can undermine system integrity and trust in the validation process. Therefore, AI in automotive development cannot just produce plausible answers. It must operate within the domain's logic, data structures, and safety constraints.

2.4.1 Automotive Software Architecture

Automotive software is complex because it needs to manage many different subsystems within a single vehicle. Modern vehicles contain networks of Electronic Control Units (ECUs), which are embedded computers responsible for various functions such as engine control, braking, seat heating, and door locking. These ECUs communicate over in-vehicle networks, most often the Controller Area Network (CAN) bus, which supports reliable real-time communication in electrically noisy environments [41].

The main challenge is not only that vehicles have many ECUs, but also that these units vary in function, software, timing requirements, and critical safety levels. Automotive software must integrate them into one coherent and predictable system. As vehicles become more connected, automated, and software-driven, this integration becomes much more difficult. Software

now drives much of automotive innovation and accounts for an increasing share of development effort and cost.

2.4.2 The Software Release Lifecycle

Given the physical consequences of software failures in safety-critical vehicles, the release lifecycle is governed by strict validation protocols and progresses through multiple hierarchical stages [17]:

1. **Subsystem Integration:** Individual software components and ECUs are tested in isolation.
2. **System Integration:** Multiple subsystems are combined and tested for interoperability and signal integrity.
3. **Full-Vehicle Integration:** The complete software suite is deployed on physical test rigs or closed-track vehicles to validate real-world performance and safety.

Each stage generates large amounts of tabular test data. As integration moves forward, the number of dependencies between components grows, making the results harder to interpret and manage. This increasing complexity increases the burden on release oversight.

The Release Manager is responsible for this process and acts as the final gatekeeper in the software release lifecycle. They ensure that safety standards, functional requirements, and cross-disciplinary dependencies are met before the software moves to the next stage.

2.4.3 Risk-Sensitive Decision Making: The Go/No-Go Process

At each integration stage, the Release Manager makes the Go/No-Go decision [20]. To do this, they must analyze large volumes of tabular test data, identify failed cases, assess whether problems in one component affect other systems, and evaluate whether a Release Candidate meets defined safety and quality thresholds.

This process is inherently risk-sensitive. Errors or delays in the analysis can affect both vehicle safety and production timelines. In this setting, using autonomous general-purpose AI agents for data analysis creates serious risks, especially when their reasoning is unclear or not fully faithful to the underlying data.

Historically, this data analytics process has been a significant bottleneck. Release Managers, who may not be experts in database querying, often rely

on small, centralized teams of data analysts to translate high-level questions into complex SQL queries or Python scripts. This manual translation is labor-intensive, slow, and prone to human error, delaying critical product timelines.

Although this bottleneck makes the process a natural candidate for AI-based automation, general-purpose LLMs cannot be deployed directly into this workflow without further constraints. Domain experts must remain able to inspect and verify the system's reasoning before its outputs inform release decisions.

Because Go/No-Go decisions affect vehicle safety, any AI assistant deployed in this environment must exhibit:

- **Transparency:** The analytical logic used to generate the report must be fully visible and interpretable by domain experts.
- **Reliability:** The system must not hallucinate constraints, miscount failures, or overlook critical test outcomes.

In this highly constrained and risk-sensitive setting, the need for domain-specific LLM-based systems becomes clear. By aligning general LLM capabilities with strict domain-specific rules, such systems can automate software release analytics without sacrificing the safety, transparency, and rigor required in the automotive industry. Therefore, the automotive setting illustrates Domain-Centric AI as an operational necessity rather than a design preference.

3

Summary of Included Papers

This chapter summarizes the four papers included in this thesis. Together, they trace a progression from general domain robustness to domain-specific AI system design. The first paper surveys meta-learning methods for domain generalization, establishing the problem of robustness under distribution shift. The second paper extends this theme to vision-language models by introducing latent domain prompt learning for domain generalization. The third and fourth papers move from model-level generalization to system-level domain integration in automotive software release analytics: first through a multi-agent LLM system for Go/No-Go decision support, and then through a pipeline architecture that uses Relational Algebra as a formal intermediate representation for more transparent and reliable analysis.

3.1 Paper I: Domain Generalization through Meta-Learning: A Survey

This paper addresses domain generalization through meta-learning, focusing on how models can remain robust when deployed in unseen domains without access to target-domain data during training [18]. Much of the surveyed literature is grounded in computer vision, where domain shifts often appear as changes in visual style, sensor characteristics, lighting, viewpoint, or image acquisition conditions. The paper positions meta-learning as a promising approach to DG because it simulates domain shifts during training and encourages models to acquire transferable knowledge across domains.

The paper provides a systematic survey of meta-learning methods for

domain generalization. Its main contribution is a taxonomy that organizes existing methods according to two axes: a generalizability axis, which describes how the feature extractor handles domain variation, and a discriminability axis, which describes how the classifier separates classes across domains. Together, these axes define four categories of approaches: minimizing inter-domain distances, maximizing intra-domain distances, minimizing intra-class distances, and maximizing inter-class distances. These categories capture whether a method promotes robustness by aligning domains, diversifying the available source data, encouraging samples from the same class to form compact representations, or increasing separation between different classes. The taxonomy is accompanied by an actionable decision graph that helps researchers and practitioners select suitable methods based on data availability, domain diversity, and the nature of the expected domain shift, making the survey both descriptive and practically useful.

Paper I establishes the theoretical foundation for the broader argument about Domain-Centric AI. It focuses on the first level of the domain integration spectrum: domain generalization. Before foundation models can be adapted to specific domains or embedded into domain-constrained systems, it is necessary to understand how models can remain robust when facing unseen domains. By analyzing how meta-learning methods manage domain variation and class separation, the paper clarifies that generalization is not simply a result of having more data. Rather, it depends on how learning algorithms structure representations so that they remain both transferable across domains and discriminative across classes.

Statement of contributions The author of this thesis conducted the primary study and literature review, analyzed and discussed the reviewed methods, designed the taxonomy, and was responsible for the majority of the writing.

Appeared in: *Artificial Intelligence Review* 57.10 (2024): 285.

3.2 Paper II: Latent Domain Prompt Learning for Vision-Language Models

This paper extends the domain generalization perspective from Paper I to foundation models, focusing on VLMs such as CLIP. Although VLMs exhibit strong zero-shot capabilities due to large-scale image-text pretraining, their performance still depends on how their capabilities are elicited for downstream tasks. Soft prompt learning addresses this by replacing manually de-

signed text prompts with learnable continuous prompts. However, learned prompts can overfit to the source domains and degrade under domain shift. Existing prompt-based DG methods often rely on explicit domain labels, which may be unavailable or ambiguous in real-world settings, such as autonomous driving, where changes in lighting, weather, and environment are continuous rather than cleanly separable into predefined domains.

To address this limitation, the paper introduces Latent Domain Prompt Fusion (LDPF), a prompt-learning framework for domain generalization in VLMs without explicit domain labels. Instead of using human-defined domain annotations, LDPF automatically discovers latent domains from image features. It then trains prompts with two complementary components: domain-agnostic tokens, which capture knowledge shared across domains, and domain-specific tokens, which capture latent style or domain characteristics. At inference time, an input image is compared with the discovered latent domain centroids, and the model dynamically fuses domain-specific text features according to these similarities. In this way, an unseen target domain is represented as a combination of latent source domains, enabling more adaptive visual-textual alignment.

Experiments on Office-Home [39], mini-DomainNet [53], PACS [27], and Terra Incognita [1] show that LDPF consistently improves over strong VLM-based baselines that do not use domain labels, including Zero-shot CLIP, CoOp [52], and CoCoOp [51]. The method achieves an average gain of 4.8 percentage points over Zero-shot CLIP. Ablation results further show that the latent domain components contribute to performance, and that replacing latent clusters with manually annotated domain labels can reduce performance, suggesting that human-defined domains may not fully capture image-specific styles. The upper-bound analysis also reveals that domain-specific prompts can be complementary, especially in challenging datasets, while simple similarity-based fusion may not always fully exploit this complementarity.

Paper II strengthens the first level of the domain integration spectrum: domain-generalizability. While Paper I surveys how meta-learning can support robustness under domain shift, Paper II shows how this problem appears in modern foundation models and how prompt learning can be made more domain-aware without relying on explicit domain labels. Thus, the paper bridges classical DG and foundation-model adaptation: it preserves the general capabilities of CLIP while introducing a lightweight mechanism for handling latent domain variation.

Statement of contributions The author of this thesis contributed to the study design, methodological development, analysis of results, and review and revision of the manuscript.

Appeared in: *ICASSP 2026 – IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2026.

3.3 Paper III: GoNoGo: An Efficient LLM-based Multi-Agent System for Streamlining Automotive Software Release Decision-Making

This paper moves from model-level domain generalization to system-level domain integration in an industrial automotive setting. It addresses the Go/No-Go software release process, where release managers and engineers must analyze large volumes of tabular test data to determine whether a software release candidate is ready to progress to the next development or integration stage [20]. Traditionally, this process relies on manual analysis and the translation of high-level stakeholder questions into database queries or Python scripts, making it time-consuming, error-prone, and dependent on specialized analysts. The paper identifies this workflow as a high-value but risk-sensitive target for LLM-based automation, where the system must support domain-specific terminology, vague user queries, interpretability, efficiency, and maintainability.

To address this challenge, the paper introduces GoNoGo, an LLM-based multi-agent system for automotive software release analytics. The system consists of two main agents: a Planner and an Actor. The Planner interprets the user's natural-language query and decomposes it into analysis steps using a knowledge base, few-shot examples, CoT, and SC. To improve risk-sensitivity, the Planner's action space is restricted to predefined atomic actions, namely, slicing and operation. The Actor then translates the planned steps into executable Python code using a coder LLM, together with memory, plugins, and a self-reflection mechanism for detecting and correcting execution errors. This division of responsibility allows the system to combine flexible natural-language interaction with a more controlled analysis workflow.

GoNoGo was evaluated on 50 industrially grounded query tasks derived from real software release analytics needs and organized into four difficulty levels. With 3-shot examples, the system achieved a 100% success rate on tasks up to Level 2 difficulty and 90% success across all tasks, including the most complex Level 4 queries. The system was also deployed and actively

used within the industrial partner's company, where pilot users reported saving approximately two hours per person for each decision-making instance. These results show that GoNoGo can automate routine release analytics, reduce reliance on specialized analysts, and help release engineers focus on higher-level evaluation.

Paper III marks the transition from domain generalization to domain specificity. Unlike Papers I and II, which focus on robustness under domain shift, this paper shows how general LLM capabilities can be embedded into a domain-specific software architecture for a high-stakes industrial workflow. By combining a domain knowledge base, constrained planning actions, few-shot examples, and code-generating agents, GoNoGo demonstrates the practical value of Domain-Centric AI: the LLM is not used as an open-ended chatbot, but as part of a structured system aligned with automotive release logic. At the same time, the paper exposes the limitations of multi-agent architectures in this setting, including reliance on few-shot prompting, iterative correction, and natural-language reasoning. These observations motivate Paper IV, which further constrains the reasoning process through a pipeline architecture based on a formal intermediate representation.

Statement of contributions The author of this thesis led the study, contributed to the design of the GoNoGo architecture, analyzed the results, coordinated the industrial case study with the industrial partners, and was responsible for the majority of the writing.

Appeared in: *IFIP International Conference on Testing Software and Systems*, Springer Nature Switzerland, 2024.

3.4 Paper IV: GateLens: A Reasoning-Enhanced LLM Agent for Automotive Software Release Analytics

This paper builds directly on the lessons learned from Paper III. GoNoGo showed that LLM-based multi-agent systems can automate routine automotive release analytics [20]. However, it relied on CoT prompting, SC, few-shot examples, and iterative agent coordination, which limited scalability, latency, token use, and maintainability. As the system was exposed to a broader user base, query diversity increased, making it more important to handle complex, ambiguous, and imprecise queries without depending on carefully curated examples. GateLens addresses this challenge by moving from flexible multi-

agent orchestration toward a more structured, zero-shot pipeline for reliable tabular analysis in automotive software release decision-making [17].

The key contribution of the paper is the use of Relational Algebra (RA) as a formal IR between natural-language queries and executable code. GateLens first translates a user query into RA expressions using a domain-specific schema and glossary, and then converts these expressions into executable Python code. This two-stage architecture makes the reasoning process more explicit: instead of relying on free-form natural-language reasoning, the system decomposes the query into formal relational operations such as selection, projection, joins, grouping, and aggregation. This formal translation helps resolve ambiguous natural-language queries by mapping vague user intent to precise schema components and RA operations. It also reduces the space for hallucination, since the model must express its reasoning through valid relational operations before code is generated. The architecture further includes in-scope filtering, allowing the system to reject queries that cannot be meaningfully answered using the available database. Together, these design choices improve transparency, reduce dependence on few-shot prompting, and bridge the reasoning-to-code gap that can arise in direct code-generation approaches.

GateLens was evaluated on two benchmarks: a designed benchmark of 50 queries across four difficulty levels and a real-world benchmark of 244 production queries collected from industrial use. With GPT-4o, GateLens achieved 100% F1 on the designed benchmark and 83.51% F1 on the real-world benchmark, outperforming the previous CoT+SC system by approximately 13 percentage points. Additional experiments showed that GateLens handled out-of-scope and imprecise queries more robustly than the baseline, and ablation studies confirmed the importance of the RA layer, with performance dropping substantially when RA was removed. The system also reduced token consumption by approximately 79% compared with the CoT+SC baseline, reflecting the efficiency of its zero-shot, single-pass architecture.

The industrial deployment further demonstrated the practical value of the approach. GateLens supported an extended pilot involving 60–80 users across different stakeholder roles and reduced the time required for Go/No-Go analytics by more than 80%, including the time engineers spent verifying system outputs. The explicit RA plan made the system's reasoning easier to inspect and diagnose, helping stakeholders validate outputs and build trust in the automation. These findings show that in safety-critical industrial environments, performance metrics alone are insufficient: domain experts must also be able to understand and verify the reasoning process before using AI-

generated results in release decisions.

Paper IV represents the most explicit form of domain specificity. It shows how general LLM capabilities can be shaped through a domain-aware pipeline that connects natural-language queries to RA and then to executable Python code. Compared to Paper III, which relied on multi-agent coordination and natural-language reasoning, GateLens demonstrates a more controlled approach to Domain-Centric AI: the LLM remains useful because of its language understanding and code-generation abilities, but its reasoning is constrained by a formal IR that reflects the structure and rules of the domain. Therefore, Paper IV provides a concrete demonstration of the thesis's central argument: high-stakes AI systems require not only general model capability, but also architectural mechanisms that make reasoning structured, transparent, and domain-grounded.

Statement of contributions The author of this thesis led the study, designed the GateLens framework, analyzed the results, coordinated the industrial deployment, and was responsible for the majority of the writing.

Status: Submitted, under review.

4

Discussion, Implications, and Future Work

The central argument of this thesis is that reliable AI in high-stakes domains requires both generalizable model capabilities and domain-aware system design. Foundation models and LLMs provide broad language understanding, reasoning, and code-generation abilities. However, these capabilities must be constrained and guided by domain-specific logic — through system architecture, formal representations, or targeted adaptation — before they can be used reliably in critical settings. The included papers study this connection at different levels: representation learning for domain generalization, prompt learning, MAS, pipeline architectures, and formal IRs.

The first two papers focus on domain generalization. This matters because a model that cannot handle domain shift is difficult to adapt or specialize. Domain generalization provides the basis for later forms of domain integration: if a model learns representations that transfer across unseen domains, it becomes a stronger starting point for domain adaptation and domain-specific system design. Together, Papers I and II show that robustness under domain shift depends not only on model capacity, but also on how learning is structured and how capabilities are elicited. Paper I shows that meta-learning and episodic training can make learned representations more robust by exposing models to simulated domain shifts, enabling transfer to unseen domains and faster adaptation from limited examples [18]. Paper II extends this idea to foundation models: with the VLM backbone kept frozen, prompt learning modifies the input context to elicit and steer existing capabilities toward downstream domains [29].

The final two papers move from model-level robustness to system-level domain specificity. In high-stakes settings, the core problem is not always

that the model has never seen the relevant terminology. Often, the model recognizes the terms but fails to reason correctly under strict operational constraints. This creates a deductive bottleneck: the model may hallucinate relationships, overlook constraints, or produce outputs that cannot be verified against domain rules. Domain-Centric AI addresses this bottleneck by placing the model inside systems that make reasoning structured, transparent, and aligned with domain-specific logic. LLMs should therefore not be treated as standalone decision-makers, but as components within architectures that structure their reasoning, keep it inspectable, and enable domain experts to verify outputs before acting on them.

The industrial results demonstrate that the value of Domain-Centric AI lies not only in automation, but also in making automation usable and efficient in practice. GoNoGo [20] established that LLM-based automation of release analytics is feasible; GateLens [17] then showed that appropriate architectural constraints can make such automation more reliable, efficient, and inspectable. In automotive release analytics, domain experts must be able to understand how an output was produced, diagnose errors, and assess whether a result is trustworthy enough to support a release decision. By exposing intermediate reasoning steps through formal representations, the system shifts human effort from manual data preparation to verification, interpretation, and decision-making. Furthermore, the right architecture can reduce computational cost: in GateLens, the structured pipeline reduced token consumption by approximately 79% compared to the CoT-based baseline and reduced Go/No-Go analysis time by more than 80%, improving both the computational and operational cost of domain-specific AI [17].

The academic implication is that reliable LLM reasoning is not guaranteed by model scaling or in-context examples alone; architectural constraint is a complementary and often more practical strategy. This thesis shows that schemas, constraints, IRs, validators, and execution environments can contribute to the intelligence of the system by encoding domain knowledge operationally. Reasoning reliability is therefore improved not only through model training, but also through system design. This has implications for how the domain integration spectrum should be understood: generalizability provides the base capability, while adaptability and specificity are achieved as much through system architecture as through model adaptation or training.

Future work can extend Domain-Centric AI in several directions. One direction is modeling domain-specific user intent. In many applications, users express goals through vague, incomplete, or domain-specific language. AI systems must infer the intended task, preserve precise user constraints when

they are provided, and still operate within domain rules. A second direction is tracking and predicting states in domain-specific applications. To achieve reliable behavior in interactive domains, the system must maintain a clear link between the current state, previous actions, and possible future outcomes. In such environments, it must keep track of visited states, attempted actions, failures, and contextual cues over long task sequences. A third direction is domain-adaptive code generation, where generated code or executable artifacts must respect domain APIs, data schemas, safety constraints, and downstream objectives. Android GUI testing is one concrete example that combines these challenges: a testing agent must infer user intent, explore unseen screens, remember prior interactions, and generate meaningful test actions instead of only maximizing surface-level coverage.

Addressing these challenges will require both system design and model adaptation. On the system side, future work will likely require advanced memory mechanisms, multi-agent coordination, formal intermediate representations, validators, and execution feedback. On the model side, reinforcement learning can be used during fine-tuning to align foundation models with domain-specific requirements, for example by improving retrieval behavior, reasoning procedures, or code generation under domain constraints. Across all these directions, the core challenge remains the same: transforming general-purpose AI into reliable systems whose reasoning is grounded in the structure and constraints of the domain.

Bibliography

[1] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.

[2] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[3] C. Cai, X. Zhao, H. Liu, Z. Jiang, T. Zhang, Z. Wu, J.-N. Hwang, and L. Li. The role of deductive and inductive reasoning in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16780–16790, 2025.

[4] S. Chen, Y. Liu, W. Han, W. Zhang, and T. Liu. A survey on llm-based multi-agent system: Recent advances and new frontiers in application. *arXiv preprint arXiv:2412.17481*, 2024.

[5] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[6] M. A. Ferrag, N. Tihanyi, and M. Debbah. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*, 2025.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[8] B. Gong, K. Grauman, and F. Sha. Learning kernels for unsupervised domain adaptation with applications to visual object recognition. *International Journal of Computer Vision*, 109(1):3–27, 2014.

[9] S. Han, Q. Zhang, W. Jin, and Z. Xu. Llm multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578*, 2024.

[10] J. He, C. Treude, and D. Lo. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025.

[11] C. Jeong. Fine-tuning and utilization methods of domain-specific llms. *arXiv preprint arXiv:2401.02981*, 2024.

[12] H. Jin, L. Huang, H. Cai, J. Yan, B. Li, and H. Chen. From llms to llm-based agents for software engineering: A survey of current, challenges and future. *arXiv preprint arXiv:2408.02479*, 2024.

[13] S. Kambhampati. Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1):15–18, 2024.

[14] S. Kambhampati, K. Valmeekam, L. Guan, M. Verma, K. Stechly, S. Bhambri, L. P. Saldyt, and A. B. Murthy. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024.

[15] Z. Ke, F. Jiao, Y. Ming, X.-P. Nguyen, A. Xu, D. X. Long, M. Li, C. Qin, P. Wang, S. Savarese, et al. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*, 2025.

[16] M. U. Khattak, H. Rasheed, M. Maaz, S. Khan, and F. S. Khan. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19113–19122, 2023.

[17] A. G. Khoee, S. Wang, R. Feldt, D. Parthasarathy, and Y. Yu. Gatelens: A reasoning-enhanced llm agent for automotive software release analytics. *arXiv preprint arXiv:2503.21735*, 2025.

[18] A. G. Khoee, Y. Yu, and R. Feldt. Domain generalization through meta-learning: a survey. *Artificial Intelligence Review*, 57(10):285, 2024.

[19] A. G. Khoee, Y. Yu, and R. Feldt. Domain-invariant prompt learning for vision-language models. *arXiv preprint arXiv:2603.28555*, 2026.

[20] A. G. Khoee, Y. Yu, R. Feldt, A. Freimanis, P. A. Rhodin, and D. Parthasarathy. Gonogo: An efficient llm-based multi-agent system for streamlining automotive software release decision-making. In *IFIP International Conference on Testing Software and Systems*, pages 30–45. Springer, 2024.

[21] P. Kumar. Large language models (llms): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260, 2024.

[22] Z. Lai, H. Bai, H. Zhang, X. Du, J. Shan, Y. Yang, C.-N. Chuah, and M. Cao. Empowering unsupervised domain adaptation with large-scale pre-trained vision-language models. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pages 2691–2701, 2024.

[23] Y. Lei, J. Li, Z. Li, Y. Cao, and H. Shan. Prompt learning in computer vision: a survey. *Frontiers of Information Technology & Electronic Engineering*, 25(1):42–63, 2024.

[24] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059, 2021.

[25] C. Li, J. Liang, A. Zeng, X. Chen, K. Hausman, D. Sadigh, S. Levine, L. Fei-Fei, F. Xia, and B. Ichter. Chain of code: Reasoning with a language model-augmented code emulator. *arXiv preprint arXiv:2312.04474*, 2023.

[26] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[27] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[28] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

[29] Z. Li, A. G. Khoee, and Y. Yu. Latent domain prompt learning for vision-language models. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12787–12791. IEEE, 2026.

[30] T. Liu, W. Xu, W. Huang, Y. Zeng, J. Wang, X. Wang, H. Yang, and J. Li. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10168–10185, 2025.

[31] D. Parthasarathy, Y. Yu, and E. T. Barr. Polymer: Development workflows as software. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pages 535–539, 2025.

[32] M. Pourreza and D. Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in neural information processing systems*, 36:36339–36348, 2023.

[33] B. Qiao, L. Li, X. Zhang, S. He, Y. Kang, C. Zhang, F. Yang, H. Dong, J. Zhang, L. Wang, et al. Taskweaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541*, 2023.

[34] B. Sel, A. Al-Tawaha, V. Khattar, R. Jia, and M. Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *arXiv preprint arXiv:2308.10379*, 2023.

[35] P. Singhal, R. Walambe, S. Ramanna, and K. Kotecha. Domain adaptation: challenges, methods, datasets, and applications. *IEEE access*, 11:6973–7020, 2023.

[36] Y. Su, X. Wang, Y. Qin, C.-M. Chan, Y. Lin, H. Wang, K. Wen, Z. Liu, P. Li, J. Li, et al. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3949–3969, 2022.

[37] C.-e. A. Tai, P. Nie, L. Golab, and A. Wong. Nl in the middle: Code translation with llms and intermediate representations. In *2025 IEEE International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, pages 295–300. IEEE, 2025.

[38] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in neural information processing systems*, 36:75993–76005, 2023.

[39] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017.

[40] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[41] S. Wang, Y. Yu, R. Feldt, and D. Parthasarathy. Automating a complete software test process using llms: An automotive case study. *arXiv preprint arXiv:2502.04008*, 2025.

[42] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[43] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[44] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, and X. Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–32, 2024.

[45] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[46] J. Yoon, R. Feldt, and S. Yoo. Intent-driven mobile gui testing with autonomous large language model agents. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 129–139. IEEE, 2024.

[47] J. Zhang, J. Huang, S. Jin, and S. Lu. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644, 2024.

[48] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124, 2023.

[49] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *International Journal of Machine Learning and Cybernetics*, 16(12):9851–9915, 2025.

[50] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. Domain generalization: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4396–4415, 2022.

[51] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825, 2022.

[52] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *International journal of computer vision*, 130(9):2337–2348, 2022.

[53] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021.

[54] Y. Zhu, Y. Zhong, J. Zhang, Z. Zhang, S. Qiao, Y. Luo, L. Du, D. Zheng, N. Zhang, and H. Chen. Why do open-source llms struggle with data analysis? a systematic empirical study. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35239–35247, 2026.

Part B

Appended Papers

1 Domain Generalization through Meta-Learning: A Survey

Arsham Gholamzadeh Khoee, Yinan Yu, Robert Feldt

Artificial Intelligence Review 57.10 (2024): 285

Abstract

Deep neural networks (DNNs) have revolutionized artificial intelligence but often lack performance when faced with out-of-distribution (OOD) data, a common scenario due to the inevitable domain shifts in real-world applications. This limitation stems from the common assumption that training and testing data share the same distribution—an assumption frequently violated in practice. Despite their effectiveness with large amounts of data and computational power, DNNs struggle with distributional shifts and limited labeled data, leading to overfitting and poor generalization across various tasks and domains. Meta-learning presents a promising approach by employing algorithms that acquire transferable knowledge across various tasks for fast adaptation, eliminating the need to learn each task from scratch. This survey paper delves into the realm of meta-learning with a focus on its contribution to domain generalization. We first clarify the concept of meta-learning for domain generalization and introduce a novel taxonomy based on the feature extraction strategy and the classifier learning methodology, offering a granular view of methodologies. Additionally, we present a decision graph to assist readers in navigating the taxonomy based on data availability and domain shifts, enabling them to select and develop a proper model tailored to their specific problem requirements. Through an exhaustive review of existing methods and underlying theories, we map out the fundamentals of the field. Our survey provides practical insights and an informed discussion on promising research directions.

I.1 Introduction

Traditional machine learning approaches assume training and testing data are independent and identically distributed (i.i.d.). However, this assumption often fails in practice due to variations in data acquisition conditions, such as changes in sensor types, lighting conditions, or environmental factors, leading to substantial performance degradation when models encounter new domains [41]; thus, they cannot cope with the out-of-distribution (OOD) challenge [7]. A domain can be defined as a specific distribution of data characterized by its own set of features, statistical properties, and underlying concepts [85]. Domain generalization (DG) and domain adaptation (DA) techniques evolved to deal with domain shifts to address out-of-distribution problems [28, 87]. When we have two distinct domains, namely the source and target domains, we can apply domain adaptation techniques, which focus on adapting a model trained on a source domain to perform well on a target domain. Meanwhile, domain generalization aims to generalize over any domain, allowing models trained on one set of domains to perform well on unseen domains. It is important to note that DA techniques assume access to target domain data [56, 65], whereas DG techniques assume no access to the target domain data, which makes it a much harder problem to solve. Compared to DA, DG is more applicable in practice. First, machine learning models are trained on a specific large dataset to perform well on that exact task. However, in real-world scenarios, we often face new tasks with limited labeled data available. Essentially, collecting labeled data and retraining the model for such "unseen" domains can be extremely costly and time-consuming. Also, it is often impossible to enumerate all the "unseen" domains in advance. Accordingly, enhancing the generalization capability of machine learning models is crucial.

Meta-learning has emerged to enable a model to quickly adapt and learn from a small amount of data or even generalize to unseen tasks [38]. Meta-learning algorithms leverage prior knowledge, patterns, and experiences acquired from similar or related tasks to make learning transferable [69]. Recently, some meta-learning algorithms have been used for domain generalization, allowing models to generalize across various domains with limited or no access to all domains during training. These algorithms learn a more generalized representation by identifying patterns that generalize across unseen domains. For instance, consider the PACS dataset [27], which comprises images from four distinct domains: photo, art, cartoon, and sketch. The goal is to train a model on a subset of these domains, such as photos, art, and cartoons, in such a way that it can also accurately classify sketches despite

never having “seen" sketches during training. This challenges the model to extract domain-invariant patterns essential for classifying images across all four domains. This “learning-to-generalize" paradigm aligns with the human learning process, where knowledge acquired from multiple situations enables individuals to adapt quickly to new situations [25].

Recently, there has been a growing interest in developing meta-learning methods for domain generalization. These methods aim to address the limitations of conventional machine learning approaches by incorporating meta-level learning procedures that enhance a model's ability to generalize across domains. More specifically, these models achieve cross-domain generalization by learning from a variety of tasks and domains during training, which prepares them to handle distributional shifts and enables more effective generalization. It is important to note that in domain generalization via meta-learning, the assumption is that there are zero target domain examples during training, contrasting with typical meta-learning that often focuses on few-shot learning for task adaptation.

Wang et al. [92] and Zhou et al. [50] discussed recent advances in domain generalization, covering the formal definition of the problem, related fields, and theoretical foundations, and they highlight commonly used datasets and applications. Additionally, Liu et al. [53] recently explored the OOD generalization problem, which addresses scenarios where the test data distribution differs from the training data. Their review encompasses methodologies categorized into unsupervised representation learning, supervised model learning, and optimization. Viltala et al. [89] published one of the first surveys on meta-learning to formalize self-adaptive learner models. Huisman et al. [38] later provided a comprehensive survey on meta-learning to explore the common challenge of how to leverage meta-knowledge to improve the performance of learning algorithms. Hospedales et al. [37] also discussed various perspectives on meta-learning by providing a novel taxonomy and investigating the utility of the most commonly used meta-learning algorithms. Vettoruzzo et al. [88] have recently examined meta-learning algorithms, discussing their benefits and challenges, and categorizing methods into metric-, model-, and optimization-based techniques, with a focus on achieving consistent evaluation metrics and computational efficiency. In contrast to previous works, to our best knowledge, this paper is the first survey collecting literature and highlighting the overall potential of meta-learning for domain generalization. In this paper, we offer a fresh perspective and a new taxonomy for classifying meta-learning approaches designed for generalization to unseen domains.

