AI4AIR: A Comprehensive Survey on Large Language Models for AI Research

Xiang Ao, *Member, IEEE,* Junhong Lian, Hanyang Li, Siyi Wang, Yiran Qiao, Yi Qiao, Jiaqi Xu, Qing He, *Member, IEEE,* and Xueqi Cheng, *Fellow, IEEE*

***Abstract*—Language-mediated automation is beginning to complement the human-centered trial-and-error process in AI research. Among current AI tools, large language models (LLMs) have become a central interface for generation, knowledge synthesis, and reasoning in research workflows. While LLMs are now widely used to support general scientific workflows such as literature review and scientific writing, their specific roles and deeper contributions to the core lifecycle of AI research itself remain insufficiently explored in a systematic manner. To bridge this gap, this survey introduces AI4AIR (short for AI for AI Research), which comprehensively reviews LLMs as pivotal components within machine learning research pipelines. We construct a structured two-dimensional taxonomy. One axis spans major research domains including natural language processing, computer vision, data mining, and general machine learning. The other follows the research pipeline stages, encompassing data engineering, model design and optimization, model evaluation, and the cross-stage closed-loop automation that connects them. Within this framework, we identify five recurring roles of LLMs, namely annotator, synthesizer, optimizer, evaluator, and orchestrator, through which LLMs contribute to AI research workflows. We further discuss bottlenecks such as contamination, hallucination, and reliability under feedback-driven use, and outline future directions for improving both the efficiency and the reliability of AI research and discovery.**

***Index Terms*—Large language models, artificial intelligence research, machine learning pipeline, AI for AI research.**

I. Introduction

THE human-centered paradigm of scientific and technical research has evolved over centuries into a structured workflow: from problem identification, literature review, and hypothesis formulation to experimental design, empirical validation, and the dissemination of discoveries or inventions through peer-reviewed publications. With the burgeoning development of large language models (LLMs) in recent years, artificial intelligence (AI) researchers have increasingly incorporated them into the established paradigm.

The work was supported in part by the National Natural Science Foundation of China under Grant U2436209, Grant 62576333, and Grant 62406307, in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDB0680201, in part by the Beijing Natural Science Foundation under Grant F251001, and in part by the Innovation Funding of ICT, CAS under Grant E461060. *(Corresponding authors: Xiang Ao and Xueqi Cheng.)*

Xiang Ao, Junhong Lian, Siyi Wang, Yiran Qiao, Yi Qiao, Jiaqi Xu, Qing He, and Xueqi Cheng are with the State Key Laboratory of AI Safety, Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing 100190, China. They are also with the University of Chinese Academy of Sciences, Beijing 100049, China. (e-mail: aoxiang@ict.ac.cn; cxq@ict.ac.cn).

Hanyang Li is with the National University of Singapore (NUS), Singapore. He was with the State Key Laboratory of AI Safety, ICT, CAS, as a research intern during this work.

The past two years, in particular, have witnessed a surge in research efforts, including leveraging LLMs for literature review [1], formulating research problems [2] and even assisting in theorem proving [3] or experimental design [4]. These advancements have accelerated scientific discovery and innovation across a wide range of disciplines. Remarkably, AI-driven scientific breakthroughs were recognized by the 2024 Nobel Prizes in Physics and Chemistry [5]. More comprehensive surveys that summarize AI-driven workflows can be found in [6], [7].

Beyond general scientific workflows, LLMs are reshaping AI research itself. That is, rather than being limited to peripheral support, they are making deeper contributions to the core of AI research, driven by their generative capabilities [8], broad pretrained knowledge [9], and emerging reasoning abilities [10]. For instance, LLMs have been used to generate synthetic data for training machine learning (ML) models [11]–[13]. They are also widely employed to annotate data in low-resource scenarios [14], [15]. Furthermore, representative studies have directly employed LLMs as predictors [16], [17], serving as alternatives to traditional AI models. Advanced LLMs, such as GPT-4o [18], have been leveraged as expert evaluators to assess the performance of developed AI models [19]–[21]. These use cases collectively demonstrate a broader trend of LLMs becoming integral components across the data, model, and evaluation stages of AI research.

Based on these observations, this survey aims to formulate a systematic perspective on the following central research question: ***How do LLMs contribute to AI research?*** To answer this question, we comprehensively review the literature in recent years and construct a taxonomy categorizing the diverse roles of LLMs in AI research. To maintain a focused scope, we concentrate on the research of designing ML models, which currently represents a mainstream and pivotal direction within the AI community.

A typical ML pipeline can be divided into several stages, including *data engineering*, *model design and optimization*, and *model evaluation*. Beyond these core stages, we further examine how LLMs support automated workflow orchestration to close the AI research loop. Following this structure, we summarize recurring roles of LLMs across the ML-centered AIR pipeline, identify current bottlenecks, and outline future directions. An overview of this iterative ML-centered AI research lifecycle is shown in Fig. 1. This framing motivates the title of our survey "AI4AIR", where the leading AI denotes AI tools and AIR denotes AI research.