As a more detailed explanation, this survey paper *provides a systematic*

overview of existing meta-learning methods tailored for domain generalization. We introduce the concept of domain generalization and clarify its significance in real-world machine learning applications. Subsequently, we delve into the theoretical foundations of meta-learning and explain its applicability to the domain generalization setting. The paper reviews various frameworks and methodologies employed in existing meta-learning approaches for domain generalization, highlighting their strengths and limitations. Additionally, we present a taxonomy to help with the effective categorization of these algorithms. Furthermore, we survey the datasets and evaluation protocols prevalent in this research area and point out the principal challenges and open questions for future study. Our goal is to present an extensive reference for researchers and practitioners interested in understanding and advancing the field of meta-learning for domain generalization.

The rest of this survey is structured as follows. In Section I.2, we provide background information on the topic and introduce the fundamental concepts of meta-learning and its applicability to the domain generalization setting. Section I.3 presents a taxonomy that can be used to categorize techniques that will be discussed in this survey. We then proceed to explore the different frameworks of meta-learning methods for domain generalization in Section I.4. Section I.5 discusses the widely adopted datasets and evaluation protocols employed in the field. In Section I.6, we cover the crucial applications of meta-learning for domain generalization and its significance in practical machine learning scenarios. Section I.7 discusses the significance of the findings, key challenges, and promising research directions. Finally, Section I.8 summarizes the main points of the survey and their implications.

I.2 Background

While deep learning models have achieved remarkable results in specific tasks, developing models that can generalize across tasks and domains remains a challenge. Initially, we will provide a concise overview of related research areas, highlighting their unique characteristics, with a comparison of the discussed learning paradigms available in Table I.1. Subsequently, we will delve into more details of meta-learning and domain generalization, along with providing a formalization of meta-learning for domain generalization, as it represents a research area specifically aimed at addressing the challenges of generalization across various tasks and domains.

I.2.1 Different Learning Paradigms

In real-world applications, data availability is often limited and conditions change over time. A practical machine learning model must adapt to these changes without retraining from scratch, as traditional retraining is resource-intensive and inefficient. To address this challenge, several learning paradigms have been introduced. This section briefly introduces the key learning paradigms to build intuition for designing efficient machine learning models.

I.2.1.1 Incremental Learning

Incremental learning involves gradually training a model on new data over time. The model is updated incrementally as new data arrives, often in batches. However, previous data is often not retained, so incremental learning can suffer from catastrophic forgetting of earlier concepts. It is also necessary to carefully design training data to avoid bias and ensure that the AI system can generalize from the new data [7, 33, 97]. The other important challenge is to deal with non-stationarity in the data, which can prevent the AI system from converging on a solution. Incremental learning can be done online or offline [34].

I.2.1.2 Online Learning

Online learning involves training a model on data that becomes available in a sequential order, sample-by-sample or mini-batch by mini-batch [36]. The model is updated each time a new sample/mini-batch arrives. Online learning algorithms aim to perform updates efficiently to accommodate new data but do not explicitly retain old knowledge.

I.2.1.3 Continual Learning

Continual learning refers to the ability of a model to learn sequentially over time from a stream of data. The goal is to learn new tasks without forgetting previous knowledge, handling different distributions in the data over time [14]. It involves retaining previously learned knowledge and effectively integrating new knowledge.

I.2.1.4 Transfer Learning

Transfer learning refers to the process of reusing a pre-trained model as the basis for a new model, where a model trained on one task is repurposed for a second related task to allow rapid progress when modeling the second

task [64]. Essentially, transfer learning involves using the knowledge learned from one task to improve the performance of another related task.

I.2.1.5 Multi-task Learning

Multi-task learning is a machine learning approach in which we try to learn multiple tasks simultaneously, optimizing multiple loss functions at once. The goal of multi-task learning is to improve learning efficiency and prediction accuracy for the task-specific models, when compared to training the models separately [99]. It leverages shared representations, which can help other tasks be learned better.

I.2.1.6 Meta-Learning

Meta-learning refers to the process of learning how to learn [25]. It involves training a model to learn new tasks quickly with minimal data by leveraging knowledge learned from previous tasks. The goal of meta-learning is to learn a good initialization of the model's parameters that can be adapted quickly to new tasks with only a few samples (also referred to as few-shot learning) [82]. In other words, it transfers knowledge from many source tasks, so it can learn a new task more quickly, more proficiently, and more stably [37].

I.2.1.7 Domain Adaptation

DA focuses on adapting a model from one source domain to perform well on a related target domain [3, 6, 13, 40, 55]. It assumes some similarity or overlap between the two domains. It leverages labeled data from the source domain and unlabeled/limited labeled data from the target domain. Also, it is closely related to transfer learning, which also aims to transfer knowledge from the source domain to the target domain.

I.2.1.8 Domain Generalization

DG goes a step further in comparison to DA. It involves training a model on multiple source domains with the goal of making it perform well on any unseen or new target domains [27, 60, 78]. So, it does not assume access to target domain data during training. DG addresses the challenge of adapting to diverse and unknown domains by learning representations that capture the underlying invariant factors across various domains.

Table I.1: Comparison of Different Learning Paradigms Highlighting Similarities and Dissimilarities Among Meta-Learning, Domain Generalization, and Related Learning Approaches.

Learning Paradigm	$p(X)$ Consistency	Y Consistency	Test Access	$\hat{p}(Y\|X;\theta^1)$
Incremental/Online Learning [36]	✓	✓	✓	$\min_\theta \mathbb{E}_{p(\mathcal{D})}[\ell(\mathcal{D};\theta)]$
Continual Learning [15]	✗	✓	✓	$\min_\theta \mathbb{E}_{p(\mathcal{T})}\mathbb{E}_{p(\mathcal{D}_\tau)}[\ell(\mathcal{D}_\tau;\theta)]$
Transfer Learning [86]	✗	✗	✓	$\min_\theta \mathbb{E}_{p(\mathcal{D}^\mathcal{T})}[\ell(\mathcal{D}^\mathcal{T};\theta)]$[2]
Multi-task Learning [77]	✓	✓	✓	$\min_{\theta^{sh},\theta^\tau} \mathbb{E}_{p(\mathcal{T})}\mathbb{E}_{p(\mathcal{D}_\tau)}[\ell(\mathcal{D}_\tau;\theta^{sh},\theta^\tau)]$[3]
Meta-Learning [94]	✓	✗	✓	$\min_\theta \mathbb{E}_{p(\mathcal{T})}\mathbb{E}_{p(\mathcal{D}_\tau)}[\ell(\mathcal{D}^{tr}_\tau,\mathcal{D}^{te}_\tau;\theta)]$, $\mathcal{D}_\tau = \mathcal{D}^{tr}_\tau \cup \mathcal{D}^{te}_\tau$[4]
Domain Adaptation [23]	✗	✓	✓	$\min_\theta \mathbb{E}_{p(\mathcal{D}^\mathcal{T})}[\ell(\mathcal{D}^\mathcal{T};\theta)]$
Homogeneous DG [50]	✗	✓	✗	$\min_\theta \mathbb{E}_{p(\mathcal{S})}\mathbb{E}_{p(\mathcal{D}_s)}[\ell(\mathcal{D}_s;\theta)]$[5]
Heterogeneous DG [81, 95]	✗	✗	✗	$\min_\theta \mathbb{E}_{p(\mathcal{S})}\mathbb{E}_{p(\mathcal{D}_s)}[\ell(\mathcal{D}_s;\theta)]$

[1] In this table, we use θ to denote the set of all trainable parameters in the model.
[2] $\mathcal{D}^\mathcal{T}$ represents the target domain.
[3] θ^{sh} and θ^τ denote the shared and the task-specific parameters, respectively.
[4] $\mathcal{D}^{tr}_\tau$ and $\mathcal{D}^{te}_\tau$ indicate the meta-train and meta-test splits of task τ's data, respectively.
[5] $\mathcal{D}_s$ and $\mathcal{S}$ represent a dataset from source domain s and a set of source domains, respectively.

I.2.2 Meta-learning for Domain Generalization: Promises and Challenges

Meta-learning is considered a promising approach for DG, as it trains models to build transferable knowledge that can be efficiently applied to different tasks and domains. This aligns with the goal of DG to generalize to unseen domains. By training on a variety of learning tasks, meta-learning enables models to acquire knowledge that is transferable across different domains. In other words, meta-learning enables models to learn how to learn, which significantly enhances their ability to adapt to new and unseen domains by leveraging previously acquired knowledge. This adaptability is crucial for DG, where the goal is to generalize beyond the training domains (see Table I.2 for a comparison between traditional machine learning and meta-learning). In meta-learning, models are trained episodically where each task or episode simulates a domain shift. This procedure effectively equips the model with the ability to handle domain shifts by encouraging the development of features that are robust to the variations between training and test domains. By repeatedly encountering a variety of tasks, the model learns to extract features that are crucial for performance across different domains, rather than overfitting to the idiosyncrasies of any single domain. These robust features are more likely to be relevant in unseen domains, thus improving the overall generalization of the model. Additionally, the intrinsic sample efficiency of meta-learning can lead to more effective utilization of limited data. Meta-learning is designed to achieve high performance with fewer examples by leveraging the knowledge gained from previous tasks. This sample efficiency is particularly beneficial in scenarios where data from new domains is scarce

or expensive to obtain.

One of the main challenges of using meta-learning for DG is ensuring a sufficient diversity of learning tasks during the meta-training phase. Without enough diversity, the model may lack the capacity to generalize effectively to completely new domains. Additionally, the degree of the distributional shift between the training domains and the test domain can be substantial. If the unseen test domain differs significantly from the training domains, the model may encounter difficulties in generalizing. Furthermore, while meta-learning is designed to be sample-efficient, in reality, it often requires a large number of tasks to achieve high performance, which might not always be feasible or accessible. It is also important to note that effective domain generalization often requires understanding causal relationships within the data, which meta-learning algorithms might not capture without proper inductive biases or domain knowledge integration.

I.2.3 Formalization of Meta-Learning for Domain Generalization

In meta-learning, the goal is to solve new unseen tasks by leveraging the knowledge gained from solving N previous tasks. Each task $\mathcal{T}_i$ corresponds to a dataset $\mathcal{D}_i = \{(x_j^i, y_j^i) | x_j \in \mathcal{X}, y_j \in \mathcal{Y}\}_{j=1}^n$, where $\mathcal{D}_i$ is divided into a support set $\mathcal{D}_i^{\mathcal{S}}$ and a query set $\mathcal{D}_i^{\mathcal{Q}}$. It is important to note that in meta-learning, all tasks are sampled from a unique task distribution $p(\mathcal{T})$.

In domain generalization, we are given data from M different source domains and aim to perform well on a new unseen domain, known as the target domain. Specifically, each domain d_i is defined as follows:

$$d_i \triangleq \{p_i(x), p_i(y|x)\}, \tag{I.1}$$

where $p_i(x)$ refers to the distribution of the input data for domain d_i, and this distribution may vary across different domains. The term $p_i(y|x)$ represents the conditional probability distribution of the target y given the input data x. In a homogeneous DG setting, $p_i(y|x)$ is assumed to be consistent across all domains, meaning the label space is the same. However, in a heterogeneous DG setting, $p_i(y|x)$ can vary, as there can be disjoint label spaces across different domains.

For instance, consider the PACS [27] dataset for homogeneous DG; this dataset comprises images from 4 distinct domains: photo, art painting, cartoon, and sketch. Each domain shares the same set of 7 classes for labeling. Tasks are drawn from these varying domains, each with its unique statistical characteristics. Consequently, there is a significant domain shift between

Table I.2: This table outlines the key differences between traditional machine learning and meta-learning across various aspects such as tasks, training datasets, purpose, loss functions, weights, and the role of these weights. The comparison highlights how meta-learning enhances adaptability and generalization to new and unseen domains by leveraging knowledge from a variety of tasks, in contrast to traditional machine learning which typically focuses on optimizing performance for a specific task.

	Machine Learning	**Meta-Learning**
Task	Image recognition, semantic analysis of text	
Training Dataset	Typically a large dataset annotated for the current task	Variety of tasks, often organized into training episodes. Each episode contains a small amount of data from a particular task.
Purpose	Learn to perform a specific task from a dataset	Learn to perform specific tasks and also adapt to new tasks using only a few examples or new datasets (domains) with unseen distributions.
Loss Function	A mathematical function that measures the discrepancy between the predicted output and the ground truth output	Measures not only the prediction error on the current task (same as machine learning loss) but also optimizes adaptability to new tasks.
Weights	Learned parameters that are adjusted during the training process to minimize the loss on that task	Base-level weights which are task-specific, and meta-level weights or hyperparameters that guide the learning process across tasks and datasets (different domains).
What are these weights good for	Transforming the raw features into a more representative space (for example, a more linearly separable space for classification tasks)	Should be able to ignore features only relevant to domain differences (robust against domain shift), focusing on the features relevant to the task.

tasks, necessitating a model capable of generalizing across these different domains to effectively handle tasks from varied distributions. In contrast, the Visual Decathlon (VD) [70] dataset includes 10 different domains, each en-

compassing a variety of categories, and is thus suitable for designing tasks for heterogeneous DG. While the aforementioned datasets focus on multi-source DG, where the model is trained across several source domains to provide representations that generalize to novel unseen domains, there are scenarios where only a single source domain is available. In these cases, the model is expected to generalize to target domains without prior exposure to multiple source domains. This challenge is known as single-domain generalization. Additionally, some problems require the model to handle an open-set of target domains as opposed to a limited or closed-set of target domains. This requirement significantly increases the difficulty of domain generalization problems.

As it can be conceived, a domain can be viewed as a special case of a task from the perspective of meta-learning. Therefore, the meta-learning paradigm can also be applied to domain generalization problems, where each domain is treated as if it were a distinct task, defined by a specific source $\mathcal{S}_i$ and associated with a dataset $\mathcal{D}_i$. It's also worth mentioning that we can leverage zero-shot learning capabilities for new unseen domains when integrating domain generalization techniques with meta-learning.

I.3 Taxonomy

To advance DG through meta-learning, it is important to get an overview of the existing approaches and their similarities and differences. Here, we introduce a taxonomy and its key dimensions (axes) with which different meta-learning approaches can be analyzed. The taxonomy provides two axes, the discriminability axis, and the generalizability axis, representing how a model learns to generalize to unseen domains. This taxonomy aims to highlight the interaction and knowledge transferability between data points from different domains and classes. The end goal is to meta-learn domain-agnostic features and discriminative classifiers that perform well on unseen domains.

Compared to other meta-learning techniques, a unique characteristic of DG is its design principle that prevents the model from adapting to specific test domains. This is because, by definition, DG focuses on generalizing to completely unseen test domains, unlike domain adaptation, where the model has some form of access or exposure to information from the test domain during training. This distinct feature requires the meta-learning approach to *generalize* instead of *adapt*. Consequently, it is crucial to balance the trade-off between domain-invariant and domain-specific features [8]. Specifically, in classification tasks, the domain-specific features that are learned must be

Figure I.1: An illustrative diagram of the meta-learning taxonomy for domain generalization. The quadrant chart highlights two principal axes: the first axis (the generalizability axis) represents the strategy of the feature extractor, contrasting the Minimization of Inter-Domain Distances with the Maximization of Intra-Domain Distances; the second axis (the discriminability aixs) depicts the classifier training process, distinguishing between the Minimization of Intra-Class Distances and Maximization of Inter-Class Distances. This diagram visually organizes domain generalization approaches, illustrating their distinct mechanisms for promoting generalization across unseen domains.

carefully curated to enhance class separability even for unseen domains. The taxonomy presented in this survey is designed to address these aspects. We divide the categorization into two principal axes: the first concerning the treatment of domain features by the feature extractor (generalizability) and the second relating to the training methodology of the classifier within the model (discriminability).

I.3.1 Generalizability Axis

The feature extractor is a pivotal component in domain generalization, as it determines how the model perceives and processes data from diverse domains to learn generalized representations applicable across various settings. Approaches along this axis can be divided into two categories: (1) *Minimization of Inter-Domain Distances* and (2) *Maximization of Intra-Domain Distances*.

I.3.1.1 Minimization of Inter-Domain Distances

Some approaches aim to extract domain-invariant features by minimizing the distance between feature representations across different domains. The primary objective is to identify and harness the characteristics that are consistent across domains. In other words, these methods mainly focus on directly inducing invariant representations without diversifying inputs, concentrating on the consistent and discriminative aspects of the data. By doing so, they reduce the model's sensitivity to the idiosyncrasies of any given domain, thereby enhancing the robustness of the model against the shifts that occur in novel environments.

I.3.1.2 Maximization of Intra-Domain Distances

Alternatively, certain approaches seek to construct a feature extractor capable of handling a more extensive variety of data by maximizing inter-domain distances. These methods utilize strategies like data augmentation, noise injection, and domain randomization to expand the training data spectrum and encourage the model to learn features that are broadly applicable rather than domain-specific. By exposing the model to a more diverse set of modified inputs, these methods strive to replicate the variability that the model will encounter in real-world scenarios, thus promoting the learning of features with better generalizability. Through the explicit diversification of inputs from a domain, these approaches develop feature extractors prepared to handle greater variance in the data.

I.3.2 Discriminability Axis

The performance of DG models is highly dependent on the effectiveness of the classifier's training methodology. The strategies in this context are categorized into two groups based on how the classifier manages distances: (1) *Minimization of Intra-Class Distances* and (2) *Maximization of Inter-Class Distances*.

I.3.2.1 Minimization of Intra-Class Distances

Most approaches simply rely on minimizing the distances between instances within the same class. The intention is to encourage the model to group similar examples together. This method ensures that the model's predictions are consistent for similar instances.

I.3.2.2 Maximization of Inter-Class Distances

Some methodologies also explicitly maximize the distance between different classes by incorporating triplet loss to train the classifier. This not only minimizes inter-class distances but also explicitly maximizes intra-class distances. By doing so, the model is encouraged to learn robust representations that distinctly separate different classes. The use of triplet loss helps to develop a feature space where classes are well-delineated, resulting in clearly separated clusters. This enhances the classifier's discriminative power and its ability to generalize across domains.

Taken together, these two axes allow us to categorize domain generalization techniques based on how they train the feature extractor and classifier components to improve generalization. Methods can differ in whether they align or diversify representations across domains, as well as how they distinguish between classes during training, as illustrated in Figure I.1.

Application of the taxonomy In Figure I.2, we present a decision graph to guide readers through the taxonomy to select the most appropriate meta-learning method for DG based on data availability and domain shifts. It categorizes meta-learning approaches based on the strategies employed for the feature extractor (generalizability axis) and the classifier training process (discriminability axis), facilitating an understanding of the relationships between various methods and their underlying principles.

I.4 Methodologies

In recent years, there has been significant interest in utilizing the meta-learning procedure to create models for DG. In this section, we will explore individual meta-learning methods for DG by describing the methodologies and motivations behind their designs. We discuss the strengths and weaknesses of each method in detail.

Meta-learning focuses on building a general model that can be used to adapt rapidly to new situations. For the first time, Li et al. exploited the idea

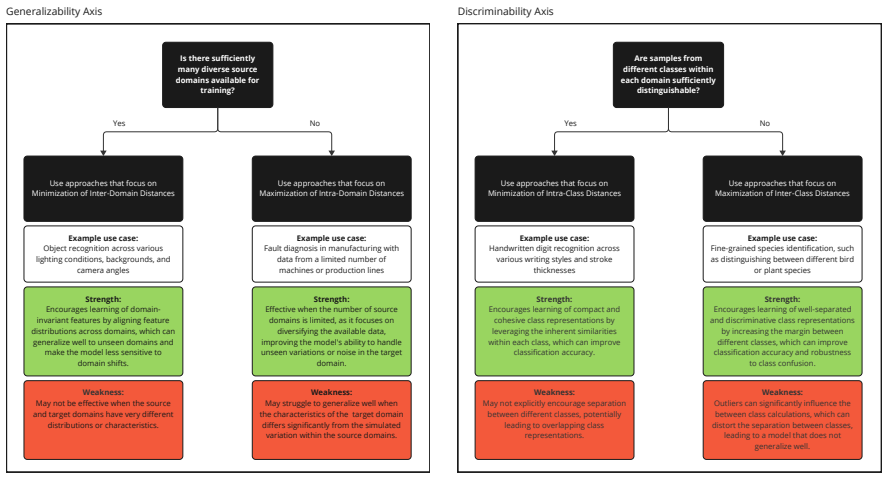


Figure I.2: A decision graph illustrating how to apply the taxonomy of meta-learning approaches for domain generalization. The decision graph categorizes techniques based on the two key aspects: the generalizability axis, which focuses on the feature extractor's strategy (minimizing inter-domain distances or maximizing intra-domain distances), and the discriminability axis, which focuses on the classifier's training process (minimizing intra-class distances or maximizing inter-class distances). This decision graph helps the reader navigate through the taxonomy, providing use cases, strengths, and weaknesses for each category to make the concepts more tangible, applicable, and actionable.

behind meta-learning for DG by dividing the data from multiple source domains into non-overlapping meta-train and meta-test sets to simulate domain shifts in the training procedure. Therefore, we will discuss DG through meta-learning, starting with meta-learning for domain generalization (MLDG) and proceeding accordingly.

I.4.1 MLDG

Recent meta-learning studies have focused on learning good weight initializations for few-shot learning, particularly Model-Agnostic Meta-Learning (MAML) [25]. Li et al. [26] proposed MLDG (meta-learning for domain generalization), which draws inspiration from the MAML approach and trains models capable of performing well on OOD data by transferring knowledge across domains instead of tasks. It is important to note that DG assumes

zero training examples from the target domain, in contrast to conventional meta-learning methods like MAML, which address few-shot scenarios using few training examples of new tasks. Through the meta-learning framework, MLDG will be exposed to domain shifts during training, enhancing its ability to handle domain shifts in various situations by learning general representations. Accordingly, they divide data from multi-source domains into meta-train and meta-test sets to simulate domain shifts. In the meta-train phase, parameters θ of the model are updated by gradient descent using the meta-train sets $\mathcal{S}_{tr}$. As a result, the following steps will be taken:

$$\begin{aligned} \nabla_\theta &= \ell'_\theta(\mathcal{S}_{tr}; \theta), \\ \theta' &= \theta - \alpha \nabla_\theta, \end{aligned} \tag{I.2}$$

where α is the meta-train (inner loop) learning rate. Following that, in the meta-test phase, we determine the loss using the meta-test sets (virtual-test domains) $\mathcal{S}_{te}$ with the parameters obtained in the meta-train phase. Consequently, this phase involves calculating gradients over gradients to update the model's parameters. The parameters update can be formulated as:

$$\theta = \theta - \beta \frac{\partial(\ell(\mathcal{S}_{tr}; \theta) + \gamma \ell(\mathcal{S}_{te}; \theta'))}{\partial \theta}, \tag{I.3}$$

where ℓ is the loss function ,and β is the meta-test (outer loop) learning rate, and γ weights meta-train and meta-test.

MLDG is built on top of MAML and improves the generalization of MAML, which is model-agnostic and easily applicable to reinforcement learning (RL) problems. However, Like MAML, it is computationally expensive since it requires the calculation of second-order derivatives.

I.4.2 MetaReg

Balaji et al. [2] introduced MetaReg (domain generalization using meta-regularization) that incorporates a regularization term trained using meta-learning to encourage models to extract domain-invariant features and reduce sensitivity to domain-specific variations to train models that are robust to domain shifts. Their proposed model consists of a feature network ψ along with p task networks θ_i for p source domains. The feature network is designed to extract more general features, and the tasks networks are utilized to enforce domain-specificity in the networks so that they can apply regularizers to make them domain-invariant. Therefore, they implement the meta-learning framework to train the regularizer term and incorporate it into the loss function. The

regularizer term will be trained using every pair of source domains (a, b). This procedure can be expressed by the following set of equations:

$$\begin{aligned} &\beta^1 \leftarrow \theta_a^{(k)} \\ &\beta^t = \beta^{t-1} - \alpha \nabla_{\beta^{t-1}} [L^{(a)}(\psi^{(k)}, \beta^{t-1}) + R_\phi(\beta^{t-1})] \\ &\quad \forall t \in \{2, \ldots, l\} \\ &\hat{\theta}_a^{(k)} = \beta^l \end{aligned} \tag{I.4}$$

$$\phi^{(k+1)} = \phi^{(k)} - \alpha \nabla_\phi L^{(b)}(\psi^{(k)}, \hat{\theta}_a^{(k)})|_{\phi=\phi^{(k)}} \tag{I.5}$$

Where Eq. (I.4) describes the inner loop of the meta-learning paradigm that performs l steps of gradient descent using meta-train set on a new task network with parameters $\theta_a^{(k)}$ which is the base model's task network parameters of the a^{th} domain at iteration k to obtain $\hat{\theta}_a^{(k)}$. In addition, Eq. (I.5) refers to the outer loop of the meta-learning strategy, which incorporates the meta-test set to minimize the unregularized loss with parameters $\hat{\theta}_a^{(k)}$ with respect to the regularizer parameter ϕ. By using the trained regularizer, we fine-tune a new model on the entire source dataset. To further add, weighted L_1 loss is used as the regularization function, i.e., $R_\phi(\theta) = \sum_i \phi_i |\theta_i|$ to build a learnable weight decay mechanism and diminish domain-specific characteristics in task networks.

Although MetraReg increases the training complexity and requires several domains for practical training, it is easily applicable to deep neural networks and helps reduce overfitting.

I.4.3 Feature-Critic Networks

Feature-Critic Networks introduced by Li et al. [51] is tailored for heterogeneous domain generalization by drawing inspiration from the meta-learning framework. The proposed model meta-learns a loss function that promotes domain robustness. In this way, they train a robust feature extractor that can be used with any classifier for addressing various problems from different domains. More specifically, they introduced a learnable auxiliary loss $\ell_\omega{}^{(\text{Aux})}$ resulting from optimizing the feature-critic network h_ω to encourage the base network to extract domain agnostic features, as illustrated in Figure I.3.

Based on the following equations, parameters θ of the base network is

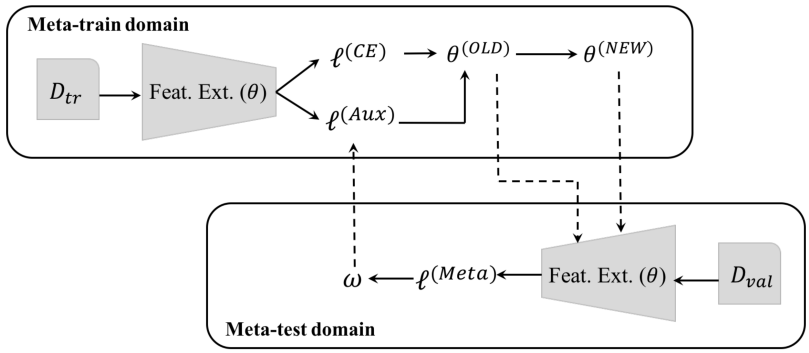


Figure I.3: Visual representation of the Feature-Critic Networks for heterogeneous domain generalization, depicting the base network's feature extraction guided by an auxiliary loss from the feature-critic networks to promote domain-invariant feature extraction. The domain-invariant features are generated by minimizing the inter-domain distances.

computed using the meta-train data $\mathcal{D}_{tr}$:

$$\begin{aligned}\theta^{(\mathrm{OLD})} &= \theta - \alpha\nabla_\theta \ell^{(\mathrm{CE})}(\mathcal{D}_{tr}|\theta),\\ \theta^{(\mathrm{NEW})} &= \theta - \alpha\nabla_\theta (\ell^{(\mathrm{CE})}(\mathcal{D}_{tr}|\theta) + \ell_\omega{}^{(\mathrm{Aux})}(\mathcal{D}_{tr}|\theta)),\end{aligned} \tag{I.6}$$

where $\ell^{(\mathrm{CE})}$ is the cross-entropy loss, and the auxiliary loss $\ell_\omega{}^{(\mathrm{Aux})}$ is defined as follows:

$$\ell_\omega{}^{(\mathrm{Aux})} := \frac{1}{N}\sum_{i=1}^{N} softplus(h_\omega(f_\theta(x_i))), \tag{I.7}$$

where f_θ is the feature extractor. Also, the softplus function is applied to ensure the output is non-negative.

In order to determine parameters ω of the feature-critic network, the following optimization will be used with the meta-test data $\mathcal{D}_{val}$:

$$\min_\omega \ tanh(\ell^{(\mathrm{CE})}(\mathcal{D}_{val}|\theta^{(\mathrm{NEW})}) - \ell_\omega{}^{(\mathrm{CE})}(\mathcal{D}_{val}|\theta^{(\mathrm{OLD})})). \tag{I.8}$$

The tanh function can be considered a softer version of gradient clipping in this context.

Ultimately, we can utilize $g_\phi \circ f_\theta$, where the trained base network f_θ will serve as a fixed feature extractor for target domains, and g_ϕ would be the classifier. For heterogeneous DG, we have N distinct classifiers, denoted as

$g_{\phi_1}, g_{\phi_2}, \ldots, g_{\phi_N}$. On the other hand, in the homogeneous DG, we have a single classifier g_ϕ that can be shared across all domains.

Feature-critic networks can effectively deal with heterogeneous DG where label spaces differ. Nevertheless, there is no guarantee that the gradients of the feature-critic networks will not conflict with those from the supervised loss, which could negatively affect the performance of the model.

I.4.4 Episodic Training for DG

A novel framework for DG is proposed by Li et al. [48] based on episodic training that resembles the meta-learning training approach. In particular, they used episodic training to train a deep neural network, decomposed into feature extractor and classifier components. Each component is trained by simulating interactions with a poorly tuned partner for the current domain. Moreover, to construct episodes, the framework uses data from domain A to be processed by the classifier trained on domain B, which has not been exposed to data from A, and vice versa. This cross-domain exposure ensures that each component becomes sufficiently robust at handling OOD data, as depicted in Figure I.4.

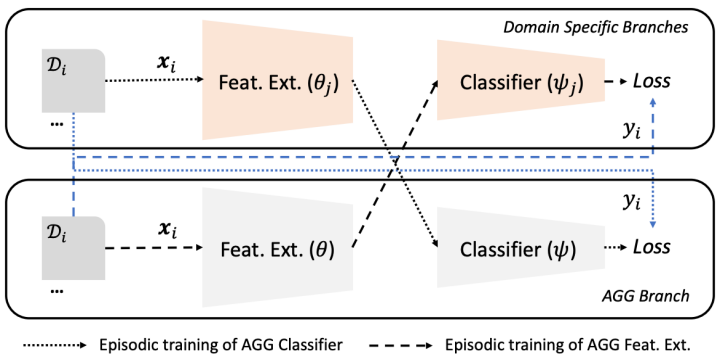


Figure I.4: Overview of Episodic Training for DG framework, illustrating the regularization process where a feature extractor is trained with classifiers from different domains and vice versa, promoting out-of-distribution robustness.

The intuition of the episodic training using a mismatched classifier is to regularize the feature extractor by asking it to produce features robust enough that even an arbitrary classifier will be able to perform properly. Accordingly, three episodic training strategies are introduced: 1) Train feature extractor to support classifiers from other domains; 2) Train classifier to ac-

cept features from other domains; 3) Train feature extractor to support a randomly initialized classifier.

The proposed approach requires only a simple modification of the training procedure and does not rely on special data augmentation or optimization algorithms. It also allows for heterogeneous label spaces across domains since no classifier needs to be shared. However, there is a risk that the feature extractor could learn degenerate representations if the randomness imposed overwhelms the true objectives of the training.

I.4.5 Meta-learning the Invariant Representation

Learning invariant representations across source domains has shown effectiveness for DG. However, overfitting to source domains can limit generalization to a target domain that differs greatly from the sources. Jia et al. [40] proposed a meta-learning algorithm via bilevel optimization to improve the out-of-domain robustness of the learned invariant representation. Most meta-learning algorithms share a common limitation: they use task objectives directly as the inner-loop and outer-loop objectives. This approach can be suboptimal because it is highly abstracted from the feature representation. To address this issue, the authors focus on a meta-learning approach aimed at reducing the discrepancy between the target domain and source domains, where in each training iteration, any domain can become the meta-target domain while the rest serve as the meta-source domains. Specifically, they developed a bilevel meta-learning procedure based on the first-order MAML framework, which achieves high computational efficiency. The paper employs a $\mathcal{Y}$-discrepancy measure [101] to quantify domain discrepancy, effectively capturing both covariate and conditional shifts between domains. Notably, $\mathcal{Y}$-discrepancy has been utilized in previous research for domain invariance learning [102]. The primary goal of this algorithm is to minimize the $\mathcal{Y}$-discrepancy between the source domains and the target domain. A gradient-based meta-learning algorithm is provided to solve the bilevel optimization problem efficiently. The algorithm alternates between sampling meta-tasks, updating the feature embedding and classifiers on the meta-training set, and updating the meta-parameters based on the meta-test performance. The inner-loop objective aims to minimize discrepancy across different source domains, while the outer-loop objective aims to minimize discrepancy between source domains and a potential target domain. In particular, adversarial learning is employed to estimate and minimize the $\mathcal{Y}$-discrepancy between domains in both the inner and outer loops of the bilevel meta-learning algorithm. In essence, this meta-learning approach reduces the discrepancy between the target domain and the convex hull of the source

domains as depicted in Figure I.5. This enables learning a robust invariant representation for improved domain generalization.

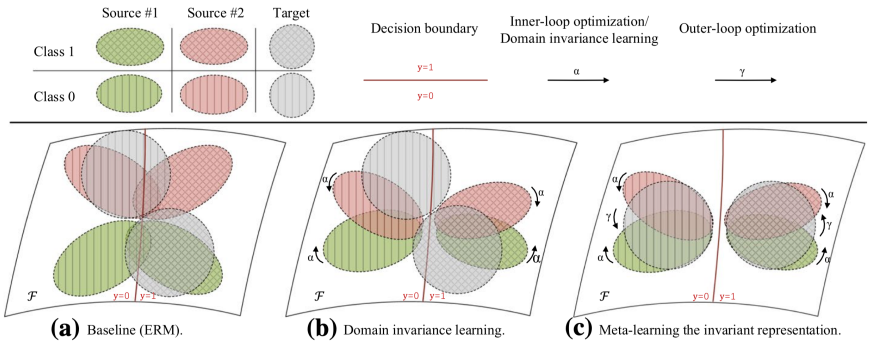


Figure I.5: Depiction of the meta-learning for invariant representation method. **(a)** Compared to the ERM baseline, **(b)** domain invariance learning decreases the discrepancy among source domains and excels in source-domain classification, yet it may still result in significant errors on the target domain. **(c)** The proposed approach employs bilevel meta-learning to further minimize the discrepancy between the target and source domains, enabling a hypothesis learned from the source domains to generalize effectively to the target domain.

The introduced model learns a robust invariant representation that generalizes well to unseen domains by optimizing the feature embedding to minimize the discrepancy between meta-source and meta-target domains during training. However, like other domain generalization methods, the effectiveness of the proposed approach may depend on the diversity of the available source domains. If the source domains lack diversity, the learned invariant representation may not generalize well to significantly different target domains.

I.4.6 MASF

Model-agnostic learning of semantic features (MASF) has been presented by Dou et al. [16], in which two complementary losses that explicitly regularize the semantic structure of the feature space are introduced. This approach aims to enhance domain generalization within the meta-learning framework by splitting source domains into meta-train and meta-test sets to simulate domain shift through an episodic training procedure. In more detail, MASF

suggests enforcing semantic features through global class alignment and local sample clustering, as illustrated in Figure I.6. These supplementary losses are explicitly derived in an episodic learning procedure.

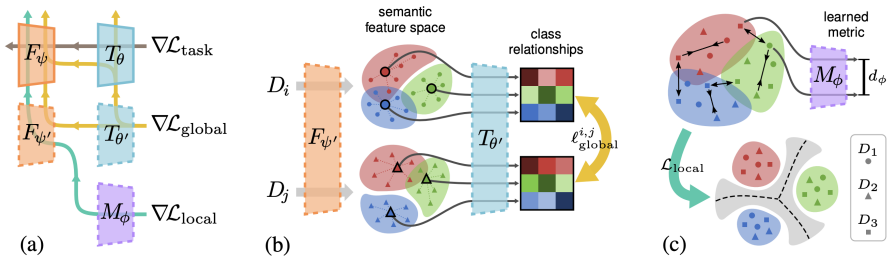


Figure I.6: Overview of the MASF approach for Domain Generalization, F_ψ, and T_θ represent the feature extractor and task network, respectively, with $F_{\psi'}$ and $T_{\theta'}$ being their updated versions after inner gradient updates on the task loss $\mathcal{L}_{task}$. M_ϕ is the metric embedding network, and D_k represents different source domains. (a) Illustration of gradients flow of episodic training with domain shift simulation; (b) Global alignment for consistent class relationships across domains; (c) Local sample clustering to promote the Maximization of Inter-Class Distances.

The objective of global class alignment is to structure the feature space such that it preserves the learned inter-class relationships on unseen data via explicit regularization. By ensuring consistent class relationships, knowledge transfer across domains is facilitated, thus improving domain generalization. This inter-class alignment is done by exploiting what the model has learned about class ambiguities in the form of per-class soft labels and enforcing their consistency within each two distinct domains by minimizing the symmetric KL divergence between their soft confusion matrix created from the collection of soft labels.

The objective of local regularization is to enhance robustness by promoting feature compactness, grouping features of samples within the same class closely together while being distinct from features of different classes. This is crucial to prevent ambiguous decision boundaries and sensitivity to unseen domain shifts. As a result, an embedding network is utilized to process the extracted features. The output of this network is then used to measure the distance between the input features of randomly selected pairs of samples from all domains. The training procedure employs either contrastive loss or triplet loss.

The proposed model captures inter-class relationships and ensures intra-

class compactness, improving the generalization of the model; however, applicability to large-scale problems with many classes may be a challenge.

I.4.7 S-MLDG

Li et al. [46] proposed a framework for sequential learning for domain generalization (S-MLDG) based on the MLDG to enhance its performance by incorporating sequential learning and lifelong learning. They train a base DG model sequentially along a trajectory spanning the source domains, optimizing the cumulative performance across the entire trajectory instead of individual steps. In the case of MLDG, it recursively applies MLDG along the trajectory, simulating lifelong DG learning, as shown in Figure I.7. This approach encompasses more unique DG episodes compared to the base algorithms. While MLDG considers N distinct domain-shot episodes, this method considers N! distinct domain-shift episodes. Furthermore, the approach defines a loss function that aggregates the performance on each step of the trajectory through domains. Additionally, it incorporates shuffling domain orders and sampling new batches for each iteration, ensuring diversity and preventing overfitting.

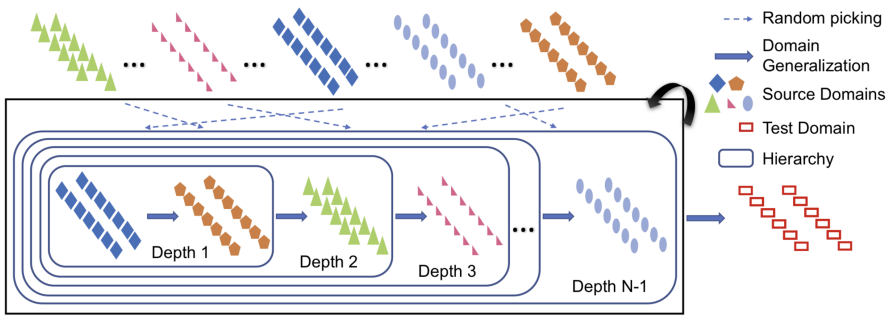


Figure I.7: Illustration of the S-MLDG training procedure, showing the sequential training across multiple domain trajectories with random sequences to simulate lifelong learning and increase the diversity of domain-shift episodes.

The fast first-order approximation, denoted as FFO-S-MLDG, constitutes a streamlined variant of the S-MLDG algorithm, drawing inspiration from Reptile to mitigate its computational overhead. FFO-S-MLDG uses a first-order approximation of the original S-MLDG algorithm, enabling it to be trained more quickly and efficiently.

Although the training complexity is higher compared to MLDG due to

the optimization over domain trajectories, this approach consistently demonstrates improvements over strong baselines on multiple benchmarks. This is due to its ability to generate more unique DG episodes.

I.4.8 MetaVIB

Meta variational information bottleneck (MetaVIB) has been introduced by Du et al. [18] as a probabilistic meta-learning model to handle uncertainty and domain shifts for domain generalization by learning domain-invariant representations using variational auto-encoders. In order to address the uncertainty associated with predictions on the unseen target domain, the model represents classifier parameters as distributions inferred from source domains. In a similar manner to other algorithms, source domains are split into meta-train and meta-test sets in each training episode to simulate domain shift, as illustrated in Figure I.8. MetaVIB is derived from novel variational bounds of mutual information by leveraging the meta-learning setting. Also, the meta-learning objective maximizes performance on meta-test while minimizing domain divergence via MetaVIB. As a result, MetaVIB learns to gradually narrow domain gaps to achieve domain invariance, while maximizing prediction accuracy through episodic training. This encourages learning efficient representations that preserve information for prediction but remove domain-specific statistics, thereby enabling the learning of domain-agnostic representations.

MetaVIB explicitly addresses the uncertainty associated with predictions for new domains through probabilistic modeling. Also, by limiting the information flow from the inputs to only what is relevant for the classification task through the information bottleneck. This prevents the model from encoding irrelevant domain-specific details. However, it does not explicitly align distributions across domains, implying a certain level of domain shift might remain.

I.4.9 M-ADA

Qiao et al. [68] introduced Meta-learning based Adversarial Domain Augmentation (M-ADA) as an effective single domain generalization method. Recognizing that acquiring data from multiple training domains might not be feasible due to constraints such as budget or privacy concerns, they proposed generating fictitious domains through adversarial training to use in the meta-test phase. As shown in Figure I.9, the model consists of a task model and a Wasserstein Autoencoder (WAE).

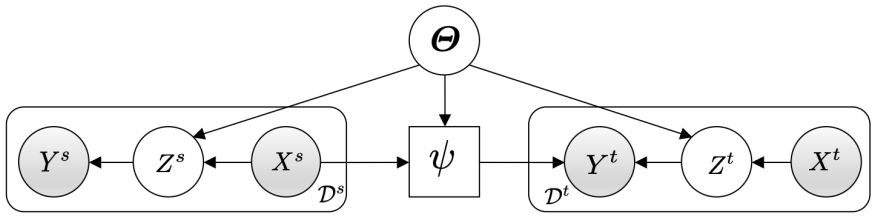


Figure I.8: Computational graph of the MetaVIB model for domain generalization, depicting the training process where domain-invariant representations are learned. Global parameters (Θ) and classifier parameters (ψ) are optimized over source domains, which are split into meta-train ($\mathcal{D}^s$) and meta-test ($\mathcal{D}^t$) sets to simulate domain shifts within each episodic training phase. During the test phase, Θ generates representations for the target domain, while ψ is used for predictions of data from the source domain. While this is a typical example of the inter-domain minimization and intra-class minimization paradigm, the distinctive feature of MetaVIB lies in its explicit probabilistic modeling approach.

The task model with parameters θ encompasses a feature extractor F, which maps the input space to embedding space z, and a classifier C that predicts labels from the embedding space z, using cross-entropy as its loss function $\mathcal{L}_{task}$. Moreover, $\mathcal{L}_{const}$ imposes a semantic consistency constraint on the adversarially augmented samples x^+. It ensures that the augmented domains $\mathcal{S}^+$ remain close to the source domain $\mathcal{S}$ within the embedding space. Accordingly, $\mathcal{L}_{const}$ is defined as:

$$\mathcal{L}_{const} = \frac{1}{2}\|z - z^+\|_2^2 + \infty \cdot 1\ \{y \neq y^+\}, \tag{I.9}$$

where $1\{\cdot\}$ is the 0-1 indicator function, so $\mathcal{L}_{const}$ will be ∞ if x and x^+ have different class labels. Essentially, $\mathcal{L}_{const}$ governs the capacity for generalizing beyond the source domain, a measure facilitated by the Wasserstein distance.

In addition, the task model learns from domain augmentations with the assistance of a WAE, denoted as V, with parameters ψ. V consists of an encoder $Q(x|e)$ and a decoder $G(x|e)$, where x and e denote the input and bottleneck embedding respectively. Also, they utilized Maximum Mean Discrepancy (MMD) as the distance metric ($\mathcal{D}_e$) to minimize the divergence between $Q(x)$ and a priori distribution $P(e)$. Therefore, we can pre-train V using the following optimization on the source domain $\mathcal{S}$:

$$\min_{\psi}\ \|G(Q(x)) - x^2\| + \lambda \mathcal{D}_e(Q(x), p(e)), \tag{I.10}$$

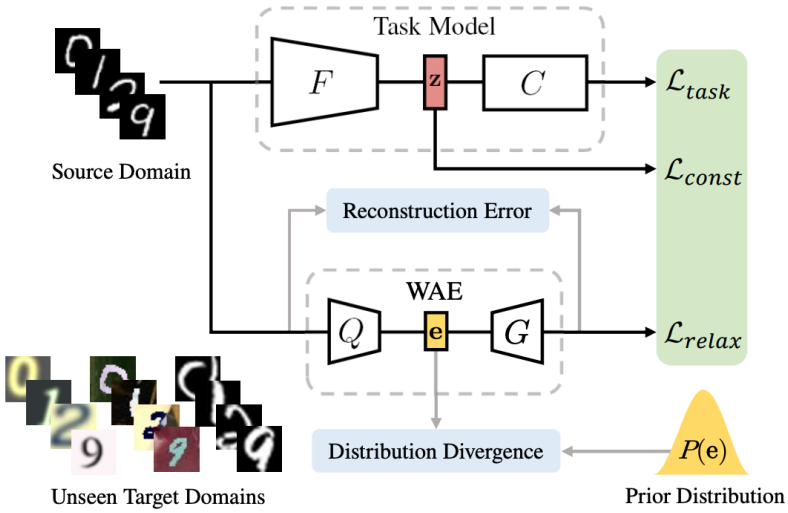


Figure I.9: Overview of M-ADA method for single domain generalization using adversarial domain augmentation. The architecture employs a task model comprising a feature extractor F and classifier C, alongside a Wasserstein Autoencoder (WAE), to generate fictitious domains.

where λ is a hyper-parameter. Following this, the reconstruction error $\mathcal{L}_{relax}$ should be maximized for domain augmentation:

$$\mathcal{L}_{relax} = \|x^+ - V(x^+)\|^2. \tag{I.11}$$

WAEs use the Wasserstein distance metric to measure the distribution distance between the input and reconstruction. As such, the pre-trained V is better equipped to capture the distribution of the source domain, and maximizing $\mathcal{L}_{relax}$ ensures larger domain transportation. Finally, the overall loss function is:

$$\mathcal{L}_{ADA} = \mathcal{L}_{task}(\theta; x) - \alpha \mathcal{L}_{const}(\theta; z) + \beta \mathcal{L}_{relax}(\psi; x), \tag{I.12}$$

where α and β are two hyper-parameters used to balance $\mathcal{L}_{const}$ and $\mathcal{L}_{relax}$.

M-ADA offers a novel adversarial domain augmentation approach using relaxed constraints; however, it requires pre-training a Wasserstein autoencoder, which subsequently adds computational overhead.

I.4.10 MetaNorm

While batch normalization is essential in training deep neural networks, small batch sizes and distribution shifts can destabilize batch statistics. MetaNorm, introduced by Du et al. [19], ensures effective batch normalization under such conditions. It uses a meta-learning setting to infer adaptive normalization statistics from limited samples instead of relying on direct calculations of batch statistics, which can be unreliable with small batches or distribution shifts. Unlike the transductive batch normalization (TBN) method, which uses the same statistics for both meta-train and meta-test phases, MetaNorm applies a non-transductive approach. Given that test samples may not always be available, MetaNorm is designed to learn to generate statistics only from the support set, and at the meta-test time, it directly applies the model to infer statistics for new tasks.

MetaNorm employs two separate feed-forward networks known as hypernetworks. One is for inferring μ, denoted as $f_{\mu}^{\ell}(\cdot)$, and the other for inferring σ, denoted as $f_{\sigma}^{\ell}(\cdot)$, in order to find the statistics for each individual channel in each ℓ convolutional layer in the meta-learning model. In essence, the hypernetworks accept sample activations as input and produce the estimated statistics as output. During the meta-training phase, the estimated statistics of the support set are applied for normalization of both support and query samples:

$$a' = \gamma \left(\frac{a - \mu_s}{\sqrt{\sigma_s^2 + \epsilon}} \right) + \beta, \tag{I.13}$$

where γ and β are jointly learned with parameters of the hypernetworks during the meta-training and directly applied at meta-test time, and ϵ is a small scalar to prevent division by zero.

As a final step, it minimizes the following KL term to learn the normalization statistics:

$$\sum_{i}^{|\mathcal{D}^s|} D_{KL}[q_{\phi}(m|a_i) || p_{\theta}(m|\mathcal{D}^s)], \tag{I.14}$$

where $q(m|a_i)$ and $p(m|\mathcal{D}^s)$ are defined as a Gaussian distribution based on their respective estimated statistics, and also a_i is a sample from the meta-source domain $\mathcal{D}^s$. MetaNorm learns to generate its own appropriate statistics and applies them to the samples in the meta-target domain. Notably, in the meta-target domain, the KL term is not used; instead, each example generates its own normalization statistics.

Although adding hypernetworks to infer normalization statistics slightly increases the overall model complexity, MetaNorm overcomes the issues of batch normalization with small batches and domain shifts, offering a promising approach to domain generalization through meta-learning.

I.4.11 DADG

Discriminative Adversarial Domain Generalization (DADG) is a novel framework proposed by Chen et al. [10] that integrates Discriminative Adversarial Learning (DAL) for extracting domain-invariant features and Meta-learning-based Cross-Domain Validation (Meta-CDV) for training a robust classifier. DAL aims to learn a domain-invariant feature representation to minimize domain variance. It achieves this by training a feature extractor to confuse a domain discriminator, which is tasked with predicting the domain of the features. Further, as depicted in Figure 1, the Gradient Reversal Layer (GRL) is placed between the feature extractor and the domain discriminator in the DAL component. During forward propagation, the GRL functions as an identity transform, conveying the features unchanged to the discriminator. Conversely, during backpropagation, the GRL reverses the gradient by multiplying it by a negative scalar (e.g., -λ), thereby updating the parameters of the feature extractor θ^m into θ^{m+1}. This reversed gradient, originating from the discriminator, encourages the feature extractor to create domain-invariant representations that deceive the discriminator.

Subsequently, Meta-CDV applies the feature extractor to supervised learning by training a classifier c_φ with parameters φ. This is done in a meta-learning manner, where the classification model will be trained on the domains seen in the previous step to update θ^{m+1} to θ^{m+2} and update φ^m to φ^{m+1}. The model's performance is then validated on cross domains to enhance the classification model. It should be noted that this evaluation is performed using the updated parameters θ^{m+2} and φ^{m+1}. In essence, the optimization of the classification model involves the third derivative with respect to θ and the second derivative with respect to φ. Particularly, Meta-CDV simulates train/test domain shifts by training the model on seen domains and validating it on a held-out domain, thereby making the classifier more robust.

DADG learns domain-invariant features using adversarial discriminative learning with GRL, which enables the adversarial mechanism in a simple way. However, the effectiveness of the discriminative adversarial learning component can diminish as the number of source domains increases.

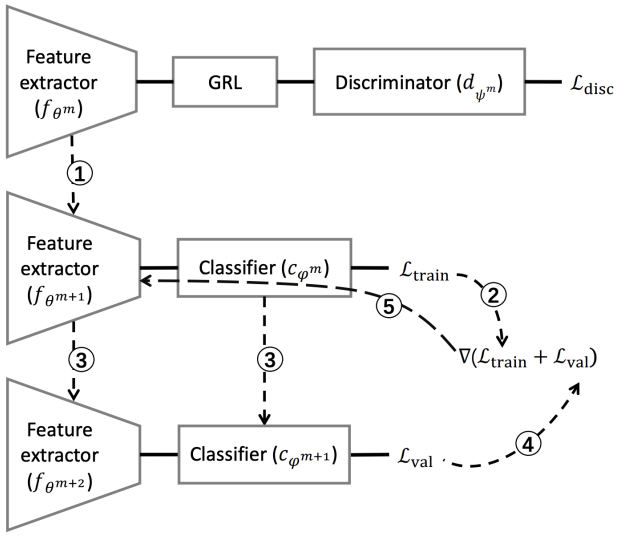


Figure I.10: Illustration of the DADG framework, which combines Discriminative Adversarial Learning (DAL) featuring a Gradient Reversal Layer (GRL) for creating domain-invariant features with Meta-learning-based Cross-Domain Validation (Meta-CDV) to improve the performance of classifiers across unseen domains.

I.4.12 Uncertainty-guided Model Generalization

Qiao et al. [67] proposed a unique solution for single domain generalization using Bayesian meta-learning, called uncertainty-guided model generalization. The primary strategy of this method is to augment the source domain's capabilities, not only in the input space, as most current data augmentation methods do, but also in the label space, guided by an uncertainty assessment. As illustrated in Figure I.11, instead of directly augmenting the input space, they introduce an auxiliary network $\psi = \{\phi_p, \phi_m\}$ to create feature perturbations h^+ by adding softplus of Gaussian noise $e \sim \mathcal{N}(\mu, \sigma)$ to generate new domain $\mathcal{S}^+$ from domain $\mathcal{S}$. The distribution parameters (μ, σ) represent the uncertainty with respect to the backbone θ. This uncertainty is further utilized to predict learnable parameters (a, b, τ), which are used to construct learnable label mixup. In learnable label mixup, learnable parameters (a, b) are used to define $\lambda \sim \text{Beta}(a, b)$ to mixup $\mathcal{S}$ and $\mathcal{S}^+$ to achieve in in-between domain interpolations. Notably, learnable parameters (a, b) control the direction and strength of domain interpolations. As a result, the following

equations are used:

$$\begin{aligned} h^+ &= \lambda h + (1-\lambda)h^+, \\ y^+ &= \lambda y + (1-\lambda)\tilde{y}, \end{aligned} \tag{I.15}$$

where $\tilde{y}$ represents a label-smoothing version of y. Particularly, label smoothing is carried out with a probability of τ, which means that we allocate a value of $\rho \in (0,1)$ to the true class and equally distribute $\frac{1-\rho}{c-1}$ to the others, where c represents the number of classes. Moreover, by learning a and b, the model can learn to modulate the Beta distribution to produce optimal λ values that result in efficient interpolations between $\mathcal{S}$ and $\mathcal{S}^+$. Accordingly, this process results in consistent domain shifts in both input and output via ϕ_p and ϕ_m, enhancing generalization across unseen domains.

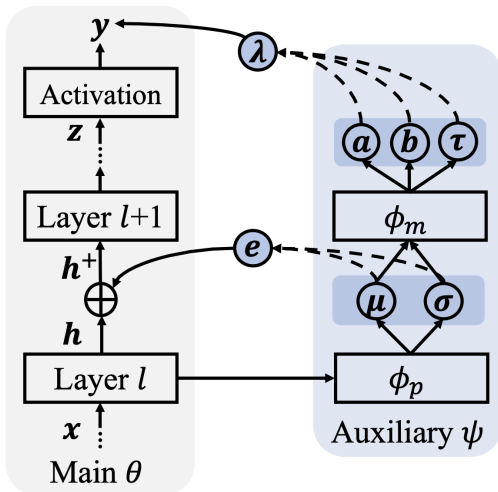


Figure I.11: Schematic representation of the uncertainty-guided model generalization approach, where an auxiliary network ψ introduces feature perturbations to form a new domain $\mathcal{S}^+$. This process utilizes uncertainty in the form of distribution parameters (μ, σ) to guide the creation of domain-augmented data and label mixup, thereby enabling the model to generalize to unseen domains by interpolating between the original and perturbed features and labels.

The authors proposed a practical Bayesian meta-learning framework that optimizes θ and ψ to maximize the posterior $p(\mathcal{S}^+)$. In other words, they utilized Bayesian inference to maximize the posterior of domain augmentations, approximating the distribution of unseen domains.

While uncertainty-guided augmentation provides a mechanism to approximate unseen target distributions, relying solely on randomized sampling procedures to cover the wide range of target domains may be inadequate for real-world distribution shifts.

I.4.13 Memory-based Multi-source Meta-learning

Zhao et al. [105] proposed a memory-based meta-learning method to address multi-source DG for person re-identification (Re-ID) called Memory-based Multi-source Meta-Learning (M^3L). The purpose of person Re-ID is to match persons with the same identity across multiple camera views. Existing works have shown the effectiveness of meta-learning in classification tasks, but its parametric classifier falls short for Re-ID. This is due to the open-set nature of Re-ID tasks, with each domain having numerous and unique identities. Consequently, the M^3L framework is equipped with a memory-based module that applies the identification loss in a non-parametric way, preventing instability in meta-optimization commonly associated with traditional parametric methods. Specifically, this framework maintains a feature memory $\mathcal{M}^i$ for each source domain $\mathcal{D}_{\mathcal{S}}^i$ consisting of n_i slots, where each slot saves the feature centroid of the corresponding identity. Identification loss is then computed using the similarities between these features and memory centroids as follows:

$$\mathcal{L}_M = -\log \frac{\exp\left(M[i]^T f(x_i)/\tau\right)}{\sum_{k=1}^{n_i} \exp\left(M[k]^T f(x_i)/\tau\right)}, \tag{I.16}$$

where $f(\cdot)$ is the feature extractor and τ is the temperature factor. In addition, the following triplet loss is utilized to train the model:

$$\mathcal{L}_{Tri} = [d_p - d_n + \delta]_+, \tag{I.17}$$

where d_p represents the Euclidean distance between an anchor feature and a hard positive feature, while d_n signifies the Euclidean distance between an anchor feature and a hard negative feature. Also, δ is defined as the margin of the triplet loss and $[\cdot]_+ = max(\cdot, 0)$. As a result, the sum of $\mathcal{L}_{Tri}$ and $\mathcal{L}_M$ constitutes the meta-train loss ($\mathcal{L}_{mtr}$) and meta-test loss ($\mathcal{L}_{mte}$).

Furthermore, M^3L incorporates a meta batch normalization layer (MetaBN) that injects meta-train feature statistics into meta-test features, diversifying them and enabling the model to simulate an expanded range of feature variations. In this way, through iterative generalization processes from meta-train to meta-test domains, the model avoids overfitting due to domain bias and

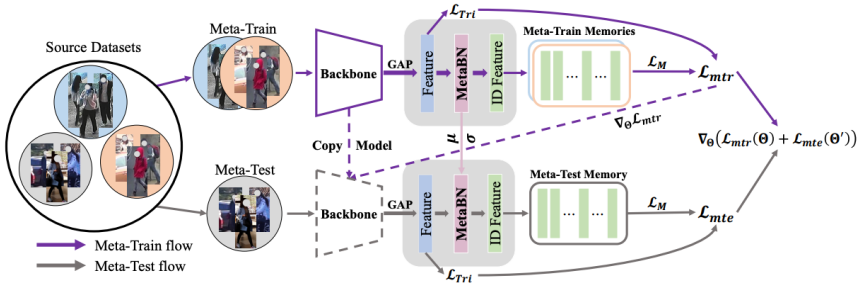


Figure I.12: Overview of the M^{3}L framework designed for person Re-ID. The training process includes dividing the source domains into one meta-test domain and multiple meta-train domains in each iteration. The model utilizes memory-based identification and triplet loss for the meta-training phase and then optimizes the meta-test loss on a copy of the model that has been updated with the meta-train loss (maximization of inter-class distance). MetaBN is also applied during the meta-test stage to enhance feature diversity (maximization of intra-domain distance). Finally, the aggregated meta-train and meta-test losses are used to update the original model for improved domain generalization.

acquires domain-invariant representations that generalize well on unseen domains.

During training, as shown in Figure I.12, source domains are divided into meta-train and meta-test sets at each iteration, employing a meta-learning approach to simulate the train-test process of DG. Therefore, it is necessary to consider the following optimization to build generalizable representations:

$$\min_{\Theta} \mathcal{L}_{mtr}(\Theta) + \mathcal{L}_{mte}(\Theta'), \tag{I.18}$$

where Θ represents the network parameters, while Θ' denotes the parameters of the model optimized by $\mathcal{L}_{mtr}$.

Although MetaBN further enhances generalization ability by diversifying meta-test features, the memory module adds computational overhead in maintaining and updating centroid features for each identity.

I.4.14 MetaBIN

Choi et al. [11] proposed Meta Batch-Instance Normalization (MetaBIN) as an effective heterogeneous single domain generalization method for person

re-identification (Re-ID). The main goal is to simulate unsuccessful generalization scenarios by combining batch-instance normalization layers with meta-learning to address challenging cases caused by both batch and instance normalization layers. A key feature of MetaBIN is its learnable balancing parameters between Batch Normalization (BN) and Instance Normalization (IN), which, depending on their bias, cause the DG model to experience under-style-normalization and over-style-normalization scenarios during meta-learning. Under-style-normalization occurs in the BN model when the model struggles to distinguish identities of samples with unexpected styles from unseen target domains. Conversely, over-style-normalization arises in the IN model, which, while efficient at eliminating instance-specific style information, can inadvertently filter out discriminative information; refer to Figure I.13 for more information.