To the best of our knowledge, no prior survey has system-

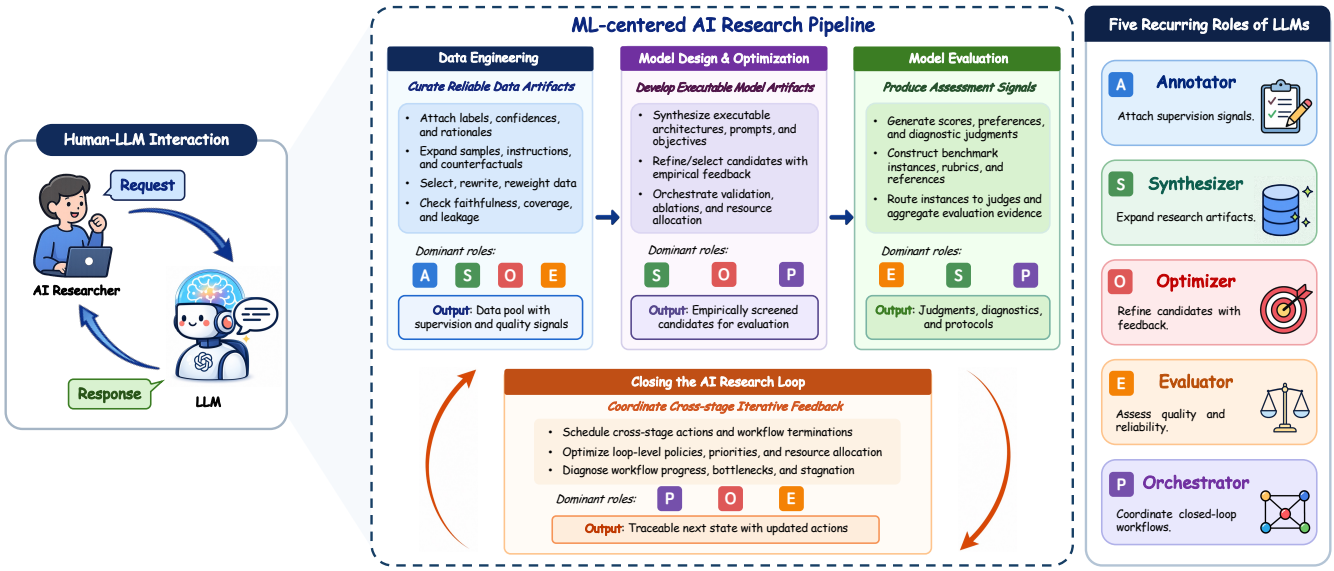


Fig. 1. Illustration of the multi-faceted roles of LLMs across the iterative ML-centered AI research lifecycle.

atically examined the intersection of the multi-faceted roles of LLMs within the AI research workflow itself. Existing reviews on specific AI sub-domains, including computer vision [22]–[24], recommender systems [9], [25], [26], and graph learning [27]–[29], primarily investigate how LLMs enhance domain-specific task performance or enable specialized model construction. Meanwhile, surveys on AutoML [30], [31] remain centered on conventional numerical optimization techniques, overlooking the semantic reasoning capabilities driven by LLMs. A more recent review [32] examines LLM-driven ML workflows in a generic AutoML setting, without distinguishing the roles of LLMs across different stages of the workflow. Furthermore, technical reviews [33]–[35] typically focus on specific enabling techniques, such as retrieval-augmented generation (RAG) or parameter-efficient fine-tuning (PEFT), dissecting their architectural, training, or inference-level designs rather than the AI research workflow itself. Similarly, in the broader realm of AI for research (AI4R), existing studies [6], [7] primarily focus on assisting the preliminary research stages, such as hypothesis generation and manuscript preparation. This scope differs substantially from that of our AI4AIR framework. These AI4R surveys emphasize the potential of LLMs in ideation or productivity enhancement within the general publication process, yet they do not delve into the core ML pipeline. In summary, prior works either restrict their scope to specific AI sub-domains or focus on general scientific utility. Consequently, the community still lacks a holistic view of the multi-faceted roles of LLMs throughout the AI research lifecycle. Table I presents a structured comparison between our survey and representative prior surveys, highlighting our distinct scope and coverage. We envision this survey serving as a comprehensive guide for the broader research community to navigate this rapidly evolving field.

The main contributions of this survey are summarized as follows.

- To the best of our knowledge, this is the first systematic survey that summarizes the utility of LLMs in ML-centered AI research, providing a structured perspective on their roles across the AI research lifecycle.
- We construct a taxonomy based on LLMs' roles in machine learning model design and development, systematically categorizing existing related works.
- Based on our review, we identify the bottlenecks of current research and discuss possible future directions for leveraging advanced AI tools to improve both the efficiency and effectiveness of AI research and discovery.

TABLE I
COMPARISON BETWEEN OUR WORK AND EXISTING SURVEYS, WHERE ●, ◐, AND ○ RESPECTIVELY DENOTE FULLY INVOLVED, PARTIALLY INVOLVED, AND NOT INVOLVED IN THE CORRESPONDING ASPECT.

Survey	Scope	Workflow View	ML Pipeline	Multi-domain	LLM Roles
Wang et al. [6]	AI for Science	●	○	◐	○
Chen et al. [7]	AI for Research	●	◐	◐	○
Awais et al. [22]	Computer Vision	○	◐	○	○
Lin et al. [25]	Recommender Systems	○	◐	○	◐
Jin et al. [28]	Graph Learning	○	◐	○	◐
Hutter et al. [30]	AutoML	◐	●	○	○
Gao et al. [33]	RAG Techniques	○	◐	○	○
Gu et al. [32]	LLMs for ML Workflows	●	●	○	○
AI4AIR (Ours)	**LLMs for AI Research**	●	●	●	●

The remainder of this survey is organized as follows. Section II defines the scope and presents the proposed taxonomy of AI tools in ML-centered AI research. Section III reviews recent advances in LLM use across major AI research sub-domains, including natural language processing (NLP), computer vision (CV), data mining (DM), and general machine learning (GML), organized around core tasks in each domain. Section IV reinterprets the roles of LLMs through the lens of the machine learning development pipeline. Section V discusses open challenges and future directions for AI4AIR, followed by the conclusion in Section VI.