MetaBIN consists of a classifier g_ϕ with parameter ϕ to predict identities and a feature extractor f_θ with parameters θ where $\theta = (\theta_f, \theta_\rho)$. Here, θ_ρ signifies the balancing parameters between BN and IN, and θ_f represents the remaining parameters.

In the base model updating process, classifier parameters ϕ and feature extractor parameters θ_f are updated by minimizing the cross-entropy loss along with the triplet loss according to the following:

$$\mathcal{L}_{tri} = [d_p - d_n + \delta]_+, \tag{I.19}$$

where d_p represents the Euclidean distance between an anchor feature and the hardest positive sample, while d_n signifies the Euclidean distance between an anchor feature and the hardest negative sample. Also, δ is defined as the margin of the triplet loss and $[\cdot]_+ = max(\cdot, 0)$.

In the meta-learning step, source domains are split into meta-train set $\mathcal{D}_{mtr}$ and meta-test set $\mathcal{D}_{mte}$. First, in the meta-training phase, only θ_ρ is updated to obtain θ'_ρ using $\mathcal{D}_{mtr}$ to simulate over-style-normalization and under-style-normalization cases via the aggregation of the following losses: the intra-domain scatter loss ($\mathcal{L}_{scat}$), the inter-domain shuffle loss ($\mathcal{L}_{shuf}$), and the triplet loss ($\mathcal{L}_{tri}$). The intra-domain scatter loss is used to spread the feature distribution for each domain by minimizing the cosine similarity between each sample and its domain centroid. The inter-domain shuffle loss is introduced to shuffle or mix up the distributions across different source domains by minimizing the distance between anchor samples and inter-domain negatives while maximizing the distance between anchor samples and intra-domain negatives. In summary, the intra-domain scatter loss $\mathcal{L}_{scat}$ and inter-domain shuffle loss $\mathcal{L}_{shuf}$ are used to simulate over-style-normalization scenarios where styles are confused. At the same time, adding the triplet loss

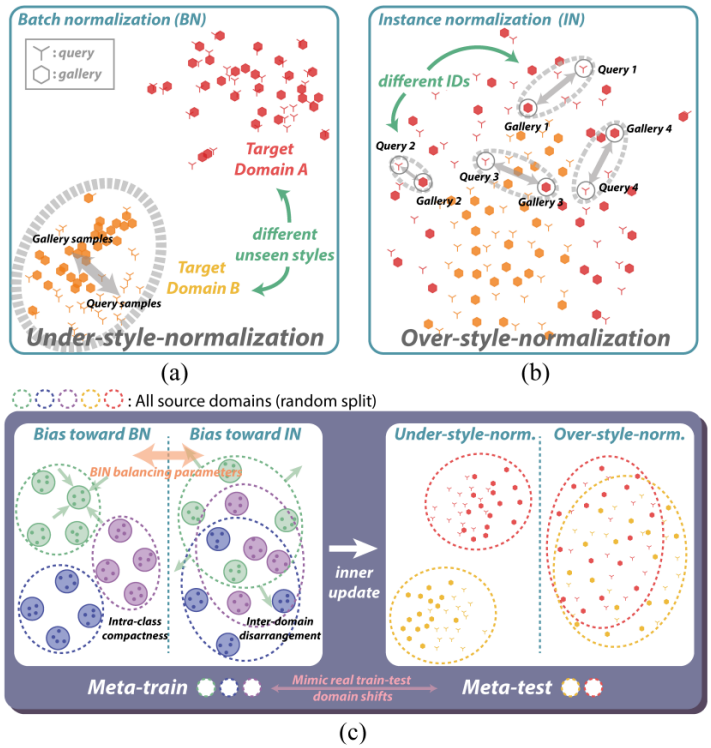


Figure I.13: Visualized here are scenarios where generalization is unsuccessful, along with the MetaBIN framework. (a) Under-style-normalization where, the BN model fails to recognize identities in novel domains due to a lack of style variation in the training data. (b) Over-style-normalization, where the IN model excessively removes style information, including identity-discriminative features. (c) The main idea behind the MetaBIN framework is to adjust the balance between BN and IN, simulating and learning from both types of normalization errors within a meta-learning framework, thereby enhancing the model's ability to generalize to unseen domains without overfitting to source domain styles.

enhances intra-class compactness regardless of style differences to simulate under-style-normalization. Next, in the meta-testing phase, the model with parameters θ_f and θ'_ρ is evaluated on $\mathcal{D}_{mte}$, and θ_ρ will be updated using $\mathcal{L}_{tri}$. Note that θ_ρ contains channel-wise balancing parameters ρ for each normalization layer, with some channels biasing θ_ρ toward IN to remove

unessential style info, while others retain BN properties.

MetaBIN improves generalization to unseen domains without needing extra networks or data augmentation; however, it may not perform well on datasets with complex variations or highly different styles between domains, as it relies on the simulated scenarios to learn a robust representation.

I.5 Datasets and Evaluations

In the DG and domain adaptation (DA) fields, datasets are specifically constructed to represent various 'domains'—each a unique distribution of data that captures a certain variation in the input space. These variations can include changes in visual appearance, sensor modality, environmental conditions, or even different tasks. The purpose of these multi-domain datasets is to simulate real-world scenarios where a model trained on limited domains must perform well on another unseen domain. The richness and diversity of domains within these datasets are pivotal to developing and benchmarking algorithms that can generalize beyond their training data.

Several important datasets, commonly used in the fields of DG and DA, have been examined in studies that leverage meta-learning for DG. Given the fundamental overlap between DA and DG, datasets from DA can also be applied to DG purposes, and vice versa. Each dataset, associated with a specific application, is summarized in Table I.3 and will be discussed in this section. This discussion will be followed by an overview of the evaluation strategies commonly employed in DG, which differ from the more straightforward evaluation systems used in DA.

I.5.1 Datasets

In the DG landscape, datasets serve as critical benchmarks for evaluating algorithm performance across varied scenarios. We begin with Rotated MNIST [28], a synthetic dataset based on the original MNIST. It comprises six domains, each containing images of various digits modified by rotations ranging from 0 to 90 degrees at 15-degree intervals. This dataset is instrumental in assessing DG algorithms for handwritten digit recognition. Additionally, to serve a similar purpose, another dataset, known as Digits-Five [110], has been introduced, which includes five sub-datasets: MNIST [44], MNIST-M [6], SVHN [62], SYN [6], and USPS [39], where each of them can be considered a different domain.

For object recognition, a task particularly sensitive to domain shifts, notable benchmarks include VLCS [22], Office-Home [22], PACS [27], CIFAR-

Table I.3: Overview of Key Datasets Used in Domain Generalization Based on Their Application, Highlighting the Number of Domains, Number of Classes, Number of Samples, and Their Description.

	# Domains	# Classes	# Samples	Description
Handwritten Digit Recognition				
Rotated MNIST [28]	6	10	70,000	Rotations (0, 15, 30, 45, 60, and 75)
Digits-Five [110]	5	10	215,695	MNIST [44], MNIST-M [6], SVHN [62], SYN [6], and USPS [39]
Object Recognition				
VLCS [22]	4	5	10,729	VOC2007 [21], LabelMe [76], Caltech-101 [24], and SUN09 [98]
Office-Home [22]	4	65	15,588	Art, Clipart, Product, and Real-World
PACS [27]	4	7	9,991	Photo, Art painting, Cartoon, and Sketch
CIFAR-10-C [35]	-	10	60,000	Artificial Corruptions of CIFAR-10 [42]
Visual Decathlon [70]	10	-	1,659,142	10 Different Datasets ([4, 12, 42, 43, 59, 61, 62, 63, 75, 83, 84])
Action Recognition				
IXMAS [96]	5	11	1,650	5 Different Camera Views, and 11 Different Subjects ([49])
Semantic Segmentation				
SYNTHIA [73]	15	13	2,700	5 Different Locations, and 5 Different Weather Conditions ([90])
Person Re-Identification				
Market-Duke [72]	2	-	69,079	Market-1501 [107] and DukeMTMC-ReID [108]
Sentiment Classification				
Amazon Reviews [5]	4	2	> 340,000	Books, DVDs, Electronics, and Kitchen Appliances

10-C [35], and VD [70]. VLCS contains images from four domains (VOC2007 [21], LabelMe [76], Caltech-101 [24], and SUN09 [98]) (see Figure I.14) and five classes. Office-Home includes images of objects from Art, Clipart, Product, and Real-World domains, spanning 65 categories. PACS covers four contrasting domains (Photo, Art painting, Cartoon, and Sketch as depicted in Figure I.14) with seven object classes. VD contains ten diverse domains, including handwritten characters, pedestrians, traffic signs, etc., with varying image categories and sizes suitable for heterogeneous DG. Also, CIFAR-10-C [35] is a robustness benchmark comprising CIFAR-10 [42] test images corrupted through 19 distortion types across five severity levels.

IXMAS [96] dataset addresses the domain generalization challenge in action recognition, which relies on learning generalizable representations within video understanding. It includes recordings of 11 actions performed by various actors, captured from multiple angles and with different cameras, to introduce domain shifts.

In autonomous vehicle development, semantic segmentation is essential, yet deep neural networks have not fully bridged the performance gap in unexplored environments. SYNTHIA [73] dataset, with its synthetic images of different locations and weather conditions, supports research to overcome this limitation.

Person Re-Identification (Re-ID) is another area where DG is paramount, especially for security and surveillance. While traditional Re-ID has been

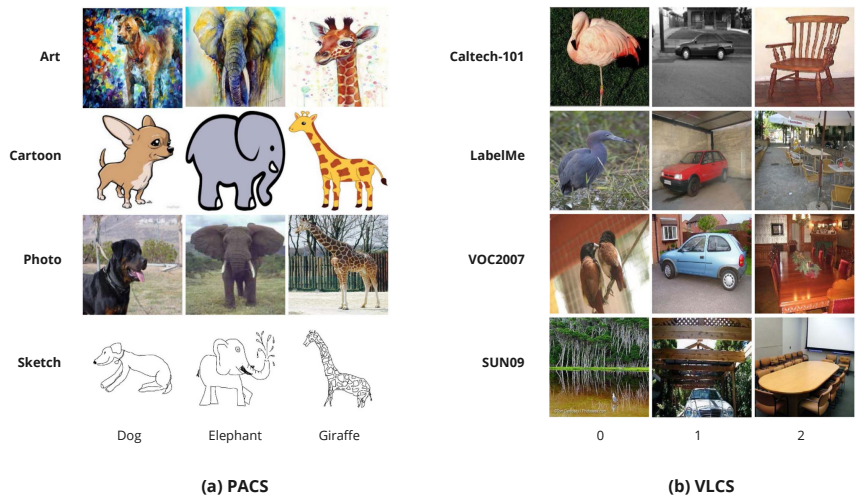


Figure I.14: Example images from two prominent domain generalization benchmarks, illustrating different types of domain shifts. In (a), the PACS dataset highlights domain shifts primarily due to changes in image style. In (b), the VLCS dataset reveals domain shifts caused by variations in environment, scene, and viewpoint, reflecting dataset-specific biases.

confined to the same camera views, cross-dataset Re-ID seeks to generalize from source to target views across varied resolutions, viewpoints, and lighting conditions. Key datasets like Market-1501 [107] and DukeMTMC-ReID [108] are essential in the advancement of this field.

Lastly, the Amazon Reviews [5] dataset, containing reviews from domains such as books, DVDs, electronics, and kitchen appliances, is crucial for sentiment classification in natural language processing.

I.5.2 Evaluations

For evaluating domain generalization algorithms, three principal strategies are commonly employed: (i) *Leave-one-domain-out validation*, which is the most prevalent strategy that is applicable when multiple source domains are available during training. It involves leaving out one source domain as the validation set while using the others for training. However, the final results heavily rely on the selection of the validation domain, leading to results that may lack stability. (ii) *Training-domain validation*, where a subset of the training data is held out for validation to select the best model. How-

ever, since there is still a divergence between real-world unseen data and the training subset, it may not achieve optimal performance. (iii) *Test-domain validation*: which involves using a random subset of the target domain data for model selection. This method can lead to the best performance since the validation and test data share the same distribution. However, it presupposes access to the target domain data, which may not be feasible in real-world applications. Using test domain data for model selection can yield misleading results, as an unrepresentative or excessively easy subset may result in optimistic performance, or a challenging subset may lead to pessimistic results [31]. Careful interpretation of results obtained using this method is therefore necessary.

In general, the average accuracy is used as the primary metric to report the performance of a DG model across held-out domains. For image classification on Cifar-10-C [35], the mean Corruption Error (mCE) metric determines model robustness to corruptions, while Relative mCE (RmCE) compares corruption robustness relative to a baseline by accounting for clean data performance. This enables fairer comparisons between models. For semantic segmentation models evaluated on the SYNTHIA [73] dataset, the standard mean Intersection Over Union (mIoU) is calculated for each unseen domain to quantify segmentation accuracy. Additionally, the VD-score metric assesses models trained on the diverse VD [70] dataset by evaluating performance across its ten distinct image classification tasks.

I.6 Applications

In real-world scenarios, finding sufficient labeled data to effectively train a model often proves challenging. This difficulty is compounded when the model encounters OOD data during testing. As such, domain generalization through meta-learning has emerged as a highly desirable solution. It enables a model to be trained on one or multiple domains and to be applied to unseen domains without the need for retraining. This approach not only alleviates the risk of overfitting but also diminishes the costs and time associated with domain-specific retraining, thus boosting the model's adaptability and operational efficiency.

The applicability of domain generalization models is evident across various disciplines, such as medical imaging, intelligent fault diagnosis, computer vision, and natural language processing. For example, Liu et al. [54] utilized meta-learning for domain generalization to segment prostate MRI images, demonstrating its potential in medical analysis. Ren et al. [71] pro-

posed Meta-GENE, a model-agnostic meta-learning framework designed for fault diagnosis in industrial prognostics and health management scenarios, which can operate in both homogeneous and heterogeneous DG settings. Wang et al. [91] applied the technique for cross-lingual semantic parsing, and Balaji et al.[2] utilized it for sentiment analysis within the Amazon Review dataset. In the area of person re-identification (Re-ID), both Choi et al. [11] and Zhao et al. [105] leveraged meta-learning for DG to recognize individuals across varying postures and perspectives. Moreover, meta-learning for DG has been applied in cross-view action recognition by Li et al. [48] and robot control through reinforcement learning by Li et al. [26]. Qiao et al. [68] showcased its utility in semantic segmentation for street scenes, and Dou et al. [16] explored its application in multi-site brain tissue segmentation.

Further extending the applications, DG has been employed in face anti-spoofing [79], image compression [103], and various medical diagnostics, including Parkinson's disease detection [60], activity recognition [20], chest X-ray analysis [58], and EEG-based seizure detection [1]. Other studies have used domain generalization for speech utterance recognition [66, 78], fault diagnosis [50, 52, 106], and brain-computer interaction [32]. Additionally, DG has found its place in brain-computer interfacing [32], and promising advancements have been made in time series forecasting for financial markets [17].

The extensive applications of meta-learning for DG position it as a focal point of interest for future research and commercial endeavors. The algorithms developed under this paradigm equip models with rapid generalization and adaptation capabilities, even in the absence of target domain data. Such a capacity is a stepping stone towards creating more powerful AI systems capable of tackling diverse real-world problems. By integrating domain generalization with meta-learning, we can unlock zero-shot learning capabilities for novel, unseen domains, enabling generalization across various tasks and domains without requiring additional data.

I.7 Discussion and Future Directions

This survey has examined diverse strategies for DG using meta-learning. The presented taxonomy provides a structured view of the field, categorizing approaches based on the treatment of the input domain and classifier training strategies, as discussed in Section I.3. Initially, DG models employing meta-learning predominantly focused on the minimization of inter-domain distances and the minimization of intra-class distances, aiming to find a com-

mon representation across domains and cohesive grouping of class instances, as this approach is the most intuitive way to develop DG methods using meta-learning. However, recent algorithms have been exploring the effectiveness of the Maximization of Inter-Class Distances or the Maximization of Intra-domain Distances to fully exploit the advantages of meta-learning. For instance, MASF [16] leverages the Maximization of Inter-Class Distances. It is the first model in meta-learning for DG that follows this paradigm, resulting in improved class discriminability and enhanced generalization to unseen domains. Furthermore, M-ADA first introduced the concept of maximizing intra-domain distances through meta-learning, laying the groundwork for future models that employ this strategy. This approach aims to diversify the input feature space, thereby enhancing the robustness of the feature extractor, particularly when there are limited training domains available. Recently, methods such as Memory-Based Multi-Source Meta-Learning (M^3L) [105] and MetaBIN [11] have been developed, leveraging the Maximization of both Intra-Domain and Inter-Class Distances. This strategic combination is designed to diversify the feature space within each domain while simultaneously ensuring that features remain discriminative for classification tasks. These methods are beneficial for use cases where different classes are not easily distinguishable, such as Re-Identification (Re-ID), by further diversifying input features and enhancing class discriminability.

The choice of strategy in DG models tends to vary depending on the nature and number of available training domains. When multiple diverse domains are available for training, employing a technique that minimizes inter-domain distances is a natural choice, as it exploits the diversity of the available data to extract domain-invariant features. On the other hand, if the training data is limited to a few domains, models that maximize inter-domain distances are preferable, as they focus on augmenting the input or feature space to create a generalizable model capable of performing well across various domains during inference. In particular, recent advances in generative AI allows more realistic synthetic data generation and diversification, which further enhance algorithms that maximize intra-domain distances.

Additionally, in scenarios where data points within a dataset are highly similar to each other, it is beneficial to utilize models that emphasize the Maximization of Inter-Class Distances. This approach encourages a clear separation between classes, enabling the model to learn effective embedding representations that facilitate discrimination between classes based on distance.

To alleviate the weaknesses of existing meta-learning approaches for DG which has been explored in Figure I.2, a promising future direction is to cre-

ate a novel distance metric that that captures the intra- and inter-domain characteristics. This metric can be utilized as a guidance for selecting appropriate methods to enable rapid adaptation to new domains.

It is also worth mentioning that the exploration of biologically plausible models in meta-learning has gained some attention and aims to bridge the gap between artificial learning systems and the mechanisms underlying learning in biological organisms. While artificial neural networks have achieved impressive results on meta-learning benchmarks, they differ substantially from the learning processes in biological systems, such as the human brain. As of recent, there has been growing interest in developing meta-learning models that are more biologically grounded and mimic the human learning system. One such model incorporates a memory system designed to prevent catastrophic forgetting, functioning similarly to an episodic memory system [29]. This model enhances the generalization capabilities of spiking neural networks by simulating an efficient episodic memory capable of storing vast amounts of information and linking similar underlying patterns. A promising direction for future development is to create innovative policies for memory adaptation within this model to handle more complex tasks, such as DG. By implementing these policies, the model can better ignore noise and outlier data and map similar classes from different domains to the same memory representation. This would empower the model to effectively tackle data from diverse domains, enhancing its generalization capabilities.

DG is a critical challenge in federated learning, where multiple clients collaboratively train a model without sharing their raw data, enhancing privacy and security [100]. However, the domain of the target test data on the server can differ greatly from the training data of each client. This discrepancy leads to decreased performance of the federated model on the target domain. RFDG [30] has integrated domain generalization into federated learning via a reinforcement learning-based feature decorrelation policy that enables clients to reweight samples from a global perspective to learn domain-invariant knowledge. The replay mechanism also addresses challenges posed by mini-batch training. This field represents a potential area where we should incorporate meta-learning for DG to enhance adaptability to different clients with limited labeled data. By leveraging meta-learning techniques, the federated model can quickly adapt to new clients by learning from a small number of labeled examples, even when there are significant distributional shifts between the source and target domains. Accordingly, meta-learning can be used to learn a shared feature extractor that is robust to domain shifts, allowing the model to extract transferable knowledge from the source domains and effectively apply it to the target domain, thereby im-

proving its ability to generalize to diverse and unseen domains encountered in real-world federated learning scenarios.

Although the presented approaches have made notable progress in domain generalization using meta-learning, there are still interesting avenues for future research. One such direction is Generalizable Label Distribution Learning (GLDL), which aims to learn a Label Distribution Learning (LDL) model that can generalize well to unseen target domains. LDL is a machine learning paradigm that assigns a label distribution to each instance, indicating the relative importance or description degree of each label. By reflecting the varying degrees of association between labels and instances, LDL provides a more detailed representation of label information. Consequently, leveraging the full distribution of labels can enhance model performance on complex tasks. A recent work by Zhao et al. [104] has expanded the scope of domain generalization research by addressing the specific challenges of LDL. In this work, the DICE framework was proposed, which learns to extract domain-invariant feature-label correlations and label-label correlations. It achieves this by aligning the prior distributions and label correlation matrices across different source domains. This work opens up new possibilities for applying meta-learning techniques to the GLDL problem, potentially enabling more robust and generalizable LDL models that can handle distribution shifts between source and target domains. Exploring the integration of meta-learning strategies with GLDL could lead to further advancements in this emerging area of research.

It is also important to note another key challenge that still needs to be addressed in this area of research: enabling algorithms to go beyond pattern recognition to understand and exploit data causality [74]. This understanding is essential for models to make consistent predictions across domains, particularly when existing correlations are unreliable [9, 57, 80]. Therefore, it is essential to provide meta-learning models with proper inductive biases that can guide the learning process toward causal inference. Inductive biases help in shaping the hypothesis space, enabling the model to prioritize learning generalizable patterns that reflect underlying causal relationships as opposed to spurious correlations. Despite challenges, meta-learning has valuable potential for DG if steered to grasp causal structures rather than merely mimic data. Developing meta-learning frameworks that inherently capture causal relationships will be pivotal for achieving true domain-agnostic capabilities.

I.8 Conclusions

Meta-learning has experienced rapid growth in interest in recent years and has been applied to many significant research topics in machine learning, including DG. In this paper, we focus on meta-learning for DG, a promising field that has attracted many researchers. We provide a comprehensive review of existing methods and present a detailed taxonomy based on two primary axes crucial for designing effective models, as well as an overview of relevant datasets, benchmarks, and applications. The taxonomy provided in this survey offers a roadmap for understanding the diverse strategies that underpin current research in the field. It is clear that both the feature extraction and classifier training processes are critical to the development of DG models. Furthermore, we share insights from our analysis of these methods and identify potential research challenges that could guide future research directions. We hope that this survey will assist newcomers and practitioners in navigating this growing field and will also highlight opportunities for future research.

Bibliography

[1] K. P. Ayodele, W. O. Ikezogwo, M. A. Komolafe, and P. Ogunbona. Supervised domain generalization for integration of disparate scalp eeg datasets for automatic epileptic seizure detection. *Computers in biology and medicine*, 120:103757, 2020.

[2] Y. Balaji, S. Sankaranarayanan, and R. Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31, 2018.

[3] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

[4] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3034–3042, 2016.

[5] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128, 2006.

[6] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. *Advances in neural information processing systems*, 29, 2016.

[7] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[8] P. Chattopadhyay, Y. Balaji, and J. Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 301–318. Springer, 2020.

[9] J. Chen, Z. Gao, X. Wu, and J. Luo. Meta-causal learning for single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7683–7692, 2023.

[10] K. Chen, D. Zhuang, and J. M. Chang. Discriminative adversarial domain generalization with meta-learning based cross-domain validation. *Neurocomputing*, 467:418–426, 2022.

[11] S. Choi, T. Kim, M. Jeong, H. Park, and C. Kim. Meta batch-instance normalization for generalizable person re-identification. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 3425–3435, 2021.

[12] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[13] G. Csurka. A comprehensive survey on domain adaptation for visual applications. *Domain adaptation in computer vision applications*, pages 1–35, 2017.

[14] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2(6):2, 2019.

[15] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[16] Q. Dou, D. Coelho de Castro, K. Kamnitsas, and B. Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in neural information processing systems*, 32, 2019.

[17] Y. Du, J. Wang, W. Feng, S. Pan, T. Qin, R. Xu, and C. Wang. Adarnn: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 402–411, 2021.

[18] Y. Du, J. Xu, H. Xiong, Q. Qiu, X. Zhen, C. G. Snoek, and L. Shao. Learning to learn with variational information bottleneck for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference,*

Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16, pages 200–216. Springer, 2020.

[19] Y. Du, X. Zhen, L. Shao, and C. G. Snoek. Metanorm: Learning to normalize few-shot batches across domains. In *International Conference on Learning Representations*, 2020.

[20] S. Erfani, M. Baktashmotlagh, M. Moshtaghi, X. Nguyen, C. Leckie, J. Bailey, and R. Kotagiri. Robust domain generalisation by enforcing distribution invariance. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 1455–1461. AAAI Press, 2016.

[21] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[22] C. Fang, Y. Xu, and D. N. Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1657–1664, 2013.

[23] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

[24] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE, 2004.

[25] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[26] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[27] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[28] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[29] A. Gholamzadeh Khoee, A. Javaheri, S. R. Kheradpisheh, and M. Ganjtabesh. Meta-learning in spiking neural networks with reward-modulated stdp. *Neurocomputing*, 600:128173, 2024.

[30] Z. Guan, Y. Li, Z. Pan, Y. Liu, and Z. Xue. Rfdg: Reinforcement federated domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[31] I. Gulrajani and D. Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[32] D.-K. Han and J.-H. Jeong. Domain generalization for session-independent brain-computer interface. In *2021 9th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–5. IEEE, 2021.

[33] H. He, S. Chen, K. Li, and X. Xu. Incremental learning from stream data. *IEEE Transactions on Neural Networks*, 22(12):1901–1914, 2011.

[34] J. He, R. Mao, Z. Shao, and F. Zhu. Incremental learning in online scenario. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13926–13935, 2020.

[35] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[36] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

[37] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[38] M. Huisman, J. N. Van Rijn, and A. Plaat. A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541, 2021.

[39] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[40] C. Jia and Y. Zhang. Meta-learning the invariant representation for domain generalization. *Machine Learning*, pages 1–21, 2022.

[41] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*, pages 158–171. Springer, 2012.

[42] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[43] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[45] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[46] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales. Sequential learning for domain generalization. In *European Conference on Computer Vision*, pages 603–619. Springer, 2020.

[47] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[48] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1446–1455, 2019.

[49] H. Li, S. J. Pan, S. Wang, and A. C. Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5400–5409, 2018.

[50] X. Li, W. Zhang, H. Ma, Z. Luo, and X. Li. Domain generalization in rotating machinery fault diagnostics using deep neural networks. *Neurocomputing*, 403:409–420, 2020.

[51] Y. Li, Y. Yang, W. Zhou, and T. Hospedales. Feature-critic networks for heterogeneous domain generalization. In *International Conference on Machine Learning*, pages 3915–3924. PMLR, 2019.

[52] Y. Liao, R. Huang, J. Li, Z. Chen, and W. Li. Deep semisupervised domain generalization network for rotary machinery fault diagnosis under variable speed. *IEEE Transactions on Instrumentation and Measurement*, 69(10):8064–8075, 2020.

[53] J. Liu, Z. Shen, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

[54] Q. Liu, Q. Dou, and P.-A. Heng. Shape-aware meta-learning for generalizing prostate mri segmentation to unseen domains. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23*, pages 475–485. Springer, 2020.

[55] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[56] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. *Advances in neural information processing systems*, 29, 2016.

[57] F. Lv, J. Liang, S. Li, B. Zang, C. H. Liu, Z. Wang, and D. Liu. Causality inspired representation learning for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8046–8056, 2022.

[58] D. Mahajan, S. Tople, and A. Sharma. Domain generalization using causal matching. In *International Conference on Machine Learning*, pages 7313–7324. PMLR, 2021.

[59] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[60] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *International conference on machine learning*, pages 10–18. PMLR, 2013.

[61] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1863–1868, 2006.

[62] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[63] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[64] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[65] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[66] V. Piratla, P. Netrapalli, and S. Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *International Conference on Machine Learning*, pages 7728–7738. PMLR, 2020.

[67] F. Qiao and X. Peng. Uncertainty-guided model generalization to unseen domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6790–6800, 2021.

[68] F. Qiao, L. Zhao, and X. Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020.

[69] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

[70] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

[71] L. Ren, T. Mo, and X. Cheng. Meta-learning based domain generalization framework for fault diagnosis with gradient aligning and semantic matching. *IEEE Transactions on Industrial Informatics*, 2023.

[72] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer, 2016.

[73] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

[74] D. Rothenhäusler, N. Meinshausen, P. Bühlmann, and J. Peters. Anchor regression: Heterogeneous data meet causality. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(2):215–246, 2021.

[75] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[76] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77:157–173, 2008.

[77] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[78] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[79] R. Shao, X. Lan, J. Li, and P. C. Yuen. Multi-adversarial discriminative deep domain generalization for face presentation attack detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10023–10031, 2019.

[80] P. Sheth, R. Moraffah, K. S. Candan, A. Raglin, and H. Liu. Domain generalization–a causal perspective. *arXiv preprint arXiv:2209.15177*, 2022.

[81] Y. Shu, Z. Cao, C. Wang, J. Wang, and M. Long. Open domain generalization with domain-augmented meta-learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9624–9633, 2021.

[82] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[83] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[84] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[85] M. Sugiyama and A. J. Storkey. Mixture regression for covariate shift. *Advances in neural information processing systems*, 19, 2006.

[86] N. Tripuraneni, M. Jordan, and C. Jin. On the theory of transfer learning: The importance of task diversity. *Advances in neural information processing systems*, 33:7852–7862, 2020.

[87] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[88] A. Vettoruzzo, M.-R. Bouguelia, J. Vanschoren, T. Rognvaldsson, and K. Santosh. Advances and challenges in meta-learning: A technical review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[89] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

[90] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31, 2018.

[91] B. Wang, M. Lapata, and I. Titov. Meta-learning for domain generalization in semantic parsing. *arXiv preprint arXiv:2010.11988*, 2020.

[92] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[93] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[94] Q. Wang, Y. Lv, Z. Xie, J. Huang, et al. A simple yet effective strategy to robustify the meta learning paradigm. *Advances in Neural Information Processing Systems*, 36, 2024.

[95] Y. Wang, H. Li, and A. C. Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3622–3626. IEEE, 2020.

[96] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer vision and image understanding*, 104(2-3):249–257, 2006.

[97] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382, 2019.

[98] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[99] Y. Yang and T. M. Hospedales. A unified perspective on multi-domain and multi-task learning. *arXiv preprint arXiv:1412.7489*, 2014.

[100] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[101] C. Zhang, L. Zhang, and J. Ye. Generalization bounds for domain adaptation. *Advances in neural information processing systems*, 25, 2012.