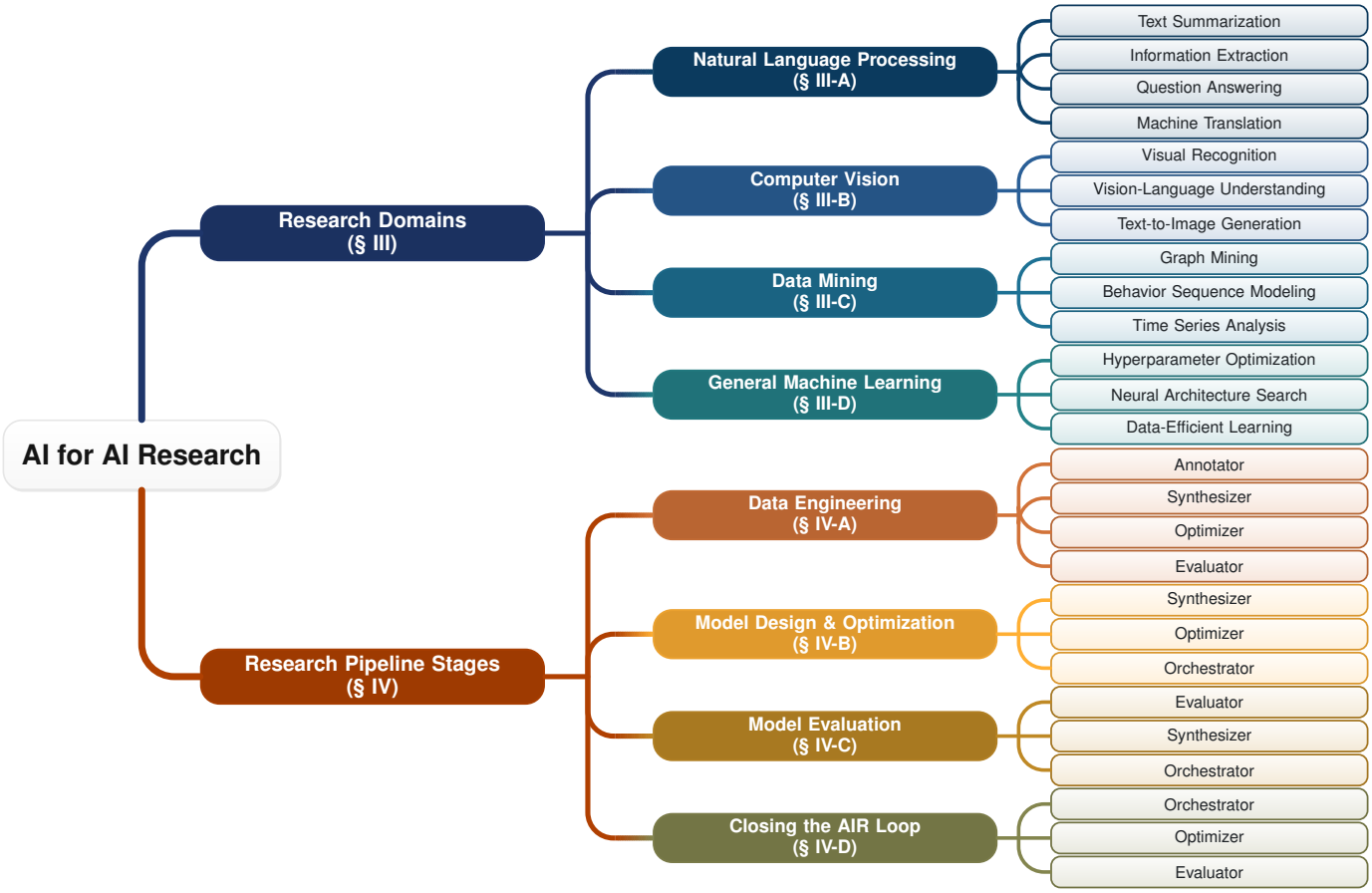


Fig. 2. A two-dimensional taxonomy of AI4AIR. The domain-oriented view summarizes representative AI sub-domains and tasks, while the pipeline-oriented view maps recurring LLM roles to ML-centered pipeline stages.

II. Overview of AI for AI Research

In this section, we clarify the scope of AI4AIR and present the taxonomy used to organize the surveyed literature. We then summarize our literature collection and inclusion criteria.

A. Scope and Definition

AI research spans a broad range of sub-disciplines, from hardware acceleration [36] and neuromorphic computing [37], [38] to symbolic reasoning [39] and control systems [40]. In this survey, we explicitly narrow our scope to *ML-centered AI research*, defined as the end-to-end process of developing ML models through a canonical pipeline, including *data engineering*, *model design and optimization*, and *model evaluation*. We focus on this setting because it represents a prevailing paradigm across major AI domains, and it provides a consistent, workflow-level basis for analyzing how LLMs intervene in research practice.

B. Why ML-centered AI Research

One fundamental difference between the AI research discussed in this survey and AI applications in other natural sciences, such as biology [41], [42] or chemistry [43], [44], lies in the execution environment. Scientific discovery in the physical world often faces a "wet-lab bottleneck," where hypotheses must be validated through time-consuming physical experiments. In contrast, ML-centered AI research largely operates in the digital realm. The core assets of AI research, including datasets, model implementations, training configurations, and evaluation metrics, are inherently machine-readable text or numeric data. This characteristic enables LLMs to perceive the research state, generate executable code, and receive immediate feedback, thereby forming a relatively frictionless digital closed-loop for iterative automation.

C. Taxonomy and Organization

The standardized ML pipeline and the language-based reasoning capabilities of LLMs have catalyzed a paradigm shift, necessitating a systematic framework to navigate this complex landscape. However, the literature on LLM-empowered AI research is rapidly expanding yet scattered across diverse problem settings. To facilitate a coherent reading path, we propose a hierarchical taxonomy in Fig. 2 as the organizational backbone of this survey. Guided by this taxonomy, we provide two complementary entry points for navigating prior work, namely a *domain-oriented* view and a *pipeline-oriented* view.

Following the domain-oriented view, Section III reviews representative advances across major AI sub-domains (e.g., NLP, CV, DM, and GML), highlighting the versatility and adaptability of these tools across distinct research topics. Within each domain, we organize the discussion around representative task categories to support fine-grained comparisons

across research settings. Complementarily, Section IV reinterprets the literature through the ML development pipeline to highlight the recurring roles of LLMs at each stage. The first three stages are artifact-centric, operating on the data, model, and evaluation artifacts produced at each stage. The fourth stage further examines cross-stage workflows, where LLMs decompose high-level objectives, coordinate tools and stage-level procedures, and leverage feedback to close the AIR loop. Building on these two perspectives, we further discuss key challenges and future directions for AI4AIR.