[102] G. Zhang, H. Zhao, Y. Yu, and P. Poupart. Quantifying and improving transferability in domain generalization. *Advances in Neural Information Processing Systems*, 34:10957–10970, 2021.

[103] M. Zhang, A. Zhang, and S. McDonagh. On the out-of-distribution generalization of probabilistic image modelling. *Advances in Neural Information Processing Systems*, 34:3811–3823, 2021.

[104] X. Zhao, L. Qi, Y. An, and X. Geng. Generalizable label distribution learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 8932–8941, 2023.

[105] Y. Zhao, Z. Zhong, F. Yang, Z. Luo, Y. Lin, S. Li, and N. Sebe. Learning to generalize unseen domains via memory-based multi-source meta-learning for person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6277–6286, 2021.

[106] H. Zheng, Y. Yang, J. Yin, Y. Li, R. Wang, and M. Xu. Deep domain generalization combining a priori diagnosis knowledge toward cross-domain fault diagnosis of rolling bearing. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2020.

[107] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[108] Z. Zheng, L. Zheng, and Y. Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *Proceedings of the IEEE international conference on computer vision*, pages 3754–3762, 2017.

[109] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[110] K. Zhou, Y. Yang, T. Hospedales, and T. Xiang. Learning to generate novel domains for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 561–578. Springer, 2020.

II

Latent Domain Prompt Learning for Vision-Language Models

Zhixing Li, Arsham Gholamzadeh Khoee, Yinan Yu

ICASSP 2026 – IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2026

Abstract

The objective of domain generalization (DG) is to enable models to be robust against domain shift. DG is crucial for deploying vision-language models (VLMs) in real-world applications, yet most existing methods rely on domain labels that may not be available and often ambiguous. We instead study the DG setting where models must generalize well without access to explicit domain labels. Our key idea is to represent an unseen target domain as a combination of latent domains automatically discovered from training data, enabling the model to adaptively transfer knowledge across domains. To realize this, we perform latent domain clustering on image features and fuse domain-specific text features based on the similarity between the input image and each latent domain. Experiments on four benchmarks show that this strategy yields consistent gains over VLM-based baselines and provides new insights into improving robustness under domain shift.

II.1 Introduction

Traditional machine learning methods often assume training and test data are drawn from the same distribution. However, in practical applications such as autonomous driving and intelligent robotics, models must operate in complex and unpredictable environments, where it is nearly impossible for training data to cover all possible scenarios. This discrepancy, known as domain shift, is the central concern of the Domain Generalization (DG) problem [16], which aims to leverage training data from one or multiple source domains so that the model performs well on unseen target domains.

With the rapid advancement of language models in natural language processing (NLP), vision models have been integrated with them, giving rise to vision-language models (VLMs) [15] such as CLIP [11]. Pre-trained on large-scale image–text pairs, VLMs exhibit strong zero-shot classification capabilities [11]. However, zero-shot classification requires carefully designed prompts, and manually crafting prompts is both time-consuming and sub-optimal. To address this, CoOp [52] introduced prompt learning, replacing hand-crafted prompts with learnable parameters (soft prompts) optimized on downstream tasks. While such methods surpass manual prompts in classification, soft prompts are prone to overfitting source domains [8], thus limiting generalization and yielding unsatisfactory DG performance. This motivates us to improve prompt learning for more robust generalization under domain shifts.

Many high-performing DG methods rely on domain labels during training [1, 13, 14]. Yet in practice, assigning precise domain labels to each image is difficult, and domain boundaries are often ambiguous. For example, in autonomous driving, images under slightly different weather or lighting conditions may not fit predefined categories such as “sunny” or “cloudy.” In such cases, algorithms depending on explicit domain labels often fail, limiting their applicability.

Inspired by [9], we propose **Latent Domain Prompt Fusion (LDPF)**, which automatically discovers latent domains and adapts prompts accordingly to enhance the DG ability of VLMs. Our method performs latent domain clustering on domain-specific style features, removing the need for domain labels and potentially capturing intrinsic characteristics more accurately than manual annotations. We design a dual-part soft prompt with domain-agnostic and domain-specific components, balancing invariant and specialized information. Finally, we introduce a domain-similarity–based fusion module that dynamically combines learned soft prompts during inference, improving visual–textual alignment.

Our main contributions are:

1. A novel soft prompt learning framework (LDPF) for domain generalization of VLMs, which does not rely on domain labels via latent domain clustering, making it suitable for complex real-world scenarios.

2. A novel fusion mechanism that represents the target domain as a linear combination of source latent domains, enabling adaptive prompt integration and more robust modality alignment.

3. Extensive evaluation on four benchmarks, with detailed analyses demonstrating the effectiveness of our approach.

II.2 Methodology

II.2.1 Problem setup

Specifically, we consider a Multi-Source Domain Generalization (MSDG) problem with L source domains $S_l = \{(x_i^l, y_i^l)\}_{i=1}^{n_l}$, each associated with a joint distribution P_{XY}^l. Note that $P_{XY}^l \neq P_{XY}^{l'}, \forall l, l' \in \{1, \cdots, L\}$ and $l \neq l'$. The goal of the MSDG problem is to learn a predictive function $f : \mathcal{X} \to \mathcal{Y}$ using source domain data such that it minimizes the prediction error on an unseen target domain $S_{target}, P_{XY}^{target} \neq P_{XY}^l, \forall l \in \{1, \cdots, L\}$ [16]:

$$\min_f \mathbb{E}[\mathcal{L}(f(x^{target}), y^{target})], \tag{II.1}$$

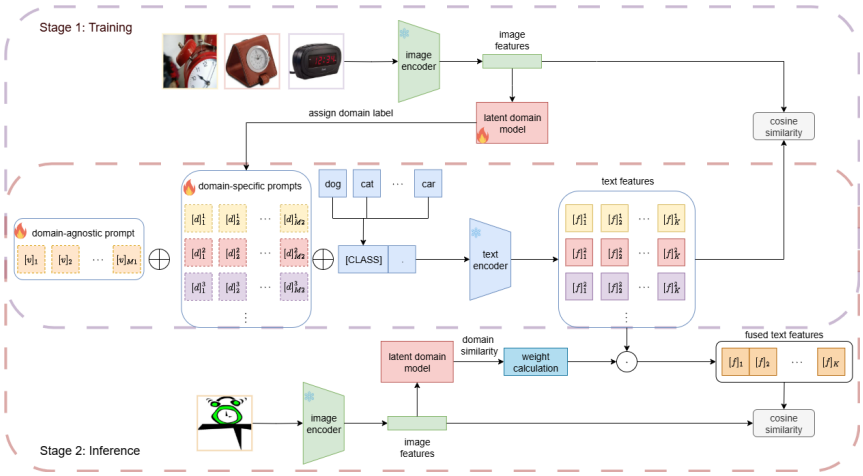
where $\mathcal{L}(\cdot, \cdot)$ is the loss function. For simplicity, we assume that the source and target domains share the same label set.

II.2.2 Soft prompt design

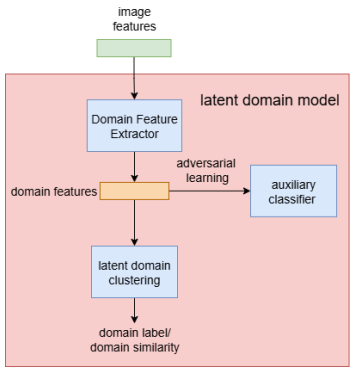
In our settings, each prompt is divided into three parts: domain-agnostic, domain-specific, and class label:

$$\mathbf{p}_k = \underbrace{[v]_1[v]_2 \cdots [v]_{M1}}_{\text{domain-agnostic tokens}} \underbrace{[d]_1[d]_2 \cdots [d]_{M2}}_{\text{domain-specific tokens}} [\text{CLASS}]_k. \tag{II.2}$$

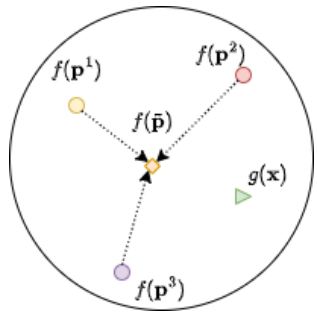
Here, $[v]$ represents the domain-agnostic tokens, $[d]$ is the domain-specific tokens, and [CLASS] represents the class label. M_1 and M_2 are hyperparameters that control the lengths of the two types of prompts, and k represents the k-th class.



(a) Latent Domain Prompt Fusion (LDPF) framework.



(b) Latent domain model.



(c) Modality alignment.

Figure II.1: Overview of the proposed framework. During training, the image and text encoders are frozen, and the Latent Domain Model assigns latent domain labels to guide the learning of domain-specific prompts. At inference, the model estimates the similarity between the input image and each latent domain, and fuses text features accordingly for better modality alignment, i.e., increasing the cosine similarity between image feature and the corresponding positive text feature.

All domains share a single domain-agnostic prompt, while an independent domain-specific prompt is trained for each latent domain. The training is performed in two stages. In the first stage, only the domain-specific prompts are used, and the model minimizes the classification loss $\mathcal{L}_{dsp}$ within

each latent domain:

$$\mathcal{L}_{dsp} = -\frac{1}{N}\sum_{s=1}^{N_s}\sum_{(x,y)\in\mathcal{D}_s}\sum_{k=1}^{K} y_k \log P(\hat{y}=k|\mathbf{x},\mathbf{d^s}), \tag{II.3}$$

Here, N denotes the total number of samples, N_s is the number of latent domains, $\mathcal{D}_s$ represents the set of all samples assigned to domain s, and $\mathbf{d^s}$ denotes the domain-specific prompt corresponding to domain s.

In the second stage, the domain-agnostic prompt is concatenated with the domain-specific prompts, and the model minimizes the classification loss $\mathcal{L}_{dap}$ across all domains:

$$\mathcal{L}_{dap} = -\frac{1}{N}\sum_{(x,y)\in\mathcal{D}}\sum_{k=1}^{K} y_k \log P(\hat{y}=k|\mathbf{x},\mathbf{p}), \tag{II.4}$$

where $\mathcal{D}$ represents the set of all samples. During this step, only the domain-agnostic prompt is updated, ensuring that it serves as a complementary component that captures domain-invariant knowledge, thereby enhancing generalization.

II.2.3 Latent domain model

To extract domain features from the outputs of the image encoder, we design a two-layer MLP network, referred to as the Domain Feature Extractor. To ensure that the extracted domain features are as independent of class information as possible, we employ an auxiliary classifier trained in an adversarial manner against the extractor:

$$\mathcal{L}_{adv} = -\frac{1}{N}\sum_{(x,y)\in\mathcal{D}}\sum_{k=1}^{K} y_k \log P(\hat{y}=k|h(f(\mathbf{x}))), \tag{II.5}$$

where $h(\cdot)$ represents the auxiliary classifier, and $f(\cdot)$ denotes the image encoder. The resulting domain features are then clustered using k-means, which assigns each sample a latent domain label. We set the number of clusters to $k = N_s$, a hyperparameter tuned via the validation set, strictly independent of ground-truth domain labels. The cluster centroids are stored and later used during inference to compute similarities between input images and latent domains.

II.2.4 Prompt fusion mechanism

This mechanism generates an adaptive text feature that aligns with the target image despite domain shift. We represent the text features from different domains as a set of basis vectors and obtain the fused feature for class i via a weighted linear combination:

$$\tilde{\mathbf{f}}_k = \sum_{s=1}^{N_s} \alpha_s \mathbf{f}_k^s, \tag{II.6}$$

where $\mathbf{f}_k^s$ is the text feature of class k in domain s, and α_s is its fusion weight. The weights are derived from the cosine similarity between the domain feature and each latent domain centroids $\mathbf{c}_s$:

$$\alpha_s = \frac{\exp(\cos(h(f(\mathbf{x}_i)), \mathbf{c}_s)/\tau)}{\sum_{j=1}^{N_s} \exp(\cos(h(f(\mathbf{x}_i)), \mathbf{c}_j)/\tau)}. \tag{II.7}$$

II.2.5 Training and inference

During training, the backbone encoders are kept frozen, and only the parameters of the soft prompts and the Latent Domain Model are updated. We adopt a Gradient Reversal Layer (GRL) [6] to implement adversarial learning. The Kuhn-Munkres algorithm [10] is used to stabilize cluster–label assignment. Total loss function is as follows:

$$\mathcal{L} = \mathcal{L}_{dsp} + \lambda(\mathcal{L}_{dap} - \mathcal{L}_{adv}), \tag{II.8}$$

where $\lambda = \frac{2}{1+\exp(-10p)} - 1$ [6]. At inference, the Latent Domain Model estimates domain similarities, which guide the adaptive fusion of text features. The classification probabilities that the input image $\mathbf{x}_i$ belongs to class k can then be computed using the following formula:

$$P(y_i = k|\mathbf{x}_i, \tilde{\mathbf{f}}) = \frac{\exp(\cos(\tilde{\mathbf{f}}_k, f(\mathbf{x}_i))/\tau)}{\sum_{j=1}^{K} \exp(\cos(\tilde{\mathbf{f}}_j, f(\mathbf{x}_i))/\tau)}. \tag{II.9}$$

II.3 Experiments

II.3.1 Datasets and implementation details

We conduct experiments on four benchmark datasets: Office-Home [39], mini-DomainNet [19], PACS [27], and Terra Incognita [1]. Our method builds

on CLIP with a ViT-B/16 [5] backbone, where the encoders are kept frozen during training. We set the domain-agnostic prompt length to 4, the domain-specific prompt length to 8, the number of latent domain to 3, while following the same hyperparameter settings as CoOp. Trained for 20 epochs on Terra Incognita, and for 30 epochs on the other datasets. For all datasets, we followed the leave-one-domain-out evaluation protocol as in [19]. Each experiment was conducted three times, and we report the average performance along with the standard deviation.

II.3.2 Main results

For comparison, we adopt a series of VLM-based baselines. Besides zero-shot CLIP, we include several prompt-learning approaches, as CoOp, CoCoOp [51], BPL [4], StyLIP [3], and DDSPL [13], as well as an adapter-based method CLIPCEIL [14]. It is worth noting that both DDSPL and CLIPCEIL require access to domain labels during training, and we add them as reference.

Table II.1: Performance comparison of different methods. We reproduce the results of Zero-shot CLIP and CoOp, while * indicates that the result is reported in [13], † indicates that the result is reported in [14]. All methods employed the ViT-B/16 backbone. The best result is highlighted in **bold** and the second is underlined.

Category	Method	Office-Home	mini-DomainNet	PACS	Terra Inc	Average
Without domain label	Zero-shot CLIP	82.03	84.10	96.10	33.95	74.05
	CoOp [52]	83.52	84.42	96.32	47.52	77.92
	CoCoOp [51]	80.70*	85.81*	96.73*	**50.40**†	78.41
	BPL* [4]	84.02	85.28	96.86	-	-
	StyLIP* [3]	84.63	-	**98.05**	-	-
	LDPF (Ours)	**85.13**±0.32	**85.82**±0.17	96.81±0.15	47.65±0.96	78.85
With domain label	CLIPCEIL† [14]	85.43	-	97.55	52.98	-
	DDSPL* [13]	85.59	86.23	98.08	-	-

The results are shown in Table II.1. From the results, our method consistently outperforms strong baselines such as Zero-shot CLIP, CoOp, and CoCoOp. In particular, on Office-Home and mini-DomainNet, it achieves the second-best average accuracy, only behind DDSPL, while surpassing all other methods that do not rely on domain labels. On PACS and Terra Incognita, the performance is comparatively weaker, suggesting room for further improvement. We analyzed the possible reasons in Section II.3.4. Overall, our approach yields an average gain of +4.8% over Zero-shot CLIP, indicating that it provides a robust generalization boost to CLIP despite not relying on domain labels.

II.3.3 Ablation study

Table II.2: Ablation study results on Office-Home dataset.

Model	Average
Remove DAP	84.91
Remove DSP	84.30
Remove L_{adv}	84.80
Remove latent domain clustering	84.53
Greedy Fusion	84.52
Average Fusion	85.05
LDPF (Ours)	**85.13**

Table II.2 reports the results of our ablation study. Removing any single component leads to performance degradation, confirming their effectiveness. The first two tests highlight the critical role of domain-specific prompts in our framework. Interestingly, replacing latent domain clustering with manually annotated domain labels results in worse performance, suggesting that human-defined domains may not fully capture image-specific styles and can even mislead the model. Experiments on the fusion mechanism further show its ability to leverage inter-domain similarities for fusion, though the gain over simple averaging remains modest, indicating potential for more advanced fusion strategies in future work.

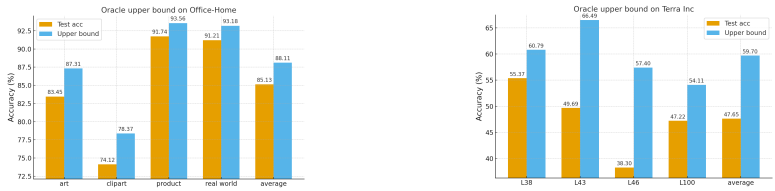
II.3.4 Upper bound analysis

We train one soft prompt per latent domain and fuse their text features at inference. As a tractable oracle, we consider **selection** instead of fusion: for each sample, an oracle chooses the most accurate single prompt. This yields the bound

$$U_{\text{sel}} = P\Big(\exists\, s :\ \hat{y}_s(\mathbf{x}) = y\Big), \tag{II.10}$$

i.e., the probability that at least one domain-specific prompt predicts correctly. Note that selection is a special case of fusion (a degenerate convex combination with a one-hot weight), and fusion can also create new representations by combining prompts; therefore the true oracle for fusion satisfies $U_{\text{fuse}} \geq U_{\text{sel}}$. In our analysis, we report U_{sel} as a conservative, easy-to-compute upper bound.

We illustrate the results on the two most representative datasets, Office-Home and Terra Incognita, in Fig. II.2. The other two datasets show trends similar to Office-Home. As observed, on Office-Home the gap between our



(a) Office-Home (b) Terra Incognita

Figure II.2: Oracle upper bound U_{sel} on two datasets.

method and U_{sel} is only 2.98%, indicating that the proposed fusion effectively integrates information from different soft prompts to yield accurate predictions. In contrast, on Terra Incognita the gap widens to 12.05%. This suggests that individual soft prompt tend to be highly specialized: one may handle certain conditions (e.g., infrared/night images) well while others fail badly. A simple similarity-based fusion averages over such inconsistent predictions, often suppressing the correct soft prompt instead of amplifying it. Therefore, Terra Incognita reveals that soft prompt complementarity is strong but cannot be exploited by naive fusion, highlighting the need for more adaptive fusion/selection strategies.

II.4 Conclusion

In this work, we studied a prompt learning–based domain generalization method for VLMs without relying on domain labels. Our framework represents an unseen target domain as a linear combination of source latent domains, allowing adaptive transfer of knowledge. Experiments on four benchmarks confirm the effectiveness of this formulation, while our upper bound analysis highlights both the promise and the current limitations of simple fusion. Future work will explore learning-based fusion or gating strategies to better exploit the complementarity among prompts.

Bibliography

[1] S. Bai, Y. Zhang, W. Zhou, Z. Luan, and B. Chen. Soft prompt generation for domain generalization. In *European Conference on Computer Vision*, pages 434–450. Springer, 2024.

[2] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.

[3] S. Bose, A. Jha, E. Fini, M. Singha, E. Ricci, and B. Banerjee. Stylip: Multi-scale style-conditioned prompt learning for clip-based domain generalization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5542–5552, 2024.

[4] M. M. Derakhshani, E. Sanchez, A. Bulat, V. G. T. da Costa, C. G. Snoek, G. Tzimiropoulos, and B. Martinez. Bayesian prompt learning for image-language model generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15237–15246, 2023.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[6] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[7] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[8] C. Ma, Y. Liu, J. Deng, L. Xie, W. Dong, and C. Xu. Understanding and mitigating overfitting in prompt tuning for vision-language mod-

els. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(9):4616–4629, 2023.

[9] T. Matsuura and T. Harada. Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11749–11756, 2020.

[10] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[12] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017.

[13] F. Xu, S. Deng, T. Jia, X. Yu, and D. Chen. Ensembling disentangled domain-specific prompts for domain generalization. *Knowledge-Based Systems*, 301:112358, 2024.

[14] X. Yu, S. Yoo, and Y. Lin. Clipceil: Domain generalization through clip via channel refinement and image-text alignment. *Advances in Neural Information Processing Systems*, 37:4267–4294, 2024.

[15] J. Zhang, J. Huang, S. Jin, and S. Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5625–5644, 2024.

[16] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2023.

[17] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825, 2022.

[18] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[19] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021.

III

GoNoGo: An Efficient LLM-based Multi-Agent System for Streamlining Automotive Software Release Decision-Making

Arsham Gholamzadeh Khoee, Yinan Yu, Robert Feldt, Andris Freimanis, Patrick Andersson Rhodin, Dhasarathy Parthasarathy

IFIP International Conference on Testing Software and Systems, Springer Nature Switzerland, 2024

Abstract

Traditional methods for making software deployment decisions in the automotive industry typically rely on manual analysis of tabular software test data. These methods often lead to higher costs and delays in the software release cycle due to their labor-intensive nature. Large Language Models (LLMs) present a promising solution to these challenges. However, their application generally demands multiple rounds of human-driven prompt engineering, which limits their practical deployment, particularly for industrial end-users who need reliable and efficient results. In this paper, we propose GoNoGo, an LLM agent system designed to streamline automotive software deployment while meeting both functional requirements and practical industrial constraints. Unlike previous systems, GoNoGo is specifically tailored to address domain-specific and risk-sensitive systems. We evaluate GoNoGo's performance across different task difficulties using zero-shot and few-shot examples taken from industrial practice. Our results show that GoNoGo achieves a 100% success rate for tasks up to Level 2 difficulty with 3-shot examples, and maintains high performance even for more complex tasks. We find that GoNoGo effectively automates decision-making for simpler tasks, significantly reducing the need for manual intervention. In summary, GoNoGo represents an efficient and user-friendly LLM-based solution currently employed in our industrial partner's company to assist with software release decision-making, supporting more informed and timely decisions in the release process for risk-sensitive vehicle systems.

III.1 Introduction

In the automotive industry, decisions about when to release software, particularly embedded software in risk-sensitive systems, carry immense weight. The complexity of modern vehicles, with their multiple levels of integration, further complicates this process. Each integration level involves one or more gating steps, with tests conducted to verify whether gate criteria are fulfilled. Gate failures can delay the integration of all dependent subsystems, regardless of their individual quality. In this intricate process, release managers, bearing the responsibility of gatekeeping could greatly benefit from assistance to make faster and better decisions.

Large language models (LLMs) present an interesting avenue for providing such assistance. In particular, LLMs have demonstrated strong capabilities in zero- and few-shot settings with in-context learning [4]. Recent advancements have improved reasoning [48], exemplar selection, and prompt design [4]. Companies now use LLMs for software engineering tasks like API testing, code generation, and documentation and research studies have already shown test automation improvements over the state-of-the-art [28].

However, when applying LLMs to *risk-sensitive domain-specific* tasks, several unique challenges must be addressed. In our research with industrial partners, the most prominent challenges include 1) Incorporating specific logic and terminology relevant to the domain; 2) Understanding and parsing high-level queries or vague language used by non-expert stakeholders and translating them into actionable plans, 3) Enabling interpretability so that domain experts can explain system functionality to stakeholders without excessive complexity, 4) Operating efficiently to meet the time-critical demands of organizational applications, mitigating potential bottlenecks related to limited LLM licensing or infrastructure, and 5) Designing the system to enable ease of troubleshooting and maintenance, ensuring that any issues can be quickly identified and resolved.

To address these challenges, we propose a multi-agent system that encodes domain-specific requirements using in-context learning. This system comprises two primary LLM agents: a Planner and an Actor (refer to Figure III.2). The Planner, which forms the core of the system, comprehends and decomposes user queries into step-by-step instructions for data analysis [18]. The Actor then synthesizes and generates executable scripts from these higher-level instructions. Within the Actor, a coder LLM utilizes the self-reflection mechanism besides a memory to produce the most effective Python script optimized for querying the given data for each instruction generated by the Planner [2, 28].

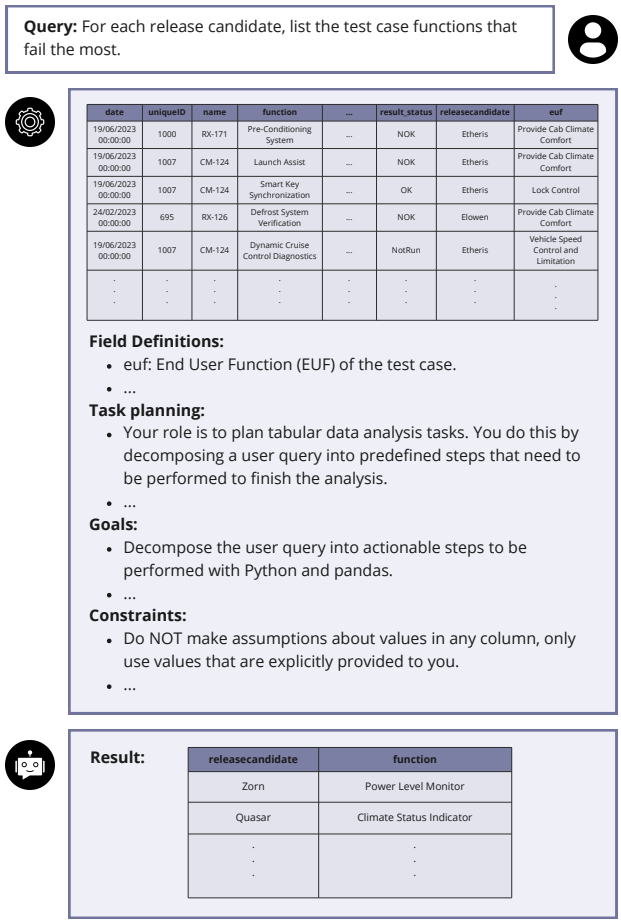


Figure III.1: An actual example demonstrating the use of the LLM-based multi-agent system for automating ad-hoc tabular data analysis.

This system provides an interface for end-users at our industrial partner, illustrated by Figure III.1, which shows a real-world example of its use. It allows end-users, like release managers, to interpret results from a business and safety perspective without needing detailed technical knowledge. For example, they can simply receive a short table that reports the test case functions that have failed the most for each release candidate as shown in Figure III.1. By reviewing this information, release managers can make well-informed decisions on whether to release the software or not, ensuring that it meets both business objectives and safety standards in the automotive industry. This approach can significantly reduce time and resources by elimi-

nating the need for various database and programming experts to achieve the desired results for end-users. Our agent automates test data analysis across multiple vehicle development integration levels, providing detailed reports on component functionality and system interactions. This assists release managers in making informed decisions about software readiness for release, accelerating development while enhancing gatekeeping reliability. Our contributions can be summarized as follows:

- We highlight the practicality of the proposed LLM-based intelligent assistant in making software release decisions within the automotive industry. This is achieved by enhancing two key capabilities:
  - **Domain-specificity:** We design a framework to handle unstructured queries from non-expert stakeholders in the automotive industry by mapping generic language to domain-specific logic using in-context learning.
  - **Risk-sensitivity:** We incorporate two predefined atomic operations to restrict the action space and improve the risk-sensitive aspect of the planner.
- Experiments on a total of 50 crafted test queries show that our proposed system is effective at analyzing data and deriving the required insights for software release decision-making.
- Our system, now deployed and actively used within our industrial partner's company, has demonstrated significant improvements in the software release decision-making process besides saving time, improving accessibility, reducing reliance on specialized analysts, and accelerating overall workflow.

The remainder of this paper is structured as follows: Section III.2 provides an overview of the manual process behind automotive software release decisions and the need for streamlining operations. Section III.3 details our approach, including a description of the architecture of our LLM-based multi-agent system and an explanation of the Planner and Actor agents. Furthermore, in Section III.4, we present our experimental setup and results. Section III.6 provides an overview of similar research in LLMs for data analysis. Finally, Section III.7 concludes the paper by summarizing key findings and discussing the broader implications of our work in the context of industrial software release management and risk-sensitive systems.

III.2 Manual Process of Release Decisions: Insights From the Industry

Deciding to go ahead, or not, with a software release in the automotive industry is a complex task involving multiple stakeholders and extensive data analysis. This section reviews the current, and typical of the industry at large, manual workflow and the need for streamlining.

Vehicle development progresses through multiple phases, each becoming more complex as more components are integrated. Numerous tests are conducted at each stage to ensure functionality and identify revisions, generating vast amounts of data. Software components require repeated testing and validation, adding to this data.

Project managers, verification engineers, and quality engineers need clear analytics and insights from these tests in order to make software release decisions. Extracting essential information is time-consuming. Quality engineers analyze data for continuous improvement, while release engineers need specific information to make informed release decisions.

Within this process, statisticians provide an overall view of the data to project managers and quality engineers for future business decisions. Manual data processing is necessary due to the critical nature of these decisions and their impact on consumer safety. However, this approach is time-consuming and prone to errors, partly due to the differing perspectives of technical data analyzers and statisticians, who may not fully understand the project managers' goals.

A critical and typical stage in this process is "Testing on Closed Track," where vehicles equipped with the necessary software release undergo systematic and rigorous testing of their systems in a controlled environment. After these tests, release managers analyze large amounts of data to decide whether to move to the next test stage. This involves manually querying data to generate reports that support informed decisions. Errors or delays in this analysis can hinder timely software release, affect business goals, and delay subsystem integration.

The deployment of an intelligent assistant has the potential to facilitate software release decisions in the automotive industry [41], particularly during the critical testing on a closed track phase. In this work, we have focused on designing such an LLM-based multi-agent system to address the challenges of this specific stage. By rapidly processing test data from closed track testing, the system can generate comprehensive reports tailored to different stakeholders' needs. For example, it can quickly compile summaries of failed tests, highlight software performance trends across vehicle models,

or analyze a specific component's behavior under various conditions. Consequently, this reduces the time spent on initial analysis, allowing release managers to focus on interpreting results and making informed decisions. This not only accelerates the development process but also enhances the accuracy and reliability of the information used in release decisions, ultimately contributing to maintaining high safety and quality standards in automotive software development.

III.3 GoNoGo: Intelligent Software Release Assistant

III.3.1 System Requirements

After discussing current needs and opinions about the software release analysis and decision-making processes with our industrial partner, we identified the following main challenges in automating data analysis:

Understanding User Queries The system must interpret queries, typically presented in natural language [16], within the specific domain context, using any provided domain-specific knowledge.