D. Literature Collection and Screening

We followed a systematic retrieval-and-screening pipeline to ensure both the coverage and relevance of the surveyed literature. The systematic retrieval covered a five-year window ending on June 30, 2025. Specifically, we queried the Semantic Scholar API and Google Scholar to retrieve candidate papers published within this period. Our queries combined keywords related to *large language models* (e.g., "large language model", "LLMs", "LLM") with those describing *research roles* (e.g., "data augmentation", "annotation", "model design", "optimization", "evaluation"). To further improve recall, we additionally collected the accepted-paper lists of top-tier computer science conferences from DBLP for the most recent three years, and performed title-based keyword matching to retrieve potentially relevant works. After deduplication, we obtained $1,440$ candidate papers as of June 30, 2025.

We then conducted a two-stage screening process. First, we designed task prompts and few-shot examples to enable an LLM to read each paper's title and abstract for coarse filtering, removing non-English papers and papers that do not involve LLMs as part of the research workflow. To prioritize impactful and widely discussed work, we used citation counts at the time of collection as an auxiliary signal for ranking and triage. This stage removed 778 papers that clearly fell outside our scope. Second, we manually inspected the remaining manuscripts and excluded those without substantive relevance to AI4AIR. A paper was included only if LLMs constitute an explicit and non-trivial component of the ML research workflow. Finally, based on our domain expertise, we supplemented a small number of relevant studies that might be missed by keyword-based retrieval to avoid overlooking important contributions. When multiple versions of the same work exist, we cite the most complete and archival version.

Since AI4AIR evolves rapidly, we further monitored major venues and preprint repositories beyond the systematic cutoff date. We selectively incorporated representative works published or released after June 30, 2025 based on domain expertise, provided they satisfied the same inclusion criteria as the core corpus. These additions do not affect the reported corpus statistics, as all quantitative counts are computed exclusively from papers retrieved within the original collection window.

III. LLMs Across AIR Sub-domains

We first take a domain-oriented view to examine where LLMs have been incorporated into research workflows across major AI sub-domains. While LLMs are general-purpose tools, their impact varies with data modalities and core challenges. We therefore review key advances in NLP, CV, DM, and GML, organizing each domain by representative research tasks to show where and how LLMs are incorporated into task-specific workflows.

A. LLMs for Natural Language Processing

LLMs originate from the pretrained language modeling paradigm, making NLP the earliest and most extensively studied setting for their methodological development and empirical validation. Early work primarily treated LLMs as unified instruction-following predictors that transfer across downstream tasks via prompting. Subsequent studies showed that LLMs can also generate instruction data for self-improvement, thereby reducing dependence on continuous human supervision [11], [45]. More recently, attention has shifted from isolated task performance to end-to-end workflows, integrating LLMs into data construction, evaluation, and iterative training loops. This turn reframes LLMs as workflow components for addressing supervision cost, evaluation reliability, and feedback-driven improvement in NLP. Broadly, NLP tasks can be grouped into natural language understanding (NLU) and natural language generation (NLG). Accordingly, we focus on four representative tasks spanning both categories to illustrate the breadth of LLM utility.

1) ***Text Summarization****:* Text summarization compresses a source document under length constraints while balancing coverage and factual consistency. The primary challenges of this task fall into two categories. One concerns the difficulty of identifying and localizing factual inconsistencies. The other relates to the cost and stability of obtaining high-quality training and evaluation signals. For faithfulness, prior work decomposes summaries into atomic claims and verifies them against external evidence, transforming faithfulness assessment into identifiable error types with reusable diagnostic interfaces [46]. To strengthen evaluation signals, recent studies use LLMs to build reference-free evaluation protocols with explicit rubrics and step-wise verification, yielding judgments that correlate well with expert assessments [20], [21]. Evaluation stability can be further improved via multi-agent peer review and debate, which reduce single-evaluator bias and increase consistency [47]. On the training side, LLMs can generate reference summaries and ranking signals for objectives such as contrastive learning, enabling smaller models to improve under controlled settings and, in some cases, surpass supervised baselines [48].

2) ***Information Extraction****:* Information extraction maps unstructured text to structured representations, such as entities, relations, or events, to support downstream retrieval and analysis. A persistent challenge lies in acquiring schema-consistent supervision under evolving label definitions and domain shifts. Expert annotations are expensive, while crowdsourced annotations suffer from inconsistent guideline interpretation. LLMs can generate task-specific labels in zero-shot or few-shot settings, providing lower-cost bootstrapping for classification and sequence labeling [14]. However, such synthetic annotations may introduce noise and systematic bias. Robust

learning methods, including reweighting and subset selection, can prioritize more reliable instances and improve training stability in noisy synthetic supervision [49]. For dialogue-driven extraction, controllable simulation couples in-context learning with structured annotation to synthesize multi-turn interactions, while consistency checks align utterances with labels to cover complex interactive settings [50]. Prompt-based diversification further expands the training distribution through structured perturbations and paraphrases [51].

*3) **Text Question Answering**:* Text question answering requires generating correct, relevant, and evidence-supported answers to natural language queries. Modern systems increasingly adopt retrieval augmentation, retrieving supporting evidence from external corpora or knowledge bases before generation. Key challenges include ensuring evidence alignment and controllable reasoning, as well as scaling benchmark maintenance and automated evaluation. Static benchmarks can become outdated and may be exposed to contamination as training corpora expand. LLMs can generate questions and judge answers for dynamic capability assessment, probing finer-grained skills beyond fixed references and reducing drift caused by delayed benchmark updates [52]. For retrieval-augmented generation, multi-dimensional diagnostics assess faithfulness, context relevance, and recall. These diagnostics reveal mismatches between retrieval and generation while reducing reliance on gold references [53], [54]. For knowledge base question answering, LLMs translate executable queries into natural language questions to synthesize training pairs. Combined with execution-guided self-training, this approach filters pseudo-labels and bridges the distribution gap between synthetic and real-world user queries [55].

*4) **Machine Translation**:* Machine translation maps source-language sentences into target-language expressions while balancing semantics, style, and domain conventions. A prominent limitation of this task is the scarcity of high-quality parallel corpora, which is particularly acute for low-resource language pairs. LLMs can generate or rewrite pseudo-parallel data, with filtering and consistency constraints to control noise. The resulting data supports distillation or incremental training, improving quality and data efficiency in low-resource settings [56]–[58]. Evaluation quality also affects iteration efficiency. Traditional n-gram metrics struggle to capture nuances in fluency and style, while scalable expert evaluation across multiple language pairs is impractical. Recent studies propose evaluation protocols based on explicit scales to enhance reusability. These methods report high correlation with expert judgments under several prompting strategies, providing reusable signals for quality tracking [59]. Alternatively, probabilistic scoring frameworks estimate quality by comparing generation probabilities against custom definitions, thus reducing reliance on task-specific training and complementing traditional metrics [19].

B. LLMs for Computer Vision

Computer vision research focuses on developing models that analyze images and videos for recognition, understanding, and generation. Unlike natural language tasks that operate over discrete symbols, visual tasks rely on complex spatial structures, instance relations, and fine-grained attributes. These characteristics lead to persistent bottlenecks in semantic specification and controllable generation. Since text-only LLMs cannot directly perceive visual content, they typically influence CV through language-side operations. Multimodal large language models (MLLMs) further integrate visual inputs, enabling alignment and reasoning grounded in visual evidence and broadening LLM applications in vision. In this subsection, we categorize recent advances into visual recognition, vision-language understanding, and text-to-image generation, highlighting how LLMs mitigate task-specific limitations and transform CV research workflows.

*1) **Visual Recognition**:* Visual recognition aims to classify, localize, or segment objects within images or videos. Despite strong closed-set performance, challenges remain in open-world coverage, long-tail generalization, and compositional reasoning. Traditional closed-set training relies on short labels and fixed taxonomies, which can leave class semantics underspecified and destabilize decision boundaries under domain shift. Meanwhile, tail classes often suffer from weak or biased representations under limited supervision. To address limited label semantics, recent studies leverage LLMs to generate fine-grained definitions and attribute phrases, enriching textual supervision and reducing ambiguity from short labels [60]–[62]. In pixel-level tasks, LLM-generated attribute descriptions provide part-level cues that improve fine-grained segmentation [63]. For compositional visual queries, LLMs can decompose high-level instructions into callable visual sub-processes, improving the transparency of tool-augmented reasoning [64]. These uses mainly enrich semantic supervision for recognition, while controllable generation improves coverage for rare or underspecified categories. Under data scarcity, LLM-guided controllable generation can synthesize training samples or visual descriptions for rare classes, improving coverage of tail categories [65], [66].

*2) **Vision-Language Understanding**:* Vision-language understanding aligns visual evidence with natural language for tasks including visual question answering (VQA) and image captioning. Long-standing bottlenecks stem primarily from the scarcity and bias of high-quality multimodal supervision signals, as well as the resource overhead and stability risks of large-scale cross-modal alignment training. Crowd-sourced annotations are expensive and prone to linguistic bias, which narrows the training distribution and hurts cross-domain generalization. To mitigate data bottlenecks, existing research proposes using LLMs to automatically rewrite readily available image captions into question-answer pairs, with consistency filtering to remove noise and establish scalable supervision construction [67]. At the model level, the focus lies on leveraging the reasoning capabilities of LLMs. Mainstream approaches prioritize cross-modal alignment between strong visual encoders and LLMs by training only lightweight bridging modules [68], [69]. Under this framework, visual information is injected into the LLM as fixed tokens, enabling few-shot adaptation through in-context learning rather than task-specific fine-tuning. For evaluation, equivalence judgment of open-ended answers has long relied on manual review,

limiting rapid iteration. Recent studies introduce LLMs to automatically evaluate open-ended video question answering, annotating instance-level evaluation rules for individual video-question pairs and refining them through adversarial mechanisms to achieve accuracy close to human experts [70].

*3) **Text-to-Image Generation**:* Text-to-image generation aims to synthesize high-fidelity images from natural language instructions. Key challenges include semantic alignment for complex prompts, spatial structure control, and interpretable evaluation. Traditional text encoders often lack capacity for dense prompts with combinatorial relationships, leading to attribute mismatches. To improve semantic alignment, recent studies introduce LLMs as text encoders, leveraging enhanced language representations for complex instructions [71]. Alternatively, lightweight adapters inject LLM semantic features into frozen diffusion backbones, with entity-attribute consistency constraints reducing mismatches in multi-object scenarios [72], [73]. On the input side, prompt rewriting and semantic completion reduce reliance on manual prompt engineering, making concise user inputs more reliably followed [74], [75]. To enhance structural control, LLMs are used to convert complex instructions into structured layouts or region-wise generation plans [76], [77]. Recent studies adopt step-wise generation with local consistency verification to further stabilize adherence to counting and spatial constraints [78]–[80]. For evaluation, LLMs generate interpretable question-answering probes to assess alignment and faithfulness. Recent studies show that VQA-based verification provides fine-grained and scalable signals for candidate ranking and error diagnosis, thereby improving iteration efficiency [81], [82].

C. LLMs for Data Mining

Data mining research spans diverse structured data, including graph structures, user interaction sequences, and time series signals. Unlike perception and language tasks that rely on explicit human annotations, supervision in data mining often comes from implicit feedback and weak signals. This setting introduces significant challenges related to noise and domain-specific constraints. In this context, LLMs are increasingly integrated into mining pipelines to connect unstructured context with structured patterns. We organize this subsection around three representative data types, covering graph mining, behavior sequence modeling, and time series analysis.

*1) **Graph Mining**:* Graph mining entails reasoning over non-Euclidean relational structures for tasks such as node classification and link prediction. A persistent challenge is the separation between topology and semantics. Graph structures capture connectivity but often lack sufficient attribute specification, which limits generalization in open-world settings. One line of work bridges this gap by serializing local neighborhoods or subgraphs into natural language descriptions [83], [84]. While transforming structural reasoning into linguistic inference, these methods expose capability boundaries when handling complex topology. To mitigate reasoning instability, structured mappings organize multiscale relations into hierarchical descriptions, thereby improving consistency in few-shot scenarios [85], [86]. Alternative approaches align graph encoders with LLMs through learnable projections [87], [88]. These methods project structural features into the language space to preserve representation efficiency while enabling interpretable outputs. In label-scarce settings, LLMs generate weak supervision signals such as pseudo-labels or semantic rationales. Downstream graph models then expand coverage through structural propagation [89], [90]. Furthermore, LLMs facilitate noise detection by identifying semantically inconsistent edges, which enhances training robustness [91]. For tasks requiring exact topological computation, recent systems delegate graph operations to external tools while using LLMs for orchestration and explanation [92].