Translating User Queries to Actionable Steps The system needs to convert the user's query into concrete steps, breaking down complex queries into simpler tasks, determining the order of operations, and selecting appropriate data manipulation or analysis techniques. Additionally, the action space must be carefully managed to adhere to risk-sensitive requirements.

Execution and Result Preparation The system must execute the planned actions, interact with data using scripts (e.g., querying databases, performing calculations, applying filters), and compile the results into the desired format for the user.

These steps rely heavily on the LLM's domain-specific knowledge and reasoning ability, crucial for effective query instruction planning [38]. Consequently, this work explores techniques for enhancing the reasoning capabilities of LLM agent systems, particularly for the analysis of tabular data in industrial contexts.

III.3.2 System Architecture

Our approach to automating tabular data analysis leverages LLMs to create an intelligent system capable of interpreting natural language queries,

executing complex analyses, and delivering desired results. The system architecture consists of two main components: the Planner supported by a Knowledge Base and Examples for few-shot learning and the Actor including coder LLM, memory module, and some Plugin components. Figure III.2 illustrates the overall architecture of the developed system.

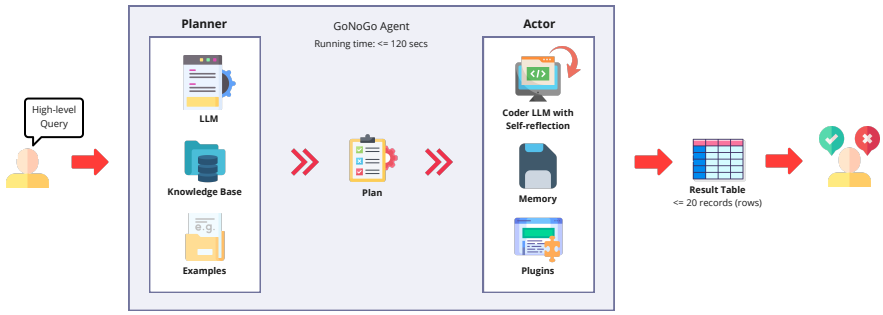


Figure III.2: Architecture of the LLM-based multi-agent system GoNoGo along with the illustration of the interaction procedure of the system. GoNoGo receives high-level queries from the end user, performs the required data manipulations, and outputs the result table as a decision support resource. GoNoGo comprises a Planner agent, which interprets queries and devises analysis strategies using Chain-of-Thought prompting and self-consistency, supported by a *Knowledge Base* and *Examples* for few-shot learning. The Actor includes a Coder LLM with a *Self-reflection* mechanism, utilizing *Memory* and *Plugins* for code generation and error resolution. The total running time of GoNoGo for one user query is approximately 120 seconds, which satisfies typical user requirements.

III.3.2.1 Planner

The Planner is the core of our system, responsible for interpreting user queries and devising appropriate analysis strategies. One of the core challenges of designing an LLM-based multi-agent system is the inherent inaccuracy of prompting. As decision-making becomes more distributed over multiple LLM agents, the uncertainty within the multi-agent system increases. To mitigate this, we centralize the complexity within the Planner, which is responsible for the majority of design choices. By focusing on the Planner as the main agent for refinement, we aim to create a system that is both interpretable and easily maintainable.

Our problem consists of two main aspects: domain-specificity and risk-sensitivity. These two characteristics frequently manifest together in real-world applications, particularly in fields such as healthcare and automotive, where unreliability and inaccuracies can have significant consequences. However, there is a noticeable gap in addressing both aspects simultaneously, let alone demonstrating such systems in practice. As part of our system, we want to explicitly address both of these aspects. As the Planner is the component with the most decision-making responsibility, these two requirements are encoded into the Planner prompts as depicted in Figure III.3.

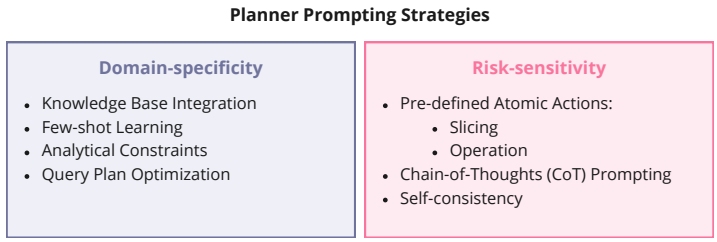


Figure III.3: Planner prompting strategies addressing domain-specificity and risk-sensitivity in the LLM-based agent system for tabular data analysis.

III.3.2.2 Domain-specificity

The Planner utilizes a Knowledge Base containing a structured description of the data and its attributes to provide the necessary context and domain-specific information in the prompts given to the LLM, enhancing the system's performance and applicability [14]. The Knowledge Base serves as a comprehensive repository of metadata, including detailed descriptions of data tables, possible states and values for essential fields, and domain-specific terminologies, as well as the semantic meanings of various data elements. This enables the system to understand and interpret high-level queries and devise appropriate analysis plans by using this information as input prompts for the Planner, taking advantage of in-context learning. This integration ensures that the entire pipeline, from query interpretation to result generation, is informed by relevant domain knowledge, enabling the LLM agent to provide more accurate, relevant, and specialized responses to user queries.

In our system architecture, we also feed some input-output pairs as examples into the Planner, allowing few-shot learning alongside the Knowledge Base. This combination enables the Planner to interpret user queries more effectively, drawing on both general knowledge and specific task examples

to formulate appropriate analysis plans. This approach makes the system a powerful tool for automated tabular data analysis across various industries and use cases.

Helpful prompts serve as constraints, enhancing the LLM's reasoning capabilities [12]. For example, constraints help the model understand that queries should account for more than just binary states for some fields. Retrieving records with 'A' and its opposite doesn't always mean retrieving all records, as other non-binary states might exist. For instance, 'successful' and 'failed' tests don't encompass all possible test statuses; there may be additional statuses to consider, such as 'N/A', that the model should take into account.

The focus is on pushing the model to generate an optimized query plan. This involves narrowing down data through filtering and selection before performing sorting and other operations on the reduced dataset to minimize processing. Accordingly, designed constraints help the agent explore the characteristics of each field and the data, providing more accurate planning.

III.3.2.3 Risk-sensitivity

We guide the Planner with two pre-defined atomic actions to limit the action space of the planner: slicing and operation. Slicing involves specifying the columns to select and the conditions for filtering rows from the data to be analyzed. Operation involves describing the operations (such as max, mean, count, etc.) to be performed on the values of one or more columns of the data obtained from the slicing step. The steps should be returned as a Python list, with each step described in natural language, including all relevant values and column names.

Also, we leverage Chain-of-Thought (CoT) prompting to further enhance the reasoning capabilities of our LLM-based agent [43]. This technique incorporates intermediate reasoning steps into the prompt, guiding the model to break down complex problems into smaller, more manageable steps [33]. This approach mimics human-like reasoning and problem-solving processes. Additionally, CoT prompting makes the agent's decision-making process more transparent by explicitly showing the reasoning steps, allowing users to understand how the agent arrived at a particular conclusion or analysis result.

We combine CoT prompting with few-shot learning by providing examples that not only show input-output pairs but also include the intermediate reasoning steps. This synergy further enhances the agent's ability to handle diverse and complex data analysis tasks [9].

To further improve reasoning, we employ self-consistency in conjunction with CoT prompting. This involves generating multiple independent reasoning paths for the same query, comparing them for consistency, and using majority voting to determine the most reliable outcome [42]. By considering multiple reasoning paths, the system becomes less likely to be misled by a single flawed chain of thought. As a result, for queries with potential ambiguity, self-consistency can help identify different valid interpretations and provide a more comprehensive answer.

III.3.2.4 Actor

The Actor is responsible for carrying out the analysis plans devised by the Planner. It consists of several interacting components: Coder LLM with Self-reflection, Memory, and Plugins, as depicted in Figure III.2.

The Coder LLM is responsible for generating executable scripts based on the Planner's instructions. This component is crucial as it translates abstract plans into concrete, executable code that can interact with the data using the required Plugins and perform the necessary analysis. It includes a Self-reflection mechanism, which works in tandem with a Memory module. This Memory stores generated code, error messages, execution results, and contextual information about the current task [7].

The Self-reflection mechanism is a sophisticated process that allows the Coder LLM to critically analyze its own output and decision-making process. When an error occurs during script execution, the Self-reflection mechanism activates, providing feedback to the Coder LLM. This feedback loop enables the LLM to analyze error messages within the task context [10, 28], facilitating iterative improvement of the generated code.

The Self-reflection mechanism offers several advantages: It enables the Coder LLM to autonomously identify and correct errors by continuously analyzing and reflecting on its own output, thereby reducing the need for external debugging and intervention. This mechanism promotes a cycle of continuous improvement, allowing each iteration to refine the scripts for progressively better performance and reliability [25]. By utilizing the Memory module, the Coder LLM can make context-aware adjustments, considering previous errors, execution results, and specific task requirements, which leads to more precise and contextually appropriate code generation. Automated error correction and iterative refinement result in a more efficient coding process, speeding up the development cycle and enhancing the robustness and reliability of the final scripts. Additionally, the self-reflective capabilities minimize the need for human intervention in the debugging process, enabling engineers to focus on more complex and high-level tasks.

This architecture enables the Actor to not only generate code for data analysis tasks but also to troubleshoot and improve its own output, resulting in a more robust and reliable automated data analysis system.

III.3.3 System Implementation

The system uses Azure OpenAI's GPT-3.5 Turbo for both the Planner and Actor agents. The Planner utilizes specially designed prompts for task planning, defining the entire data analysis task by specifying the details of each step in the plan. Moreover, the Actor uses predefined prompts to generate the required Python code for executing each step of the provided plan with the pandas library, performing tasks on the given data.

III.4 Experiments

III.4.1 Data

The data used for analysis at our industrial partner is called "GoNoGo" data and is updated after testing each function of every software component in each vehicle. This internal company data contains about 40 different fields and is critical for release decisions, as it includes detailed information regarding the performance and functionality of software components. It provides the necessary information for determining whether to advance a vehicle to the next phase of development and allow it to be driven on open roads. Although the data is updated after each test, we used a dataset of 55,000 records to report our experiments.

Stakeholders often ask questions like "What are the test case functions that fail the most for release candidate X?" or "What is the Y-status of X?" where X is the release candidate's name and Y is a specific functionality. Answering these questions requires domain knowledge and an understanding of the data to extract and communicate the answers accurately. By analyzing this data, release managers can determine if a vehicle meets the necessary criteria to progress to the next development phase or be driven on public roads. This ensures that only vehicles that meet stringent safety and quality standards are advanced, maintaining high standards in automotive software development.

III.4.2 Benchmark Overview

To evaluate the GoNoGo system's performance, we developed a benchmark based on 15 initial analysis tasks. These tasks were defined with the help

of release engineers, quality engineers, and verification engineers. We identified the most common high-level queries and criteria frequently used by these end users in their workflows. Our goal was to design tasks that capture the nuances and complexity of the demanding queries necessary for their decision-making processes. These tasks were then translated into explicit table analysis queries that GoNoGo could process, ensuring that the benchmark reflects real-world scenarios and challenges typically encountered by these professionals. We created definitive ground-truth solutions for these queries using Python, breaking down the solutions into smaller code chunks representing operations such as filtering, grouping, and sorting. For each query, we generated a series of query ablations by incrementally adding code chunks and formulating corresponding queries that these chunks would solve. This method expanded our original 15 queries into 50 query ablations, each with a corresponding ground-truth solution and Python code.

In this way, we established queries with four levels of difficulty:

Level 1 These are the simplest queries, typically involving a single operation such as filtering or sorting.

Level 2 These queries combine two or three basic operations, such as multiple filtering followed by sorting.

Level 3 These queries involve more than three operations, potentially including grouping and aggregating.

Level 4 These are the most complex queries, requiring multiple advanced operations such as grouping and aggregating, for calculating statistics, beyond basic filtering and sorting.

This incremental approach to query complexity allows us to assess GoNoGO's performance at various levels of difficulty. It helps identify at which point, if any, the system's performance begins to degrade, and provides insights into its capabilities in handling increasingly complex table analysis tasks.

This benchmark allows for objective evaluation of the GoNoGo's ability to handle increasingly complex table analysis tasks, ensuring a comprehensive assessment of its performance across a spectrum of difficulty levels.

III.4.3 Evaluation

The evaluation process involves comparing the GoNoGo system's results against manually generated ground-truth results. The comparison is based on a strict matching criterion [8]. For a match to be considered successful,

Table III.1: Performance evaluation of the GoNoGo system with varying numbers of example queries across different levels of task difficulty.

# Examples	Task Difficulty	# Total Tasks	# Success	# Failed	Performance
0-shot	1	16	3	13	18.75%
	1-2	32	6	26	18.75%
	1-3	44	9	35	20.45%
	1-4	50	11	39	22%
1-shot	1	16	15	1	93.75%
	1-2	32	27	5	84.37%
	1-3	44	32	12	72.72%
	1-4	50	34	16	68%
2-shot	**1**	**16**	**16**	**0**	**100%**
	1-2	32	31	1	96.87%
	1-3	44	38	6	86.36%
	1-4	50	41	9	82%
3-shot	**1**	**16**	**16**	**0**	**100%**
	1-2	**32**	**32**	**0**	**100%**
	1-3	**44**	**41**	**3**	**93%**
	1-4	**50**	**45**	**5**	**90%**

the system's output must contain the same columns as the ground truth. Additionally, each record in the system's output must exactly match a corresponding record in the ground truth, including all values across different fields. The system's output must also contain the same number of records as the ground truth, with no missing or extra entries. This strict matching ensures that the output is not just similar, but identical in structure and content to the expected result. If the agent's output satisfies all these criteria when compared to the ground truth, the task is marked as successful; otherwise, it is considered a failure. The model's performance is then quantified by calculating the success rate, defined as the ratio of successful tasks to the total number of tasks.

III.4.4 Results

We present our experiment results on the GoNoGo system in Table III.1. We evaluated its performance across different levels of task difficulty using 0-shot, 1-shot, 2-shot, and 3-shot examples. GoNoGo achieved high performance with 3-shot examples.

Initially, we assessed GoNoGo's ability to handle the simplest queries involving basic operations like filtering or sorting (Level 1). We then incrementally increased the complexity by including queries that combined Level 1 and Level 2 difficulties, followed by those incorporating Level 1 to Level 3 difficulties. Finally, we evaluated GoNoGo's performance on the full spectrum of tasks, including the most complex queries (Level 4), which re-

quire multiple operations such as filtering, sorting, grouping, and calculating statistics.

Our observations indicate that GoNoGo with 3-shot examples is particularly effective for solving queries with task difficulty up to Level 2 and can handle these tasks without error. For more complex tasks involving Level 3 or Level 4 difficulties, human intervention is recommended to perform the necessary manipulations and computations, rather than relying solely on the automated system.

III.5 Threats to Validity

We identify the following threats to the validity of our study:

Limitation of the Created Benchmark Our study relies on a benchmark specifically created for evaluating the system. While this benchmark is designed to be comprehensive, it may not cover all potential scenarios and edge cases encountered in real-world applications. Efforts have been made to design queries and tasks to be as comprehensive as possible by involving verification engineers to mitigate subjectiveness. However, despite these efforts, the limitation remains that it may not capture every potential scenario and edge case. This limitation could affect the generalizability and robustness of our findings.

Selection of the Foundation Model The choice of the foundation model, in this case GPT-3.5 Turbo, which is considered a widely used LLM in recent studies, might influence the results. Different foundation models, such as GPT-4, GPT-4o, Claude 3, or LLaMA 3, may yield better performance levels and interpretations of the same tasks. However, we limited the project to using GPT-3.5 Turbo and focused on improving its reasoning and planning capabilities. Besides, our framework is flexible and can be easily applied to different pre-trained models. The dependency on a single model means that our conclusions may not hold if another model were used.

III.6 Related Work

The application of tabular data in machine learning holds significant potential, ranging from few-shot learning for data analysis to end-to-end data pipeline automation.

Integrating LLM with tabular data presents several substantial challenges [18]. Most foundation models are not trained on tabular data, making it difficult for them to process and interpret this type of data effectively. To mitigate this issue, pre-training LLMs using tabular data or fine-tuning on specific tasks are two commonly adopted options. [15] described different phases and strategies for LLM training, and [40] provided guidelines for enterprises who are interested in fine-tuning LLMs.

In particular, recent literature has seen a growing interest in pre-training and self-supervised learning (SSL) approaches using tabular data. [21] emphasizes SSL for non-sequential tabular data (SSL4NS-TD), categorizing methods into predictive, contrastive, and hybrid learning, and discussing application issues such as automatic data engineering and cross-table transferability. In contrast, [30] introduces TapTap, a novel table pre-training method that enhances tabular prediction and generates synthetic tables for various applications. Finally, [26] introduces Tabular data Pre-Training via Meta-representation (TabPTM), which enables training-free generalization across heterogeneous datasets by standardizing data representations through distance to prototypes. The common theme across these works is the enhancement of tabular data handling through innovative pre-training and SSL techniques, though they differ in their specific methodologies and application focuses, ranging from generating synthetic data to improving model generalization and manipulation capabilities. [29] proposes Tabular Foundation Models (TabFMs), leveraging a pre-trained LLM fine-tuned on diverse tabular datasets to excel in instruction-following tasks and efficient learning with scarce data.

Pre-training aims to enhance LLMs' capability of handling tabular data in general. However, it does not necessarily improve their performance on specific tasks. On the other hand, fine-tuning pre-trained LLMs have demonstrated potential for enhancing tabular data manipulation on specific tasks. [31] introduced TableLLM, a robust 13-billion-parameter model designed for handling tabular data in real-world office scenarios. In particular, TableLLM incorporates reasoning process extensions and cross-way validation strategies, outperforming existing general-purpose and tabular-focused LLMs. [14] explored zero-shot and few-shot tabular data classification by prompting LLMs with serialized data and problem descriptions, achieving superior performance over traditional deep-learning methods and even strong baselines like gradient-boosted trees. [34] addressed question answering over hybrid tabular and textual data, fine-tuning LLaMA 2 using a step-wise pipeline, resulting in TAT-LLM, which outperforms both prior fine-tuned models and large-scale LLMs such as GPT-4 on specific benchmarks. [24] focused on ap-

plying LLMs to predictive tasks in tabular data, enhancing LLM capabilities through extensive training on annotated tables.

Industrial considerations One known issue is that LLMs often memorize tabular data verbatim, leading to overfitting. [2] highlights that despite their nontrivial generalization capability, LLMs perform better on datasets they were exposed to during training compared to new, unseen datasets. This indicates a tendency towards memorization, necessitating robust testing and validation protocols. This issue is particularly critical for companies' internal data and tasks that a foundation model has not encountered before, as public benchmarks do not necessarily predict performance on these internal tasks. In addition, it is worth noting that some applications have stringent data privacy policies, a concern increasingly being addressed in the literature [3, 5, 46]. In our work, we assume that the data resides within a secure local network, and we do not address data privacy issues in this paper. In industrial settings, practical constraints such as interpretability, user-centric adaptation, ease of development and maintenance, latency requirements, and IT infrastructure limitations are crucial. Our objective is to design a system that addresses these industrial needs without unnecessary complexity and excessive resources typically required by pre-training and fine-tuning LLMs.

III.7 Conclusion

We present the GoNoGo, an LLM-based multi-agent system designed to streamline software release decisions in the automotive industry by analyzing and deriving insights from real-world data using Python code. We have employed this system within our industrial partner's company, which is significantly assisting release managers and reducing the number of engineers engaged in this process, allowing them to focus on their high-level tasks.

The impact of our system extends beyond automation, transforming how automotive companies manage their software release cycles. It reduces the time and effort required for data analysis while increasing decision accuracy and reliability. This shift allows engineers and managers to focus on higher-level tasks, accelerating the overall development and deployment process by bridging the gap between raw data and actionable insights, driving the industry towards more efficient, data-driven software release practices. Without GoNoGo in place, our industrial partner would experience more wasted time and effort across various teams and employees, with the decision-making process becoming significantly prolonged. Pilot users have reported saving approximately 2 hours per person each time they make a decision, highlight-

ing the system's positive impact on efficiency and the industrial partner's overall business goals.

III.8 Acknowledgement

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Bibliography

[1] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[2] S. Bordt, H. Nori, V. Rodrigues, B. Nushi, and R. Caruana. Elephants never forget: Memorization and learning of tabular data in large language models. *arXiv preprint arXiv:2404.06209*, 2024.

[3] A. T. P. Boudewijn, A. F. Ferraris, D. Panfilo, V. Cocca, S. Zinutti, K. De Schepper, and C. R. Chauvenet. Privacy measurements in tabular synthetic data: State of the art and future research directions. In *NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI*, 2023.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] A. N. Carey, K. Bhaila, K. Edemacu, and X. Wu. Dp-tabicl: In-context learning with differentially private tabular data. *arXiv preprint arXiv:2403.05681*, 2024.

[6] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[7] X. Chen, M. Lin, N. Schärli, and D. Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

[8] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, et al. Chatbot arena: An

open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

[9] J. Dagdelen, A. Dunn, S. Lee, N. Walker, A. S. Rosen, G. Ceder, K. A. Persson, and A. Jain. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418, 2024.

[10] Y. Dyachenko, N. Nenkov, M. Petrova, I. Skarga-Bandurova, and O. Soloviov. Approaches to cognitive architecture of autonomous intelligent agent. *Biologically Inspired Cognitive Architectures*, 26:130–135, 2018.

[11] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag. TabLLM: Few-shot Classification of Tabular Data with Large Language Models.

[12] J. Huang and K. C.-C. Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

[13] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

[14] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[15] R. Patil and V. Gudivada. A review of current trends, techniques, and challenges in large language models (llms). *Applied Sciences*, 14(5):2074, 2024.

[16] A. Rahimi and H. Veisi. Integrating model-agnostic meta-learning with advanced language embeddings for few-shot intent classification. In *2024 32nd International Conference on Electrical Engineering (ICEE)*, pages 1–5. IEEE, 2024.

[17] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

[18] B. van Breugel and M. van der Schaar. Why Tabular Foundation Models Should Be a Research Priority, June 2024.

[19] K. VM, H. Warrier, Y. Gupta, et al. Fine tuning llm for enterprise: Practical guidelines and recommendations. *arXiv preprint arXiv:2404.10779*, 2024.

[20] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[21] W.-Y. Wang, W.-W. Du, D. Xu, W. Wang, and W.-C. Peng. A survey on self-supervised learning for non-sequential tabular data. *arXiv preprint arXiv:2402.01204*, 2024.

[22] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[23] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[24] Y. Yang, Y. Wang, S. Sen, L. Li, and Q. Liu. Unleashing the Potential of Large Language Models for Predictive Tabular Tasks in Data Science, 2024.

[25] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[26] H.-J. Ye, Q. Zhou, and D.-C. Zhan. Training-free generalization on heterogeneous tabular data via meta-representation. 2023.

[27] J. Ye, M. Du, and G. Wang. DataFrame QA: A Universal LLM Framework on DataFrame Question Answering Without Data Exposure, 2024.

[28] J. Yoon, R. Feldt, and S. Yoo. Intent-driven mobile gui testing with autonomous large language model agents. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2024.

[29] H. Zhang, X. Wen, S. Zheng, W. Xu, and J. Bian. Towards foundation models for learning on tabular data. *arXiv preprint arXiv:2310.07338*, 2023.

[30] T. Zhang, S. Wang, S. Yan, J. Li, and Q. Liu. Generative table pre-training empowers models for tabular prediction. *arXiv preprint arXiv:2305.09696*, 2023.

[31] X. Zhang, J. Zhang, Z. Ma, Y. Li, B. Zhang, G. Li, Z. Yao, K. Xu, J. Zhou, D. Zhang-Li, J. Yu, S. Zhao, J. Li, and J. Tang. TableLLM: Enabling Tabular Data Manipulation by LLMs in Real Office Usage Scenarios, 2024.

[32] Z. Zhang, Y. Yao, A. Zhang, X. Tang, X. Ma, Z. He, Y. Wang, M. Gerstein, R. Wang, G. Liu, et al. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *arXiv preprint arXiv:2311.11797*, 2023.

[33] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[34] F. Zhu, Z. Liu, F. Feng, C. Wang, M. Li, and T.-S. Chua. TAT-LLM: A Specialized Language Model for Discrete Reasoning over Tabular and Textual Data, Feb. 2024.

IV

GateLens: A Reasoning-Enhanced LLM Agent for Automotive Software Release Analytics

Arsham Gholamzadeh Khoee, Shuai Wang, Robert Feldt, Dhasarathy Parthasarathy, Yinan Yu

Submitted, under review

Abstract

Ensuring reliable data-driven decisions is crucial in domains where analytical accuracy directly impacts safety, compliance, or operational outcomes. Decision support in such domains relies on large tabular datasets, where manual analysis is slow, costly, and error-prone. While Large Language Models (LLMs) offer promising automation potential, they face challenges in analytical reasoning, structured data handling, and ambiguity resolution. This paper introduces GateLens, an LLM-based architecture for reliable analysis of complex tabular data. Its key innovation is the use of Relational Algebra (RA) as a formal intermediate representation between natural-language reasoning and executable code, addressing the reasoning-to-code gap that can arise in direct generation approaches. In our automotive instantiation, GateLens translates natural language queries into RA expressions and generates optimized Python code. Unlike traditional multi-agent or planning-based systems that can be slow, opaque, and costly to maintain, GateLens emphasizes speed, transparency, and reliability. We validate the architecture in automotive software release analytics, where experimental results show that GateLens outperforms the existing Chain-of-Thought (CoT) + Self-Consistency (SC) based system on real-world datasets, particularly in handling complex and ambiguous queries. Ablation studies confirm the essential role of the RA layer. Industrial deployment demonstrates over 80% reduction in analysis time while maintaining high accuracy across domain-specific tasks. GateLens operates effectively in zero-shot settings without requiring few-shot examples or agent orchestration. This work advances deployable LLM system design by identifying key architectural features—intermediate formal representations, execution efficiency, and low configuration overhead—crucial for domain-specific analytical applications where accuracy, traceability, and stakeholder trust are paramount.

IV.1 Introduction

Reliable decision-making in data-intensive domains depends on the ability to accurately analyze large volumes of structured data. In sectors such as automotive manufacturing, healthcare, finance, and regulatory compliance, critical decisions hinge on interpreting tabular datasets that capture test results, operational metrics, or validation records. These datasets often pass through gating steps—critical checkpoints where predefined quality or compliance standards must be met. Failures at these gates can cascade through interconnected processes, delaying dependent workflows regardless of their individual quality. Analysts tasked with safeguarding decision quality must process vast quantities of data. While essential for ensuring accuracy and reliability, this process is time-consuming and prone to human error in data interpretation.

The software industry's transition from manual to automated processes has entered a new era with the emergence of Large Language Models (LLMs) [4, 23]. Companies are increasingly integrating these AI agents into their workflows, seeking more cost-effective and optimized solutions for complex software engineering tasks [19]. However, direct application of LLMs to structured data analysis for decision support is hindered by limitations in interpretable reasoning and understanding of technical specifications [2, 26].

To address these challenges, we introduce *GateLens*, a reasoning-enhanced LLM agent [27] for reliable tabular data analysis to support decision-making in domain-specific contexts. We validate the architecture in the automotive software release domain, where the need for precise, transparent, and efficient analysis is particularly acute.

A key challenge in LLM-based analytical agents is the potential mismatch between natural-language reasoning and the actual computation implemented in generated code—a reasoning-to-code gap that becomes more pronounced as analytical complexity increases. GateLens addresses this challenge by integrating structured relational analysis with domain-specific expertise through a reasoning layer built on Relational Algebra (RA) as an extension of Chain-of-Thought (CoT). This reasoning layer systematically breaks down complex validation tasks into discrete, formally grounded analytical steps. We adopt this approach to address essential limitations of vanilla CoT for our work: its reasoning steps are opaque and often cannot be mapped directly to executable code, its operations are not compositional and cannot be independently reused or debugged, and its reasoning is unstructured with operations blended together without clear intent or formal grounding. In contrast, our RA-based reasoning layer aims to ensure that

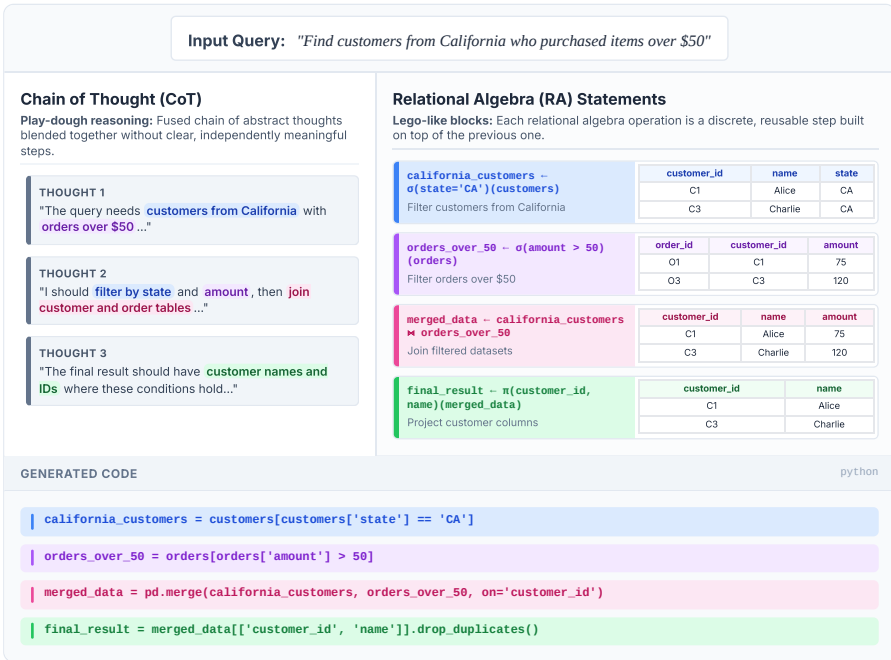


Figure IV.1: Comparison of conventional Chain-of-Thought (CoT) reasoning and its extension using Relational Algebra (RA) statements. (Left) CoT employs unstructured, fused reasoning where multiple analytical concepts are blended together in informal thoughts without clear separation. Operations cannot be independently mapped to executable code snippets, making reasoning steps opaque and difficult to debug in isolation. (Right) RA-based reasoning adopts a structured, compositional approach where each relational algebra operation (filtering, joining, projecting) is a discrete, formally grounded step. Each operation is independently interpretable, reusable, and can be directly mapped to executable code. The color-coding illustrates how RA maintains clear boundaries between distinct analytical steps, whereas CoT conflates multiple operations within single reasoning thoughts. This structural distinction enables transparent, formally grounded reasoning that can be systematically verified and debugged at each stage.

each step is independently interpretable and reusable, reasoning is grounded in formal relational algebra, and intent is made explicit through well-defined operations. Figure IV.1 illustrates this distinction between our structured

RA-based approach and conventional CoT reasoning.