*2) **Behavior Sequence Modeling**:* Behavior sequence modeling captures preference structure and interest evolution from interaction histories to support recommendation and personalized retrieval. Primary challenges stem from sparse and noisy implicit feedback, exposure bias, and cold-start issues in identifier-based representations. LLM-derived user and item semantics can mitigate feature scarcity by injecting world knowledge into identifier-based representations, improving transfer under limited behavioral evidence [93]–[95]. Furthermore, LLM-based subset selection and instruction-style synthesis also support data-centric iteration by reducing manual curation [96], [97]. Personalized retrieval further suffers from underspecified and context-dependent intents, especially in conversational settings. Query rewriting and semantic expansion with LLMs convert short or contextualized inputs into self-contained queries, improving candidate recall [98]–[100]. Given that generative rewriting may introduce drift, constrained alignment with offline feedback has been explored to stabilize rewritten intents and downstream retrieval objectives [101]. Synthetic query-document or query-item pairs derived from limited seeds or item content enable weakly supervised retriever adaptation at scale and reduce supervision cost [102], [103]. LLM-based reranking and explanation generation operate within retrieved candidate sets, improving discrimination while limiting hallucination risk and controlling latency [104], [105]. For evaluation, LLM-based judging has been adopted to automate relevance assessment at scale, while robustness protocols and calibration remain important for reliable model comparison [106].

*3) **Time Series Analysis**:* Time series analysis involves forecasting, anomaly detection, and diagnosis for continuously varying signals. We focus on uses where LLMs construct supervision, add external context, or support diagnostic evaluation for time-series models. Key bottlenecks arise from the semantic gap between numerical observations and high-level contexts, as well as the difficulty of obtaining supervision that transfers across domains. Addressing data scarcity, recent studies use LLMs to synthesize realistic signal segments in few-shot settings and to construct instruction data that aligns time series with text for downstream understanding and reasoning [107], [108]. Complementary work prompts LLMs to generate pseudo-labels and natural-language rationales from unlabeled sensor streams, which can be used as supervision for detection models [109]. Beyond generation, researchers further exploit LLM-based judgments to train lightweight scorers for data selection and cleaning, improving robustness

under heterogeneous sources [110]. To incorporate external context, recent work applies LLMs to social media streams to extract sentiment and event signals, and then fuses them with numerical predictors for tasks such as stock forecasting [111]. For anomaly diagnosis, cross-modal conditioning frameworks have been proposed to generate evidence-grounded explanations from numerical inputs with frozen LLMs [112]. Other systems combine retrieval tools with LLM reasoning over logs and traces to localize root causes and to produce actionable summaries [113]. Emerging benchmarks evaluate reasoning over heterogeneous telemetry that includes time series, logs, and traces, which supports standardized evaluation for root cause analysis [114]. In addition, studies use LLMs to translate signal statistics into structured narratives for expert-facing report generation in finance and healthcare [115], [116].

D. LLMs for General Machine Learning

Beyond modality-specific domains such as NLP, CV, and DM, many machine learning workflow components are largely modality-agnostic and are primarily shaped by development-loop complexity rather than input format. These settings are challenging because the decision space is often discrete or mixed-type, while reliable feedback is expensive since most candidates require non-trivial training or validation. Recent optimization loops increasingly use LLMs to propose, revise, or explain candidate decisions. We organize this subsection around three representative tasks that frequently dominate practical iteration cost: hyperparameter optimization, neural architecture search, and data-efficient learning.

1) ***Hyperparameter Optimization****:* Hyperparameter optimization selects training and regularization configurations under a fixed budget to maximize validation performance. It usually involves black-box search over mixed-type variables under wall-clock, memory, and convergence constraints. To reduce early search cost, one line of work converts task descriptions, dataset characteristics, and resource constraints into structured textual inputs. LLMs then use this context to propose plausible initial configurations, reducing wasteful trials and moving exploration beyond blind sampling [117], [118]. During iteration, another line of work normalizes historical configurations, metrics, failure patterns, and key logs into structured records. LLMs then extract patterns across rounds, so later candidates can explicitly avoid known invalid regions and focus sooner on high-potential ranges [119], [120]. Under sparse observations, LLMs have also been embedded into sequential black-box optimization loops. They condition candidate generation on previous solutions, observed scores, and textual constraints, improving early-stage sampling efficiency [121], [122]. A similar text-conditioned search paradigm has also been extended to the construction of executable objectives. By encoding high-level task goals and rollout feedback as structured context, LLMs iteratively generate and revise reward functions, reducing the cost of manual objective engineering in reinforcement learning workflows [123].

2) ***Neural Architecture Search****:* Neural architecture search (NAS) aims to identify network structures that balance predictive performance, computational cost, and deployment constraints. Unlike hyperparameter tuning, the central challenge of NAS is that the structural space is highly combinatorial and not continuous. Naive generation can yield invalid structures, repeated motifs, or candidates that violate training constraints, while reliable ranking often requires costly training or surrogate evaluation. To improve exploration quality, one category of studies converts architectures from graph or module representations into serializable discrete symbol sequences, or builds unified encoding formats across domains. LLMs then describe, generate, and parse candidate structures in a consistent manner, which reduces invalid sampling and expands coverage [124], [125]. On the feasibility side, some work explicitly encodes operator sets, dimensionality constraints, and computational budgets into the input specification and couples the LLM output with parsing and validation steps, so that generated structures are more likely to satisfy trainability constraints and avoid early invalid trials [126]. To lessen dependence on expensive training feedback, recent studies introduce low-cost structural quality signals and performance prediction information, combining them with LLMs to perform candidate pre-screening and early ranking. These signals can reduce the number of expensive evaluations and guide search under tighter budgets [127], [128]. In addition, code-level controlled rewriting and population-based selection mechanisms are employed to maintain candidate diversity and suppress mode collapse, allowing the search process to expand the explorable range while preserving structural validity and result reproducibility [129]–[131].

3) ***Data-Efficient Learning****:* Data-efficient learning focuses on preserving model generalization and robustness when labels are scarce, class distributions are long-tailed, supervision is noisy, or training budgets are limited. The key difficulty is that data selection, augmentation, and reweighting strategies often depend on manual heuristics, and evaluating each strategy incurs additional training cost. To make data construction more adaptive, recent studies encode model feedback, error distributions, and candidate pools as structured text inputs. LLMs then iteratively select data subsets with higher utility, while also supporting nondifferentiable metrics as selection signals [132]–[135]. In imbalanced scenarios such as long-tailed recognition, research further employs LLMs to generate semantically faithful augmented samples targeted at underrepresented classes, improving the effective coverage and model robustness for tail categories [136]. Beyond sample-level strategies, injecting task-relevant priors from natural language metadata provides another way to reduce sample complexity. Related work uses LLMs to transform variable names and descriptions into actionable inductive biases, which regularize learning in low-data regimes and support more interpretable feature selection [137], [138]. In extreme settings with only a few examples or high-level requirement descriptions, studies further map task requirements to executable training configurations through LLMs, which supports rapid model adaptation under limited budgets [139].

E. Summary and Discussion

The preceding subsections suggest two recurring patterns across AI sub-domains. First, LLM use is shaped by how

readily domain artifacts can be represented through language-mediated interfaces. In NLP, these interfaces are already present in textual inputs, labels, references, answers, and evaluation criteria. In CV and DM, they are constructed through descriptions, prompts, layouts, graph verbalization, query rewriting, semantic enrichment, and external event narratives. In GML, the focus moves from domain data to development decisions, where configurations, architectures, objectives, task priors, and data-selection strategies become the main objects of generation and refinement.

Second, these artifacts differ in validation cost. Labels, descriptors, synthetic examples, textual transformations, and quality signals can often be checked through inspection, filtering, semantic consistency, or external evidence. Executable specifications and development decisions instead require empirical validation through training, search, ablation, or downstream evaluation. As validation cost increases, LLM use tends to move from semantic artifact construction toward candidate generation and refinement under external feedback.

Taken together, recurring LLM uses across AI sub-domains are shaped by artifact form, accessibility, and verifiability. LLMs can help turn implicit research knowledge into operational artifacts, such as reusable supervision, semantic context, scalable judgment signals, and testable candidates. Their reliability, however, depends on grounding in task-specific rules, external evidence, executable feedback, or validation metrics.

IV. Reimagining the AIR Pipeline with LLMs

Section III reviewed how LLMs enter AI research workflows across major sub-domains. Although these applications differ in domain-specific tasks and research artifacts, they reveal recurring functional roles that are not fully captured by a domain-oriented view. In this section, we shift from a domain-oriented view to a pipeline-oriented view, focusing on how recurring LLM roles operate within and across different stages of AI research. We summarize five recurring roles, namely annotator, synthesizer, optimizer, evaluator, and orchestrator. Rather than mapping these roles to stages one-to-one, we characterize each stage by its dominant role combinations. When a study combines multiple functions, we assign it to the role corresponding to the primary artifact adopted by the workflow. This stage-role view organizes the literature into four stages: data engineering, model design and optimization, model evaluation, and AIR-loop closing.

For clarity, Table II summarizes the notation used in this section. We write X_s^ϕ for a stage-specific LLM role. Here, $X \in \{A, S, O, E, P\}$ denotes annotator, synthesizer, optimizer, evaluator, or orchestrator, and $s \in \{\mathrm{D}, \mathrm{M}, \mathrm{V}, \mathrm{L}\}$ indexes the four pipeline stages. The superscript ϕ indicates the LLM parameters, while the stage subscript specifies the artifact space on which the role acts. For example, S_{D}^ϕ, S_{M}^ϕ, and S_{V}^ϕ represent synthesizers operating on data, model-development, and evaluation artifacts, respectively. By contrast, $\mathcal{T}_s^\phi$ denotes a stage- or loop-level state transformation that integrates role-level LLM outputs, rather than serving as an additional LLM role. Its state space remains stage-specific, with the data state represented explicitly and the other states kept abstract due to their heterogeneous artifacts. We use ρ_s and $\mathcal{H}_s$ for stage-level requirements and feedback, ρ_X for role-specific requirements, and $\mathcal{C}_\bullet$, $\mathcal{F}_\bullet$, and $\mathcal{R}_\bullet$ for role-local context, feedback, and reference or probing information. In particular, $\mathcal{H}_s$ refers to feedback that affects the stage- or loop-level state transformation, whereas $\mathcal{F}_\bullet$ refers to feedback consumed by a specific role operator.

TABLE II
Summary of Notations for LLM Roles in the AIR Pipeline.

Notation	Meaning
ϕ	LLM parameters.
$A^\phi, S^\phi, O^\phi, E^\phi, P^\phi$	Role operators (annotator, synthesizer, optimizer, evaluator, orchestrator).
$\mathrm{D}, \mathrm{M}, \mathrm{V}, \mathrm{L}$	Pipeline stages (data, model, evaluation, AIR-loop closing).
X_s^ϕ	A role X acting at stage s.
$\mathcal{T}_s^\phi$	Stage-/loop-level state transformation integrating role-level LLM outputs.
$\rho_\bullet, \mathcal{C}_\bullet, \mathcal{R}_\bullet$	Requirement, context, and reference signals.
$\mathcal{H}_s, \mathcal{F}_\bullet$	Stage-level feedback and role-local feedback signals, respectively.

A. LLMs for Data Engineering

Data engineering is the data-facing stage of the ML-centered AIR pipeline. Here, LLMs work with samples and their auxiliary fields, including labels, rationales, instructions, queries, metadata, and quality or utility signals. The goal is to make the initial data pool more useful for training, validation, or evaluation, while improving the information content and reliability of the resulting artifacts.