GateLens simplifies three critical aspects of release validation:

1. Test Result Analysis: Analyzing test execution outcomes is foundational to release validation. This involves analyzing pass/fail patterns across comprehensive test suites, identifying recurring failures, and validating test coverage metrics. In automotive software, where a single release might involve a large number of test cases across multiple vehicle functions, this task becomes particularly demanding. Release engineers must not only identify failed tests but also understand their patterns, assess coverage adequacy, and evaluate test execution stability.

2. Impact Assessment: Impact Assessment is a systematic process for evaluating how software issues affect vehicle functionality and safety during release validation. It involves three phases: first, a critical failure analysis identifies the root cause and immediate effects of an issue, such as an ABS module causing a 200ms brake signal delay that exceeds the 100ms threshold. Second, a component-level impact evaluation traces how the issue propagates through interconnected systems, assessing both direct effects, like problematic emergency braking, and indirect effects, such as reduced stability control performance. Finally, an integration risk assessment quantifies the severity of these impacts against safety thresholds and functional requirements, categorizing issues like the ABS delay as system-wide risks with critical severity. This structured process enables engineers to understand system-wide effects, ensuring all safety and functionality requirements are met before release.

3. Release Candidate Analysis: The final quality gate involves evaluating Release Candidates (RCs) against predefined quality gates and criteria. This encompasses analyzing whether a particular RC meets all quality thresholds, identifying potential release blockers, and validating compliance with release requirements. In automotive software, where releases must meet stringent safety and quality standards, this analysis requires careful validation of each RC against established criteria, ensuring all prerequisites for a safe and reliable release are satisfied.

The traditional release validation process demands extensive manual effort. Release engineers meticulously analyze test results, assess impacts, verify RCs against quality gates, and report findings to stakeholders, such as release managers. As automotive software systems grow increasingly complex, these manual workflows become more challenging, time-consuming, and error-prone.

This work aims to streamline release validation by automating key analysis workflows, enabling engineers to focus on high-value analysis and discussion. By providing deeper analytical insights, the proposed approach reduces

the time needed to deliver accurate validation results, empowering release managers to make informed decisions more efficiently. Our **contributions** can be summarized as follows:

- We design an *architecture optimized for time- and safety-critical environments*, minimizing LLM invocations while preserving reasoning depth for reliable tabular analysis. GateLens operates in a zero-shot setting, avoiding the need for few-shot examples or multi-agent coordination, which improves generalizability, execution speed, reduces maintenance overhead, and enhances transparency.

- We introduce a *scalable and maintainable framework for automotive software release validation*, developed in response to observed limitations in traditional planning-based multi-agent system. GateLens handles diverse user queries with higher robustness and clarity, supporting effective decision-making across a wider range of release engineering tasks.

- We conduct a *comprehensive empirical evaluation*, including comparisons with a CoT+SC system, ablation of the RA module, and performance across multiple LLMs (GPT-4o and Llama 3.1 70B). These experiments demonstrate GateLens's performance advantages in complex and ambiguous queries.

- We report on *real-world industrial deployment* in a partner automotive company, where GateLens reduces analysis time by over 80% and demonstrates strong generalization across user roles, highlighting its practical value and deployment-readiness in safety-critical release validation workflows.

IV.2 Background and Motivation

Software release decisions in the automotive industry involve multiple stakeholders and extensive data analysis. Modern vehicles integrate hundreds of software components, each requiring rigorous testing and validation. The process advances through distinct phases: component-level testing, integration testing, system-level validation, and vehicle validation testing. Component-level testing verifies individual software modules, integration testing ensures proper interaction between components, system-level validation examines the complete system behavior, and vehicle validation testing evaluates software performance under real vehicle conditions on closed tracks.

The development cycle grows in complexity with each integration phase. This increasing complexity presents challenges in managing large-scale test results, tracking interdependencies between components, correlating test failures across different subsystems, and maintaining historical context for recurring issues. The iterative nature of software testing and validation further expands this data ecosystem.

The wide range of stakeholders in the release process creates additional challenges in data interpretation and presentation. Project managers need high-level progress indicators, verification engineers require detailed technical insights, quality engineers focus on trend analysis and improvement metrics, and release engineers need specific release-readiness indicators. This variety of perspectives necessitates different views of the same underlying data, making the analysis process more complex.

This diversity is reflected not only in perspective but also in the level of granularity required from the underlying data. For example, senior management may ask high-level questions such as: "List all vehicles that have not yet received global approval," seeking a consolidated overview without reference to schema details or subsystem-specific attributes. In contrast, release managers or verification engineers may pose highly specific queries such as: "What are the baseline, phase, RC, and EUF values for RM-320 in the latest test suite?" These questions rely on detailed knowledge of domain terminology and schema structure. Although both types of queries operate on the same data foundation, they demand substantially different views and levels of abstraction.

Release managers function as gatekeepers in the software deployment pipeline. They handle test result analysis, cross-system impact assessment, decision-making, stakeholder coordination, and safety compliance verification. The manual workflow introduces vulnerabilities: time-intensive processing, potential errors in interpretation, decision delays, and communication barriers between technical and business teams.

Within this process, statisticians provide an overall view of the data to project managers and quality engineers for future business decisions. The existing manual approach faces several limitations, particularly regarding time and resource constraints. These include labor-intensive data analysis, delayed response to critical issues, limited capacity for comprehensive analysis, and bottlenecks in the release pipeline. Communication challenges further complicate the process, with misalignment between technical analysts and statisticians, varying interpretations of project requirements, inconsistent reporting formats, and knowledge transfer gaps.

Internal Testing on Closed Track represents a crucial validation phase.

Release managers must analyze extensive datasets to evaluate progression readiness. The manual query process for report generation can impact release timelines, business objectives, and subsystem integration schedules.

The deployment of intelligent assistants presents opportunities to address these challenges through automated data processing and analysis, standardized reporting frameworks, real-time insights generation, and stakeholder-specific view generation. However, current automation solutions, including basic LLM implementations, face limitations in understanding complex technical specifications, maintaining structured analysis steps, handling domain-specific requirements, and processing automotive validation data systematically.

These limitations highlight the need for enhanced solutions combining domain expertise with advanced analytical reasoning capabilities. Such solutions must facilitate efficient decision-making while maintaining high safety and quality standards in automotive software development [41]. Effective intelligent assistants can transform the release decision process, enabling release managers to prioritize result interpretation and strategic decision-making over routine data analysis.

IV.3 Approach and Methodology

The complexity of automotive software release validation demands a system that can bridge the gap between human-centric inquiries and precise technical analysis. GateLens addresses this challenge by utilizing LLM agents to transform natural language queries into actionable insights through systematic analysis. At its core, GateLens must fulfill three fundamental requirements:

1. Query Understanding: The system must accurately interpret diverse user queries, ranging from high-level management questions to detailed technical inquiries about specific test cases.

2. Query Transformation: The system needs to transform these interpretations into structured formal expressions, ensuring consistent and verifiable reasoning structures.

3. Analysis Execution: The system must generate and execute precise analysis code that processes validation data according to these formal expressions.

The architecture of GateLens is driven by these fundamental requirements, establishing a systematic pipeline for transforming user queries into analytical results. The system leverages RA to enhance LLMs' reasoning capabilities and transform user queries into formal relational operations. To

support this transformation, GateLens employs domain-specific data schemas that guide its relational modeling and enhance the generation of RA expressions. This approach ensures both accessibility for non-technical stakeholders and the precision required for automotive software validation.

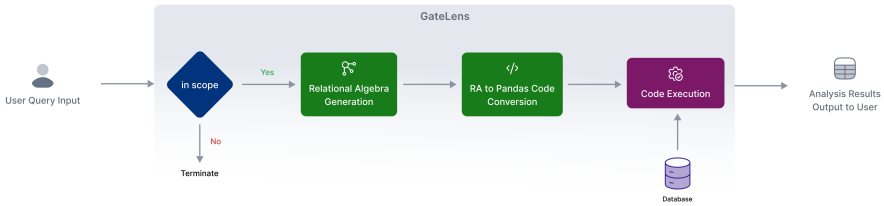


Figure IV.2: GateLens top-level architecture: The system processes high-level queries from the end user, generates the necessary data manipulation code using the enhanced reasoning layer with the help of RA, executes it, and outputs the result table as a decision-support resource.

IV.3.1 System Overview

The primary objective of GateLens is to generate executable code that performs precise test data analysis based on user queries. The system's workflow consists of two main phases: query interpretation and code generation. As illustrated in Figure IV.2, GateLens first processes user queries through an LLM agent that translates natural language inputs into formal RA expressions. This translation incorporates a detailed relational model of the test data, ensuring precise specification of automotive domain concepts. The resulting RA expressions serve as a pivotal intermediate representation that is more transparent to both LLM agents and humans. In the second phase, these formal expressions are passed to the coder agent, which generates executable desirable code, such as SQL or Python code, to perform the required analysis on the test data and produce results.

IV.3.2 Core Components

The system architecture consists of two primary components that work in tandem to transform natural language queries into executable code: the query interpreter agent and the coder agent. The query interpreter agent first translates user queries into RA expressions, providing a structured framework for analytical reasoning. The coder agent then converts these formal expressions into executable code, completing the transformation pipeline.

This two-stage approach ensures both analytical precision and efficient implementation, where the prompt engineering flow and prompt structure are presented in Figure IV.3.

IV.3.2.1 Query Interpreter

The query interpreter agent is responsible for converting user queries into formal RA expressions, providing a precise framework for analytical reasoning. Before initiating this translation, the agent consults the knowledge base, comprising the data schema and domain-specific context. The data schema provides a detailed understanding of the dataset, including its relational modeling, field descriptions, data types, and enriched metadata capturing domain-specific acronyms and terminology mappings. This glossary-enhanced schema is injected into the prompt context, enabling accurate resolution of domain-specific terms into formal schema attributes during RA construction.

To handle imprecise queries without compromising data privacy or exceeding LLM context windows, GateLens employs a selective strategy for exposing categorical information. For low-cardinality attributes (e.g., fields with only a few distinct categories such as test status), all valid options are explicitly enumerated within the schema metadata. In contrast, for high-cardinality attributes, actual database values are never exposed; instead, the schema specifies structural patterns or expected formats (e.g., standard prefixes or identifier conventions). This hybrid design keeps the prompt context compact and privacy-preserving while providing the LLM with sufficient context to generate accurate filtering conditions.

Using this information, the agent verifies whether the query is relevant and within the scope of the dataset [25]. This validation step ensures that only supported and meaningful queries are processed, improving both accuracy and efficiency. Once the query is confirmed to be in scope, the agent leverages the data schemas to interpret and decompose the query into formal RA expressions.

The agent's primary function is to map natural language queries into formal RA expressions, enhancing LLM reasoning through structured decomposition [18]. This approach extends traditional Chain-of-Thoughts (CoT) [43] reasoning by constraining the model to think within a formal system framework [48]. Instead of generating free-form solutions, the agent must express analytics using standard relational algebra notations through a limited set of standard operations: selection, projection, union, set difference, cartesian product, and rename as basic operations, as well as derived operations such as join, intersection, and division and complemented by aggregation func-

tions like average, minimum, maximum, sum, and count.

By limiting operations to this standard set, the agent effectively handles ambiguous queries through formal translation, ensures technical precision, and prevents deviation from analytical requirements. The formal nature of RA enables query optimization, which the agent incorporates by prioritizing data reduction operations early in the expression chain. This optimization strategy involves applying filters first, then performing expensive operations on the reduced dataset, thereby minimizing processing time and resource utilization.

The translation to RA offers two significant advantages. First, it makes the analytics more transparent in technical terms, allowing for clear interpretation and validation of the reasoning process. Second, it ensures that every solution generated is precisely defined and feasible for implementation, preventing the agent from proposing impractical or undefined analytical approaches.

IV.3.2.2 Coder

The coder agent is responsible for generating executable code from given RA expressions. Upon receiving an RA expression, the agent follows precise instructions to produce code that delivers the final analytical results. This capability allows the agent to generate complete, self-contained code at once, eliminating the need for step-by-step generation and execution phases.

To ensure the generated code meets quality standards, we designed prompts that specifically instruct the LLM to include essential validation mechanisms. The prompts explicitly require data type validation instructions for numerical and categorical field handling, null value checking procedures to maintain data integrity, validation of numerical operations, and validation of join conditions to ensure proper matching of key columns between tables.

By generating the entire code in a single pass rather than through iterative refinement, the agent significantly reduces processing overhead, system response time, and resource consumption while minimizing potential errors that could arise from multiple execution steps. This streamlined yet precise approach ensures both efficiency and reliability in the analysis pipeline, fully aligned with the formal rigor of RA expressions and improving overall system responsiveness to user queries.

IV.3.3 Data Handling

A key architectural decision in GateLens is its indirect interaction with test data. Rather than exposing sensitive test data directly to LLM agents, which

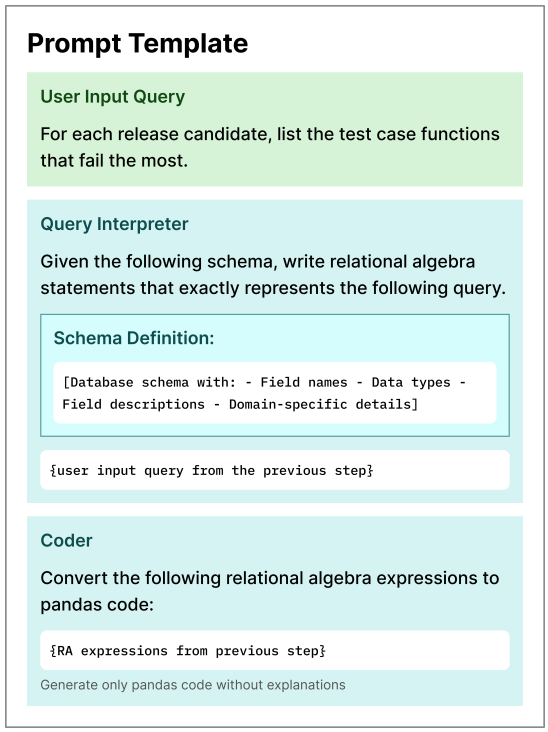


Figure IV.3: Overview of the prompt engineering flow and prompt structure within the GateLens system architecture.

could raise privacy concerns [3] and exceed input context limitations, the system operates on data schemas and relational models. This approach serves multiple critical purposes:

- **Privacy Protection:** Sensitive automotive test data remains secure within the organization's infrastructure
- **Scalability:** The system can handle large-scale test datasets that would exceed LLM context windows
- **Knowledge Integration:** Data schemas and relational models serve as an essential knowledge base, providing necessary structural understanding without raw data exposure

The final execution of the generated code runs on the test data in the target environment, maintaining data privacy while delivering precise analytical results.

IV.4 Experimental Evaluation

In this section, we aim to answer the following research questions:

RQ1: How effectively does GateLens address user queries and deliver accurate results across various query categories?

RQ2: How robust is GateLens in handling out of scope queries and imprecise queries?

RQ3: How does the RA reasoning procedure contribute to the overall performance of GateLens?

RQ4: To what extent does RA-based reasoning eliminate the need for in-context learning?

IV.4.1 Experimental Setup

To address the research questions introduced in Section IV.4, we designed and conducted extensive experiments to evaluate the performance of GateLens.

The experimental data comprises two distinct benchmarks. The first benchmark consists of 50 queries designed with the assistance of release engineers, quality engineers, and verification engineers. To assess GateLens's performance across a spectrum of query complexities, these queries are categorized into four difficulty levels. The four levels of query difficulty are defined as follows:
Level 1 Simple queries involving a single operation such as filtering or sorting.
Level 2 Queries combining two or three basic operations, such as multiple filtering followed by sorting.
Level 3 Queries involving more than three operations, potentially including grouping and aggregating.
Level 4 Complex queries requiring multiple advanced operations beyond basic filtering and sorting, such as grouping and aggregating for statistical calculations.

The second benchmark is derived from real-world user queries collected from production logs at our partner company. These queries were sourced from the historical logs of an agentic system that employed a well-established tabular data reasoning approach combining CoT [43] prompting with Self-Consistency (SC) [42]. This system was used in production to support software release analytics, and the collected queries reflect a wide range of user

roles, query types, and domain-specific requirements. While this system performs effectively in many scenarios, its limitations become apparent as the range of roles and users expands, leading to a significant diversification of queries. This broader query diversity exposes the system's reliance on few-shot examples, making it less capable of handling highly complex, ambiguous, or ill-defined queries that require greater flexibility and adaptability. Nevertheless, this preliminary system played a critical role in data collection for GateLens by providing query logs used to develop and validate our approach. From these logs, we filtered out near-duplicates and selected 244 frequently repeated unique queries, which we then organized into eight functional categories based on their purposes. The ground truth for the 244 real-world queries was established through a two-stage manual annotation process: two domain experts independently solved an initial subset of 20 queries to align on interpretation and expected outputs, after which the remaining queries were divided and annotated following the agreed-upon criteria, with periodic cross-checks to ensure consistency.

In order to assess GateLens's performance, we run experiments with two large language models (LLMs): GPT-4o, a leading commercial model, and Llama 3.1 70b, a recently released open-source model. We also benchmark GateLens against the CoT+SC [20] agentic system currently used to support the company's release decisions. Our comparative analysis is designed to quantify the improvements introduced by GateLens's novel architecture.

To address the challenge of handling out-of-scope user queries during real-time interactions, GateLens incorporates an in-scope filtering mechanism as explained in Section IV.3.2. This mechanism ensures that the system only attempts to process queries that fall within its scope, thereby improving reliability and reducing errors. Performance evaluation focused on two key aspects:

1. **Quality of responses**: Measured using precision, recall, and F1 Score, which reflect the system's ability to address relevant queries correctly.

2. **Coverage of relevant queries**: Ensuring the system does not reject a significant proportion of valid queries, thus maintaining broad applicability.

Additionally, an ablation study is conducted to examine the contribution of the RA reasoning mechanism of GateLens.

In our experiments, the evaluation of system performance is based on the following definitions: A **True Positive (TP)** occurs when the system produces a result that matches the manually generated ground-truth result, specifically when the final output of the executed code matches the expected

ground-truth output. A **False Positive (FP)** occurs when the system provides an incorrect result, meaning the executed code produces output that differs from the ground-truth. A **False Negative (FN)** occurs when the system fails to provide any result for a query. **True Negatives (TNs)** are not applicable since we focus on valid queries producing meaningful output.

Based on these definitions, we calculate Precision, Recall, and F1 scores to assess the performance of the system. Precision ensures that incorrect results are minimized, recall ensures relevant queries are addressed, and the F1 score balances the two to provide an overall assessment of system performance. By relying on a closed-set benchmark with established ground truths, these metrics enable us to rigorously isolate and measure the impact of our architectural choices during lab validation, setting the stage for the real-world industrial evaluation detailed in Section IV.6.

IV.4.2 Performance in Addressing User Queries (RQ1)

Table IV.1: Comparison of average token consumption and reduction between CoT+SC and GateLens across four difficulty levels. The evaluation was conducted on the first benchmark, which consists of 50 designed queries with annotated difficulty levels, with both agents utilizing GPT-4o.

Level	# Queries	GateLens with GPT-4o			GateLens with Llama 3.1 70B			CoT+SC with GPT-4o			CoT+SC with Llama 3.1 70B		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
1	16	100%	100%	**100%**	100%	43.75%	60.87%	93.33%	87.5%	90.32%	100%	93.75%	96.77%
2	16	100%	100%	**100%**	100%	62.5%	76.92%	100%	81.25%	89.66%	92.31%	75%	82.76%
3	12	100%	100%	**100%**	100%	50%	66.67%	91.67%	91.67%	91.67%	90.91%	83.33%	86.96%
4	6	100%	100%	**100%**	100%	33%	49.62%	66.67%	66.67%	66.67%	60%	50%	54.55%
Total	**50**	**100%**	**100%**	**100%**	**100%**	**47.31%**	**63.52%**	**87.91%**	**81.77%**	**84.57%**	**85.81%**	**75.52%**	**80.26%**

Table IV.2: Performance comparison of GateLens and the CoT+SC system across different categories on the second benchmark, which consists of 244 real-world queries.

Category	# Queries	GateLens with GPT-4o			GateLens with Llama 3.1 70B			CoT+SC with GPT-4o		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Column Operations	17	64.7%	64.7%	64.7%	50%	11.76%	19.04%	76.47%	76.47%	**76.47%**
Complex Multi-Condition Queries	77	86.3%	81.82%	**84%**	100%	24.68%	39.59%	84.75%	64.94%	73.53%
Conditional Calculations	8	100%	87.5%	**93.3%**	100%	37.5%	54.55%	87.5%	87.5%	87.5%
Data Filtering	32	89.66%	81.25%	**85.25%**	90.91%	31.25%	46.51%	86.21%	78.13%	81.97%
Duplicate Removal	78	87.67%	82.05%	**84.77%**	100%	23.08%	37.5%	75.93%	52.56%	62.12%
Grouping and Aggregation	10	80.0%	80.0%	**80.0%**	100%	30%	46.15%	83.33%	50%	62.5%
Metadata Queries	13	91.67%	84.61%	**88.0%**	100%	15.38%	26.66%	46.15%	46.15%	46.15%
Table Generation	9	88.89%	88.89%	**88.89%**	100%	44.44%	61.53%	88.89%	88.89%	**88.89%**
Total	**244**	**86.02%**	**81.14%**	**83.51%**	**92.61%**	**27.26%**	**41.44%**	**83.15%**	**63.52%**	**70.61%**

We conducted experiments to compare the performance of GateLens across the two introduced benchmarks. The first benchmark, consisting of 50 queries categorized by difficulty levels, was used to evaluate and compare the per-

formance of GateLens and CoT+SC. Both systems were tested using GPT-4o and Llama 3.1 70B as their underlying LLMs. The results are summarized in Table IV.1.

The results demonstrate that GateLens with GPT-4o significantly outperforms GateLens with Llama 3.1 70B, indicating GPT-4o's superior capability for interpreting and generating RA. Similarly, CoT+SC with GPT-4o outperforms its Llama 3.1 70B variant, with the performance gap growing as query complexity increases. CoT+SC performance declines with query complexity. This underscores the importance of the RA reasoning mechanism in GateLens, which enables effective handling of complex, unstructured queries by decomposing them into logical, structured expressions. Most notably, GateLens with GPT-4o achieved optimal performance on this benchmark, maintaining 100% accuracy across all difficulty levels. This stems from integrating RA reasoning into our framework. By translating queries into RA expressions, GateLens explicitly captures the logical structure of operations, enhancing both the clarity and precision of the generated code. The intermediate RA conversion allows the system to focus on the relevant table operations while filtering out irrelevant elements in the query, greatly enhancing the problem-solving capabilities of the LLM agent.

For the second benchmark, results in Table IV.2 show that GateLens (GPT-4o) and CoT+SC (GPT-4o) significantly outperformed GateLens with Llama 3.1 70B. This performance disparity is primarily due to the strict code generation requirements of the task, including table filtering, merging strategies, and key-value mapping operations, where GPT-4o demonstrated markedly superior capabilities.

GateLens with GPT-4o outperformed CoT+SC (GPT-4o) across most categories, particularly evident in Metadata Queries (those seeking basic table information). For example, when processing the query "Give me the list of release candidates," the CoT+SC system often fails to identify the correct field. A common failure mode in CoT+SC occurred when user queries included typographical errors or incorrect casing in field names, with the system directly using the erroneous fields without correction. GateLens addresses this limitation through its query-to-RA transformation process, which incorporates the database's relational model, adjusts query fields to match table formats, and can handle fuzzy matching to detect and correct field names, enabling the system to resolve typographical errors and ambiguous queries effectively. This approach improves accuracy and resilience, particularly in real-world scenarios where user queries may not always adhere to strict formatting standards.

In addition, compared to the CoT+SC solution, GateLens uses signifi-

Table IV.3: Comparison of average token consumption and reduction between CoT+SC and GateLens across four difficulty levels. The evaluation was conducted on the first benchmark, which consists of 50 designed queries with annotated difficulty levels, with both agents utilizing GPT-4o.

Level	Agent	Avg Input Tokens	Avg Output Tokens	Avg Total Tokens	Token Reduction
1	CoT+SC	11,905	186	12,091	–
	GateLens	2,239	420	2,658	↓ 78%
2	CoT+SC	13,747	368	14,116	–
	GateLens	2,428	701	3,129	↓ 78%
3	CoT+SC	14,726	441	15,168	–
	GateLens	2,432	698	3,130	↓ 79%
4	CoT+SC	17,103	452	17,555	–
	GateLens	2,505	847	3,352	↓ 81%

cantly fewer tokens due to its effective use of RA as the intermediate representation and its zero-shot architecture. In Table IV.3, we compare the token usage of both systems across different difficulty levels. Notably, while the CoT+SC approach requires increasingly massive input contexts, primarily to accommodate few-shot examples, GateLens maintains a highly compact input footprint. As the query difficulty increases, the token reduction becomes even more significant, demonstrating the efficiency and scalability of the RA-based approach in handling complex queries.

RQ1 findings: GateLens with GPT-4o achieved 100% F1 score on the first benchmark across all difficulty levels and 83.51% F1 score on the second benchmark with 244 real-world queries. It outperformed CoT+SC (GPT-4o) by approximately 13 percentage points on real-world queries, indicating that the RA reasoning mechanism effectively addresses both complex logical operations and real-world noise, such as typos and ambiguous field names.

IV.4.3 Robustness: Handling Out of Scope and Imprecise Queries (RQ2)

To assess the robustness of our approach in handling diverse user queries under real-world conditions, we conducted further experiments focusing on filtering out-of-scope queries as well as processing imprecise queries. For this purpose, the data analysis team at our industrial partner company manually selected 37 out-of-scope queries and 50 imprecise queries from the historical logs of the first-generation system, which are used to perform targeted evaluations.

Out-of-scope queries are those that cannot be meaningfully answered

using the available data. For example, a query like "What is the most beautiful truck?" requires subjective judgment and cannot be resolved through database operations; it should be identified and filtered as out of scope. On the other hand, imprecise queries are those that can be answered using the database but contain ambiguous or inexact terms. For instance, a query such as "Find some trucks for cases that are NOK" is considered imprecise because while it seeks truck names where test results are "NOK" (failed), it uses ambiguous terminology - referring to "trucks" instead of the actual database field "name", and mentions "NOK" without specifying the "test_result" field. Such imprecise queries require mapping informal language to precise database fields and conditions for proper execution.

IV.4.3.1 Handling Out of Scope Queries

We compared GateLens with other models; the results can be found in Table IV.4. The results demonstrate that GateLens with GPT-4o achieved the best performance, particularly in terms of precision, which is approximately 40% higher than other models, indicating GateLens' ability to avoid generating incorrect results.

The superior precision of GateLens with GPT-4o can be attributed to two key aspects of its design. First, its robust filtering mechanism ensures that out-of-scope queries are identified and excluded early in the processing pipeline, preventing irrelevant results. Second, the conversion of raw natural language queries into structured RA expressions enables the model to isolate and capture task-relevant components of a query. This structured approach considerably decreases erroneous outcomes and enhances the model's ability to handle complex and diverse query formulations in real-world scenarios.

Table IV.4: Model comparison for out-of-scope queries.

Model	Precision	Recall	F1 Score
GateLens with GPT-4o	92.5%	100%	**96.10%**
GateLens with Llama 3.1 70B	52.94%	97.30%	68.57%
CoT+SC with GPT-4o	51.10%	89.19%	64.97%

CoT+SC showed significantly lower precision due to the variability of real-world queries and the inconsistency of user narratives, which often contain a mix of relevant and irrelevant content. This variability increases uncertainty and poses challenges for models that struggle to identify task-relevant information. Although all models demonstrated high recall, this did not translate into accurate processing.

Table IV.5: Model comparison for imprecise queries.

Model	Precision	Recall	F1 Score
GateLens with GPT-4o	92.86%	78%	**84.78%**
GateLens with Llama 3.1 70B	92.86%	26%	40.63%
CoT+SC with GPT-4o	90%	36%	51.43%

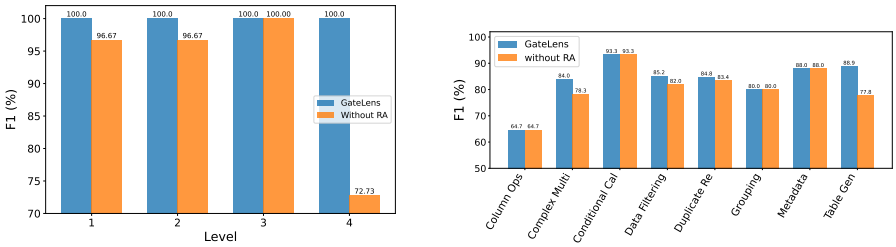
IV.4.3.2 Handling Imprecise Queries

To further assess the robustness of our approach, we evaluated its performance in handling imprecise queries, which posed two primary challenges. First, some queries are informal and conversational in style, appearing unrelated to data analysis but actually carrying relevant intent. Second, many queries referred to fields using terms differing from the column headers.

The results of these experiments are presented in Table IV.5. As shown, GateLens with GPT-4o demonstrates the best overall performance. In terms of precision, all methods performed relatively well, indicating that when results are generated, they are likely to be correct. However, our method significantly outperformed the others in recall, highlighting its ability to handle a larger portion of the imprecise queries. As a result, GateLens with GPT-4 achieved a substantially higher F1 score compared to other methods, demonstrating that it not only processes most queries but also produces accurate results for them.

The observed performance gap between GateLens with GPT-4o and the other models can be attributed to their inherent limitations. Specifically, the Llama 3.1 70B model struggled to interpret user queries that deviated from the exact column header descriptions in the database schema. In such cases, Llama 3.1 70B often converted only the clearly defined parts of the query into RA, leading to incomplete execution and reduced accuracy. On the other hand, CoT+SC exhibits low recall, as it is highly susceptible to confusion by ambiguous query elements. This causes CoT+SC to frequently generate incorrect code that fails execution, significantly lowering its recall rate.

RQ2 findings: GateLens demonstrates high robustness in real-world conditions. It effectively filters out-of-scope inputs, achieving approximately 40% higher precision than the baseline system. Furthermore, it handles imprecise or informal queries with superior performance, more than doubling the recall (78% vs. 36%). This confirms that the system can manage ambiguous user inputs without sacrificing the accuracy of the generated code.



(a) The first benchmark with annotated difficulty levels.

(b) The second benchmark with real-world user queries

Figure IV.4: Comparison of the original method and the method without the RA module across different datasets.

IV.4.4 Effectiveness of the RA module (RQ3)

To evaluate the impact of the RA module that converts user queries into RA expressions, we conducted experiments by removing the RA module from the framework and comparing the results to the original system. The outcomes, shown in Figure IV.4, demonstrate significant performance degradation across both benchmarks when operating without the RA module.

In the first benchmark, performance declined most notably for Level 4 queries, showing a drop exceeding 27%. These queries, which involve advanced operations like grouping, aggregating, and statistical calculations, demonstrated that RA translation is particularly crucial for handling queries with multiple, intricate operations. Similarly, the second benchmark shows decreased performance in complex tasks such as multi-condition filtering, duplicate data removal, and table generation, further emphasizing RA's effectiveness in managing complex database operations.

The RA module maintained consistent performance for simpler queries, demonstrating its versatility across varying complexity levels. By transforming natural language into precise, logical representations, the RA module serves as a key bridge between user intent and code execution. This translation process enables the code generator to produce accurate, efficient executable code for data analysis tasks.

RQ3 findings: The RA module is a critical component for handling query complexity. Removing it results in substantial performance degradation (over 27% for complex queries), confirming that translating natural language to RA provides necessary structural guidance for accurate code generation.

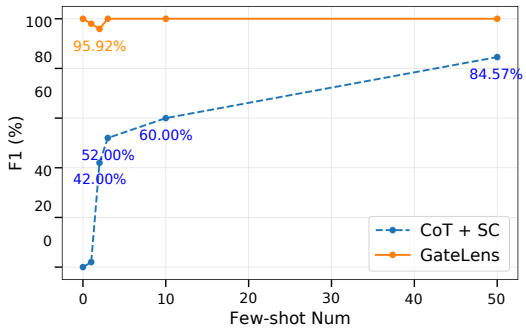


Figure IV.5: Comparison of GateLens against CoT+SC across different numbers of few-shot examples.

IV.4.5 Role of Few-Shot Examples (RQ4)

To investigate the effect of including few-shot examples in prompts, we conducted experiments by varying the number of examples provided to both GateLens and CoT+SC. This experiment is performed on the first benchmark containing 50 designed queries, with results illustrated in Figure IV.5.

The results demonstrate that GateLens relies heavily on its RA translation process, achieving 100% F1 even in a 0-shot setting without any examples. Interestingly, when a small number of examples are added, GateLens becomes slightly biased toward them, leading to a minor degradation in performance (dropping to 95.92% at 2 examples). However, it regains optimal performance at 3 examples and remains at 100% thereafter. In contrast, CoT+SC's performance heavily depends on in-context examples, achieving only 42% F1 with 2 examples and showing suboptimal results without sufficient few-shot examples. Performance improved steadily with more examples, reaching 84.57% at 50 examples.

RQ4 findings: GateLens does not require many-shot examples to achieve high performance, delivering optimal results in a 0-shot setting. This contrasts with CoT+SC, which depends on carefully curated examples. This independence from few-shot examples makes GateLens more efficient for real-time applications by reducing context size and computational overhead.

IV.5 Qualitative Assessment Examples

To provide a closer view into the inner workings of the GateLens frame-

Table IV.6: An example of an input query and its corresponding relational algebra expressions translated using a domain-specific database schema, along with the executable code generated from the relational algebra and the resulting output produced when executed over the database.

Stage	Content
Input Query	*Compute the relative frequency of not run tests for each RC and function*
Relational Algebra	**Step 1:** $R0 \leftarrow \sigma_{\text{result\_status="NotRun"}}(\text{df})$ **Step 2:** $R1 \leftarrow \gamma_{\text{releasecandidate,function,countNotRun}\rightarrow\text{count}}(R0)$ **Step 3:** $R2 \leftarrow \gamma_{\text{releasecandidate,function,totalcount}\rightarrow\text{count}}(\text{df})$ **Step 4:** $R3 \leftarrow R1 \bowtie_{\text{R1.releasecandidate=R2.releasecandidate} \wedge \text{R1.function=R2.function}} R2$ **Step 5:** Result $\leftarrow \gamma_{\text{R3.releasecandidate,R3.function,relativeFrequency}\rightarrow\frac{\text{R3.count}}{\text{R3.totalcount}}}(R3)$

Python Code	`# Step 1: Select rows where result_status is "NotRun"`
	`R0 = df[df['result_status'] == 'NotRun']`
	
	`# Step 2: Group by releasecandidate and function, count "NotRun" entries`
	`R1 = R0.groupby(['releasecandidate', 'function']).size()`
	`  .reset_index(name='count')`
	
	`# Step 3: Group all rows by releasecandidate and function, count total entries`
	`R2 = df.groupby(['releasecandidate', 'function']).size()`
	`  .reset_index(name='totalcount')`
	
	`# Step 4: Join R1 and R2 on releasecandidate and function`
	`R3 = pd.merge(R1, R2, how='inner',`
	`  left_on=['releasecandidate', 'function'],`
	`  right_on=['releasecandidate', 'function'])`
	
	`# Step 5: Compute relative frequency of "NotRun" tests`
	`R3['relativeFrequency'] = R3['count'] / R3['totalcount']`
	`result = R3[['releasecandidate', 'function',`
	`  'relativeFrequency']]`

Output	**RC** **Function** **Freq** ALIEN func_001 0.833333 ALIEN func_002 0.142857 ALIEN func_003 0.125000 ALIEN func_004 0.142857 PACMAN func_001 0.325000 ⋮ ⋮ ⋮ Total rows: 3,662

work, we analyze two representative examples. Each example presents the relational algebra solution generated from the input query using a domain-specific relational model, the corresponding executable code created from

Table IV.7: An example of an incorrect implementation where the relational algebra translation is semantically correct but the generated code fails to properly implement the projection operation, returning all occurrences of the baseline value rather than the distinct value as specified by the relational algebra.

Stage	Content
Input Query	*What is the latest baseline on RM-320?*
Relational Algebra	**Step 1:** $R0 \leftarrow \sigma_{\text{name="RM-320"}}(\text{df})$ **Step 2:** max_date $\leftarrow \text{MAX}(\pi_{\text{date}}(R0))$ **Step 3:** $R1 \leftarrow \sigma_{\text{date=max\_date}}(R0)$ **Step 4:** Result $\leftarrow \pi_{\text{baseline}}(R1)$

Python Code	`# Step 1: Select rows where the truck name is RM-320`
	`R0 = df[df['name'] == 'RM-320']`
	
	`# Step 2: Identify the latest date from the selected rows`
	`max_date = R0['date'].max()`
	
	`# Step 3: Select rows that have this latest date`
	`R1 = R0[R0['date'] == max_date]`
	
	`# Step 4: Project the baseline field`
	`result = R1['baseline'].reset_index(drop=True)`

Output	**Baseline** 23_T2_w2509_ABC_12 23_T2_w2509_ABC_12 23_T2_w2509_ABC_12 23_T2_w2509_ABC_12 23_T2_w2509_ABC_12 ⋮ Total rows: 132 (all duplicates)

the relational algebra statements, and the resulting output after execution.

Table IV.6 represents an example that produces the desirable result. The agent correctly decomposes the input query into a sequence of relational algebra statements that include filtering, grouping by release candidate and function, counting entries with the status "NotRun," joining the intermediate results, and finally computing the relative frequency. This process shows how the system is capable of understanding the semantics of the query, applying the required relational transformations, and leveraging the domain-specific schema to produce the correct result. This stepwise reasoning lays the foundation for semantic decomposition and precise query interpretation, which then naturally translates into valid executable code.

Table IV.7 is an example that illustrates a failed case. Although the rela-

tional algebra translation is semantically sound, the resulting code does not correctly apply projection—it returns all occurrences of the same baseline value rather than the distinct value expected by the relational algebra. This issue arises because the plan omits the duplicate-elimination operator (δ), which should be used to ensure the final result contains unique entries. As a result, while the output does not match the intended result, it is still potentially useful to the end user, as it contains relevant baseline information.

Overall, these examples highlight how the reasoning-enhanced LLM agent supports accurate and explainable code generation by decomposing user queries into relational algebra operations that align closely with domain-specific requirements. The relational modeling serves as a crucial guide for transforming natural language queries into semantically coherent, executable plans.

IV.6 Industrial Deployment: Lessons Learned

Table IV.8: GateLens (zero-shot) vs CoT+SC (few-shot) performance across different roles on the second benchmark. For each role tested, CoT+SC was trained using examples from the other two roles only (leave-one-role-out approach).

Roles	# Queries (244 in total)	GateLens	CoT+SC (Few-shot with All Roles)	CoT+SC (Few-shot with Leave-One-Role-Out)		
		F1 Score	F1 Score	Without Mechanic	Without Project	Without Software
Mechanically-oriented	36	94.25%	80.60%	73.85% ↓ (-6.75%)	86.57%	80.60%
Project-oriented	193	80.21%	78.93%	78.93%	76.22% ↓ (-2.71%)	78.40%
Software-oriented	15	100%	100%	100%	100%	82.76% ↓ (-17.24%)

The deployment of GateLens at a partner automotive company has provided valuable insights into integrating AI-assisted analytics into complex industrial workflows, specifically for streamlining decision-making in automotive software release validation.

Automotive software integration at the company typically occurs across three hierarchical stages: subsystem (control unit), system (multiple control units), and full vehicle levels. Each stage involves extensive testing, with results stored in a central database. Critical Go/No-Go decisions are made at these stages to determine whether a release meets quality thresholds. However, stakeholders from diverse backgrounds—including project managers, mechanical engineers, and software engineers—must query the raw data to evaluate product quality. Many lack expertise in data analytics, creating bottlenecks and delays in the decision-making process.

Previously, these analytics were managed by a small team of 2–3 full-time analysts, who were often overwhelmed by the volume and diversity of requests. Scaling the team to meet the current demand would have required

tripling its size. GateLens addresses this challenge by automating much of the workload, enabling more efficient decision-making. Currently, GateLens is in an extended pilot phase, supporting a pool of 60-80 users. The analytics team has transitioned to a support role, helping stakeholders articulate their needs into clear, actionable prompts for the system. User adoption of GateLens has progressed in phases:

- **Small-Scale Pilot**: The initial deployment within the analytics team established benchmarks.
- **Expanded Pilot**: Five additional users from varied backgrounds contributed to refining the benchmarks.
- **Wider Rollout**: The current phase involves a larger group of 60–80 users. Feedback has been highly positive, with stakeholders recognizing GateLens's ability to simplify and accelerate complex analyses.

Since the launch, the number of both new and recurring users has grown, encompassing diverse roles and types of queries, thereby demonstrating the tool's increasing utility and trust. GateLens significantly reduces the time and effort required for complex analyses, but the shift towards automation also requires users to take on more responsibility in defining and clarifying their needs. The transition from a primarily supportive tool to a more fully automated system is ongoing, demanding a gradual approach with careful calibration to ensure the tool continues to meet evolving needs.

As the system was opened to a broader audience, the diversity of query types increased substantially. While the initial CoT+SC-based agent performed well for a relatively homogeneous user group, its performance became increasingly sensitive to the coverage of few-shot examples. Approaches that rely heavily on few-shot prompting are inherently constrained by example selection and may struggle with previously unseen query patterns. This sensitivity limits their scalability in dynamic industrial environments where new query types continuously emerge. For this reason, we prioritized architectural choices that improve robustness and generalization across roles and query styles.

To explore the system's generalizability, we categorized the roles within the company into three groups: mechanically-oriented, project-oriented, and software-oriented roles. Mechanically-oriented roles typically focus on truck-specific data filtering. Project-oriented roles often combine meta-queries with conditional filters for release management and statistical analysis. Software roles emphasize truck software applications and user functions. We can see from Table IV.8, both GateLens (zero-shot) and CoT+SC (few-shot) exhibit

differences in system performance across these groups, which is likely stemming from the complexity and variety of their typical queries. Nevertheless, the results demonstrate that GateLens is capable of supporting all groups to a high degree. To further evaluate CoT+SC's dependency on few-shot examples, we conducted a **leave-one-role-out** experiment. In this approach, examples from a specific role are excluded in each iteration. For instance, 'without software' indicates that all examples from the software-oriented role have been removed, while the total number of examples is maintained by substituting them with examples from other roles. This highlights the potential challenges with the robustness and generalizability of techniques that rely on few-shot examples. This is a crucial factor to consider in industries where diverse teams collaborate and a wide range of queries may arise.

The impact of automated systems, such as GateLens, on the release process has been substantial. Compared to the previous manual process, GateLens has reduced the time required for Go/No-Go analytics by more than 80%, significantly improving operational efficiency. Crucially, the reported 80% reduction reflects an operational end-to-end improvement that includes the time engineers spend verifying system outputs. In safety-critical industrial environments, AI-generated results are never accepted without validation; therefore, laboratory correctness metrics alone are insufficient to assess real-world impact. A traditional LLM pipeline that directly translates a natural-language query into executable code and returns a final result is not practical in this context. When such black-box outputs are cross-checked against existing dashboards—a natural and common validation strategy—any discrepancies become difficult to diagnose. If the reasoning process is opaque, engineers cannot trace the source of the error, which directly undermines trust and limits usability. On the other hand, when the reasoning process is expressed in natural language, it becomes possible to follow where things went wrong, but precise intervention remains difficult because natural language is inherently fuzzy and imprecise.

In contrast, GateLens generates an explicit relational algebra (RA) plan that decomposes the input query into logical, stepwise building blocks before producing executable code. This intermediate representation mirrors how experienced developers would structure complex analyses. As a result, engineers can inspect whether the decomposed plan is logically sound before or alongside reviewing the final output. This makes discrepancies diagnosable rather than opaque; the RA-based intermediate representation decreases the effort required for validation and actively builds user trust over time. Consequently, stakeholders can now focus on high-level decision-making, freed from the burden of data preparation and analysis.

A key advantage of GateLens lies in its domain-specific design. Unlike general-purpose tools like TaskWeaver [33] or AutoGen [45], GateLens is tailored to automotive workflows, making it easier to understand, debug, and adapt to automotive common procedures. This focus on domain relevance ensures that the system aligns more closely with stakeholders' needs while providing reliable and nuanced support.

Post-deployment monitoring revealed practical failure modes that highlight challenges of LLM adoption in industrial settings. Most issues did not stem from complex reasoning errors, but from ambiguities in user queries. The two most common cases were: (i) implicit constraints, where users assumed a specific test environment or time frame without stating it explicitly, and (ii) highly localized team jargon that led to incorrect schema mappings. To address these issues, we systematically documented recurring patterns, refined prompt guidelines, and extended the schema metadata with an explicit jargon glossary, incorporated as domain-specific context to improve term-to-column alignment. This process also repositioned the central analytics team toward supporting clearer, more explicit query formulation.

In summary, the deployment of GateLens demonstrates how domain-specific AI solutions can transform critical workflows in the automotive sector. By automating labor-intensive processes and enhancing decision-making, GateLens has delivered measurable improvements in efficiency and user satisfaction. However, its success depends on ongoing refinement and careful management of the transition to full(er) automation. Balancing automation with user empowerment remains crucial, particularly in a complex industry like automotive, where diverse stakeholder needs must be met.

GateLens represents a promising step forward, showcasing the potential of AI-driven systems to improve not only the automotive domain but also other industries requiring robust, scalable solutions for intricate processes.

IV.7 Related Work

General-purpose LLMs are primarily designed for and trained on natural languages. Working with tabular data requires specialized adaptations to effectively handle its structured and heterogeneous nature [10, 39, 43]. First, the structured tabular data is typically transformed into serialized text. The performance of the LLM may depend on this transformation [28]. Subsequently, the serialized text data is used as input to the LLM for various tasks, such as question-answering, summarization, or logical reasoning. Common approaches to improve LLM performance include prompt engineering, pretraining, fine-tuning, and Retrieval-Augmented Generation (RAG).

Pre-training and fine-tuning [8, 14, 31, 40, 47] often face scalability concerns. Although resource-efficient training techniques have been proposed to mitigate the substantial computational demands of LLMs [13, 22], in safety-critical applications with evolving data and requirements, training LLMs presents significant challenges due to the constant need for rigorous validation and verification. This ongoing necessity substantially increases resource demands for development and maintenance, potentially exceeding the capacities of many companies. Techniques such as RAG have been employed to dynamically integrate external knowledge bases during inference, reducing the need for frequent model updates [11, 49]. However, such methods can pose challenges in safety-critical industrial settings as well, since both retrieval modules and model components must undergo synchronized updates to maintain relevance, reliability, and compliance with validation and verification requirements. Costs would also be especially high with fine-tuning since retuning would be needed when new and improved base LLMs are released and should be incorporated.

Prompt engineering techniques are among the most resource-efficient methods for improving LLM output [16, 35]. From a user standpoint, when the input is natural language, prompting techniques can be broadly categorized based on the type of language generated by the LLM. These include outputs in natural language, structured languages [25], or symbolic languages. When the generated language is natural language, LLMs often fail to consistently follow instructions, particularly when the instructions are complex or require precise, step-by-step execution [32]. This inconsistency arises because natural language, while flexible and expressive, can be ambiguous and prone to misinterpretation by LLMs. Structured languages include general-purpose languages (e.g. Python) [46], query languages (e.g. SQL) [9, 21, 29], configuration formats (e.g. YAML or JSON), or other Domain-Specific Languages (DSLs) [7, 12]. These languages are subsequently interpreted and/or executed by either external tools, the same LLM, or another LLM agent. This approach offers significant advantages, as it enables precise execution of tasks. Another popular type of output is symbolic languages. Literature shows that symbolic representations provide a more rigorous framework for articulating premises and intent, which can enhance reasoning capabilities [30].

In this paper, we introduce a novel prompt-only (training-free) approach that bridges natural language and executable code through RA, a symbolic formalism designed for relational modeling and ideally suited for analyzing tabular data. Unlike prior work that often relies on complex multi-agent planning, our approach leverages RA as a lightweight intermediate repre-

sentation to enable precise query normalization, disambiguation of natural language input, and efficient code generation. RA acts as an abstraction layer that can target multiple execution backends (e.g., Python, SQL), providing adaptability across systems. In GateLens, we generate Python code to support practical industrial deployment and high-performance execution. GateLens is *training-free*, *feed-forward* (single-pass, without looping or multi-agent orchestration), and thus easier to verify, trace, maintain, and trust – qualities critical for safety-critical industrial applications.

IV.8 Discussion and Conclusions

This study introduced GateLens, a reasoning-enhanced LLM architecture for reliable tabular analysis, applied to domain-specific software release validation in the automotive industry. By introducing RA as an intermediate representation before code generation, GateLens addresses the "Unfaithful Chain-of-Thought (CoT) reasoning" problem in code generation, where reasoning steps in CoT explanations do not accurately reflect the model's actual thought process [38]. Specifically, GateLens divides the analysis into two steps: (1) natural language queries are first translated into RA expressions, and (2) these RA expressions are then converted into executable code. We use Python as our target language in step (2) due to its widespread use in our partner company and the model's strong performance in Python, which benefits from more extensive training data. However, this step is also compatible with RA-to-SQL generation, enabling flexibility across backends. The inclusion of the RA reasoning module is a critical factor in improving robustness and scalability, as evidenced by superior F1 scores in both benchmarking and industrial evaluations.

Our findings regarding few-shot learning (RQ4) highlight a notable architectural advantage of GateLens over conventional few-shot learning approaches. While CoT+SC's performance improves with more examples, this approach introduces several practical and technical challenges. Increasing example count expands input context size, which increases inference time and computational cost due to the quadratic complexity of Transformer-based models—particularly problematic for real-time and resource-constrained applications [1, 6]. This also escalates operational costs through increased token usage (higher cloud API fees) and computational demands [6]. Additionally, the quality and selection of examples significantly impact performance [15]; poorly chosen or noisy examples can degrade reasoning and lead to overfitting or failure to generalize on complex tasks [34, 50]. As more examples are added, the risk of including irrelevant or contradictory rationales

amplifies, potentially confusing the model and reducing accuracy [5, 50]. Moreover, the larger context risks exceeding the maximum length, which can lead to truncation or lost information [1], compromising the model's response quality. In long contexts, critical content may be ignored, resulting in a phenomenon known as "lost in the middle," which adversely affects overall performance [24]. Finally, crafting high-quality, task-specific CoT examples is labor-intensive [36, 37], and while many-shot in-context learning (ICL) may improve performance on specific tasks, generalization to new tasks remains limited without careful prompt engineering. GateLens circumvents these issues by achieving optimal performance in a zero-shot setting, relying on logical RA translation rather than using manually designed many-shot examples. By maintaining a compact input footprint, GateLens reduces average total token consumption by approximately 79% compared to the CoT+SC baseline in our production logs, leading to proportionally lower inference latency and API usage costs.

While our lab-based quantitative metrics provide a critical, controlled baseline for comparing architectural choices and measuring robustness against ground truths, we acknowledge that precision and recall on curated queries cannot guarantee absolute correctness in open-ended deployment. Therefore, these lab validations serve primarily to complement—rather than replace—our industrial evaluation. In real-world deployment serving 60-80 users, GateLens demonstrates significant practical value through its user-friendly interface and robust query processing capabilities, with users particularly appreciating the flexibility to input, debug, and refine queries easily. This marks a substantial advancement in industrial data interaction, successfully handling complex and ambiguous queries while providing practical support for faster decision-making in safety-critical software release processes. GateLens demonstrates significant practical advancements by reducing analysis time by over 80% while maintaining high accuracy in test result interpretation, impact assessment, and release candidate evaluation.

Our implementation insights highlight the advantages of focusing on training-free and single-pass agent systems by foregrounding the perception phase in the code generation pipeline. This modular architecture opens opportunities for incorporating emerging LLM capabilities while preserving the system's practical utility in safety-critical industrial applications. A phased deployment program is ongoing and shows that multiple stakeholder groups can be supported, though the evolving roles and analytical needs require continued refinement.

Although instantiated in the automotive release domain, the GateLens architecture can be applied to other domains with comparable analytical

requirements. Its core components—the RA-based intermediate reasoning layer, the separation between query interpretation and code generation, in-scope validation, and zero-shot operation without reliance on few-shot examples—do not depend on automotive-specific properties. Adapting the system to a new domain primarily requires replacing the schema and domain knowledge base, while preserving the architectural reasoning pipeline. Domains characterized by structured tabular data, complex stakeholder queries, and safety- or compliance-critical decision-making (e.g., healthcare analytics, financial auditing, or certification workflows) share these properties, suggesting broader applicability.

Future work will focus on validating this architectural transferability and testing alternative LLM configurations to enhance reliability across other safety-critical industries. By bridging the gap between flexible natural language interaction and rigorous analytical standards, this approach demonstrates the potential for reasoning-enhanced LLMs to transform industrial workflows across a broad spectrum of critical applications.

IV.9 Threats to Validity

The validity of our findings is subject to several potential threats. First, this framework depends on well-defined, static schemas for accurate query decomposition, which fundamentally limits its effectiveness in environments with incomplete or frequently changing schemas. Second, the benchmarks and query scenarios used for evaluation, derived from historical data and real-world queries, may not fully capture the diversity and complexity of potential use cases, which could impact the robustness of the system in broader deployments. Finally, the system's performance depends on the specific LLM configurations used, such as GPT-4o and Llama 3.1 70B, and their ability to interpret and generate RA expressions. Future work will address these limitations through broader domain testing, expanded evaluation scenarios, and alternative LLM configurations.

IV.10 Acknowledgement

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation, and by the Gender Initiative for Excellence (Genie) at Chalmers University of Technology, funded by the Chalmers University Foundation.

Bibliography

[1] R. Agarwal, A. Singh, L. Zhang, B. Bohnet, L. Rosias, S. Chan, B. Zhang, A. Anand, Z. Abbas, A. Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024.

[2] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[3] A. T. P. Boudewijn, A. F. Ferraris, D. Panfilo, V. Cocca, S. Zinutti, K. De Schepper, and C. R. Chauvenet. Privacy measurements in tabular synthetic data: State of the art and future research directions. In *NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI*, 2023.

[4] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[5] Y. Cui, P. He, X. Tang, Q. He, C. Luo, J. Tang, and Y. Xing. A theoretical understanding of chain-of-thought: Coherent reasoning and error-aware demonstration. *arXiv preprint arXiv:2410.16540*, 2024.

[6] Y. Cui, P. He, J. Zeng, H. Liu, X. Tang, Z. Dai, Y. Han, C. Luo, J. Huang, Z. Li, et al. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*, 2025.

[7] H. Dai, B. Wang, X. Wan, B. Dai, S. Yang, A. Nova, P. Yin, M. Phothilimthana, C. Sutton, and D. Schuurmans. UQE: A query engine for unstructured databases. *Advances in Neural Information Processing Systems*, 37:29807–29838, 2024.

[8] H. Dong, Z. Cheng, X. He, M. Zhou, A. Zhou, F. Zhou, A. Liu, S. Han, and D. Zhang. Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks. *arXiv preprint arXiv:2201.09745*, 2022.

[9] X. Dong, C. Zhang, Y. Ge, Y. Mao, Y. Gao, J. Lin, D. Lou, et al. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*, 2023.

[10] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. J. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, C. Faloutsos, et al. Large language models (llms) on tabular data: Prediction, generation, and understanding-a survey. 2024.

[11] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[12] P. Glenn, P. P. Dakle, L. Wang, and P. Raghavan. Blendsql: A scalable dialect for unifying hybrid question answering in relational algebra. *arXiv preprint arXiv:2402.17882*, 2024.

[13] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

[14] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag. TabLLM: Few-shot Classification of Tabular Data with Large Language Models.

[15] X. Huang, L. L. Zhang, K.-T. Cheng, F. Yang, and M. Yang. Fewer is more: Boosting llm reasoning with reinforced context pruning. *arXiv preprint arXiv:2312.08901*, 2023.

[16] Z. Jin and W. Lu. Tab-cot: Zero-shot tabular chain of thought. *arXiv preprint arXiv:2305.17812*, 2023.

[17] A. G. Khoee, Y. Yu, R. Feldt, A. Freimanis, P. A. Rhodin, and D. Parthasarathy. Gonogo: An efficient llm-based multi-agent system for streamlining automotive software release decision-making. In *IFIP International Conference on Testing Software and Systems*, pages 30–45. Springer, 2024.

[18] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

[19] M. Leung and G. Murphy. On automated assistants for software development: the role of llms. *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1737–1741, 2023.

[20] C. Li, J. Liang, A. Zeng, X. Chen, K. Hausman, D. Sadigh, S. Levine, L. Fei-Fei, F. Xia, and B. Ichter. Chain of code: Reasoning with a language model-augmented code emulator. *arXiv preprint arXiv:2312.04474*, 2023.

[21] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. C. Chang, F. Huang, R. Cheng, and Y. Li. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs.

[22] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, and T.-S. Chua. Data-efficient fine-tuning for llm-based recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 365–374, New York, NY, USA, 2024. Association for Computing Machinery.

[23] M. Liu, J. Wang, T. Lin, Q. Ma, Z. Fang, and Y. Wu. An empirical study of the code generation of safety-critical software using llms. *Applied Sciences*, 14:1046, 2024.

[24] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[25] L. P. Manik, Z. Akbar, H. F. Mustika, A. Indrawati, D. S. Rini, A. D. Fefirenta, and T. Djarwaningsih. Out-of-scope intent detection on a knowledge-based chatbot. *International Journal of Intelligent Engineering & Systems*, 14(5), 2021.

[26] N. Marques. Using chatgpt in software requirements engineering: a comprehensive review. *Future Internet*, 16:180, 2024.

[27] E. Miehling, K. N. Ramamurthy, K. R. Varshney, M. Riemer, D. Bouneffouf, J. T. Richards, A. Dhurandhar, E. M. Daly, M. Hind, P. Sattigeri, et al. Agentic ai needs a systems theory. *arXiv preprint arXiv:2503.00237*, 2025.

[28] D. Min, N. Hu, R. Jin, N. Lin, J. Chen, Y. Chen, Y. Li, G. Qi, Y. Li, N. Li, et al. Exploring the impact of table-to-text methods on augmenting

llm-based question answering with domain hybrid data. *arXiv preprint arXiv:2402.12869*, 2024.

[29] R. Mouravieff, B. Piwowarski, and S. Lamprier. Learning relational decomposition of queries for question answering from tables. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10471–10485, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[30] L. Pan, A. Albalak, X. Wang, and W. Y. Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023.

[31] V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, 2024.

[32] C. M. Pham, S. Sun, and M. Iyyer. Suri: Multi-constraint instruction following for long-form text generation. *arXiv preprint arXiv:2406.19371*, 2024.

[33] B. Qiao, L. Li, X. Zhang, S. He, Y. Kang, C. Zhang, F. Yang, H. Dong, J. Zhang, L. Wang, et al. Taskweaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541*, 2023.

[34] C. Qin, A. Zhang, C. Chen, A. Dagar, and W. Ye. In-context learning with iterative demonstration selection. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7441–7455, 2024.

[35] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[36] K. Stechly, K. Valmeekam, and S. Kambhampati. Chain of thoughtlessness? an analysis of cot in planning. *Advances in Neural Information Processing Systems*, 37:29106–29141, 2024.

[37] C.-Y. Tai, Z. Chen, T. Zhang, X. Deng, and H. Sun. Exploring chain-of-thought style prompting for text-to-sql. *arXiv preprint arXiv:2305.14215*, 2023.

[38] M. Turpin, J. Michael, E. Perez, and S. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

[39] B. van Breugel and M. van der Schaar. Why tabular foundation models should be a research priority. *arXiv preprint arXiv:2405.01147*, 2024.

[40] K. VM, H. Warrier, Y. Gupta, et al. Fine tuning llm for enterprise: Practical guidelines and recommendations. *arXiv preprint arXiv:2404.10779*, 2024.

[41] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[42] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[43] Z. Wang, H. Zhang, C.-L. Li, J. M. Eisenschlos, V. Perot, Z. Wang, L. Miculicich, Y. Fujii, J. Shang, C.-Y. Lee, et al. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*, 2024.

[44] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[45] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

[46] J. Ye, M. Du, and G. Wang. DataFrame QA: A Universal LLM Framework on DataFrame Question Answering Without Data Exposure, 2024. Version Number: 1.

[47] T. Zhang, X. Yue, Y. Li, and H. Sun. Tablellama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6024–6044, 2024.

[48] Z. Zhang, Y. Yao, A. Zhang, X. Tang, X. Ma, Z. He, Y. Wang, M. Gerstein, R. Wang, G. Liu, et al. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *arXiv preprint arXiv:2311.11797*, 2023.

[49] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, and B. Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.

[50] Z. Zhou, R. Tao, J. Zhu, Y. Luo, Z. Wang, and B. Han. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances in Neural Information Processing Systems*, 37:123846–123910, 2024.