Let $\mathcal{D}_{\mathrm{raw}} = \{d_i\}_{i=1}^n$ be the initial data pool, where each d_i is a learnable data artifact. The data-stage transformation is then abstracted as

$$\mathcal{D}_{\mathrm{eng}} = \mathcal{T}_{\mathrm{D}}^\phi(\mathcal{D}_{\mathrm{raw}}, \rho_{\mathrm{D}}, \mathcal{H}_{\mathrm{D}}). \tag{1}$$

The term ρ_{D} specifies data-stage requirements, and $\mathcal{H}_{\mathrm{D}}$ collects stage-level feedback such as data-quality signals, human review, or downstream utility information. Here, $\mathcal{D}_{\mathrm{eng}}$ is the resulting data-stage state, represented as $\{(d, \eta_d) \mid d \in \mathcal{D}_{\mathrm{adp}}\}$.

For each adopted artifact d, the optional field η_d stores LLM-produced metadata, such as labels, rationales, confidence scores, quality indicators, or selection decisions. When no such signal is generated for an artifact, we set $\eta_d = \emptyset$. The adopted set may include raw, revised, synthesized, selected, or reweighted artifacts after filtering. These outcomes arise from four recurring LLM operations: supervision attachment, artifact expansion, pool-level optimization, and reliability assessment, corresponding to the four roles below.

*1) **Annotator**:* Annotation attaches supervision to an existing data artifact rather than creating a new one. The attached signal may be a class label, pseudo-label, preference label, semantic attribute, structured tag, or rationale. This role is most useful when labeling guidelines can be stated in natural language and human annotation is costly or reserved for uncertain cases.

The annotator operator A_{D}^{ϕ} maps an artifact d, an annotation requirement ρ_{A}, and optional context $\mathcal{C}_d$ to a supervision signal and auxiliary annotation metadata:

$$(\hat{z}_d, c_d, e_d^{\mathrm{A}}) = A_{\mathrm{D}}^{\phi}(d, \rho_{\mathrm{A}}, \mathcal{C}_d),$$

where $\hat{z}_d$ denotes the generated supervision signal, c_d denotes an optional confidence score, and e_d^{A} denotes an optional annotation rationale. When available, these auxiliary signals support calibration, auditing, selective re-annotation, active human review, and downstream filtering.

Annotation research has progressed around two questions that clarify what LLM-based labeling contributes beyond low-cost label generation. The first question is whether natural-language guidelines can replace fixed labeling schemas without undermining label fidelity. Gilardi et al. [14] demonstrated this feasibility on text-classification subtasks such as relevance, stance, topic, and frame detection, while later work [15] showed that schema-grounded verification is needed before low-confidence cases are escalated to humans. Under this view, the confidence field c_d is not merely an auxiliary report. It becomes a routing signal for allocating human effort.

The second question is whether the annotation budget can be allocated more strategically than uniform bulk labeling. LLMaAA [140] addresses this question through active selection of informative instances, AFaCTA [141] derives confidence from reasoning-path consistency, and LLM-GNN [89] propagates LLM-generated node labels through graph training. Taken together, these studies suggest that the value of A_{D}^{ϕ} lies not only in attaching labels, but also in how confidence is produced, consumed, and audited across the data pool.

*2) **Synthesizer**:* Synthesis expands the data state by producing additional or richer artifacts from limited seeds, existing corpora, or structured sources. The generated artifacts may be training examples, counterfactual variants, instruction–response pairs, queries, descriptions, reasoning traces, synthetic demonstrations, or structured semantic fields. This role addresses data scarcity, long-tailed coverage, missing task formats, and underrepresented reasoning patterns.

Synthesis operates over a seed pool $\mathcal{D}_{\mathrm{seed}} \subseteq \mathcal{D}_{\mathrm{raw}}$, conditioned on requirement ρ_{S} and optional context $\mathcal{C}_{\mathrm{S}}$:

$$\tilde{\mathcal{D}} = S_{\mathrm{D}}^{\phi}(\mathcal{D}_{\mathrm{seed}}, \rho_{\mathrm{S}}, \mathcal{C}_{\mathrm{S}}),$$

where $\tilde{\mathcal{D}}$ denotes the synthesized data-artifact pool. In practice, LLM-based synthesis may involve generation, rewriting, counterfactual construction, query expansion, or tool-assisted data construction. The generated artifacts typically undergo task-specific checks, grounding checks, consistency validation, execution-based verification, or human inspection before their accepted or revised forms are incorporated into $\mathcal{D}_{\mathrm{adp}}$.

A useful way to read S_{D}^{ϕ} is by the validation burden of the generated artifact, because validation burden determines whether linguistic fluency alone suffices for adoption. In settings with relatively light validation burden, seed-conditioned methods expand small task pools or attribute specifications into instruction–response data, pairwise comparisons, attributed samples, multilingual embeddings, or utility-weighted examples [11], [13], [45], [49], [142]. Here, adoption can often begin with semantic, diversity, or format-level checks, though execution-level verification is sometimes still applied. A stricter setting arises when generated artifacts must remain consistent with an existing corpus. Corpus-conditioned methods therefore rewrite or extend data while preserving labels, discourse structure, or multilingual correspondence [50], [143]–[145].

The highest validation burden appears when correctness is defined outside language alone. Caption-derived VQA pairs must remain visually faithful [67], executable KB queries must run [55], synthetic visual data must support recognition [12], graph-conditioned examples must respect topology [87], [146], and retrieval-oriented query rewrites must improve downstream recall [101]. This setting is especially consequential for AIR because acceptance is governed by structural validity, execution outcomes, or downstream retrieval feedback rather than by fluency alone.

*3) **Optimizer**:* Optimization operates above the level of a single generated artifact. It makes pool-level decisions about retention, revision, reweighting, ranking, and removal. Its objectives include downstream robustness, coverage, diversity, difficulty control, fairness, and sample efficiency.

Pool-level optimization uses the candidate set $\mathcal{D}_{\mathrm{cand}} \subseteq \mathcal{D}_{\mathrm{raw}} \cup \tilde{\mathcal{D}}$, the requirement ρ_{O}, and feedback $\mathcal{F}_{\mathrm{O}}$ to produce an optimized data pool or selection decision:

$$\mathcal{D}^{\star} = O_{\mathrm{D}}^{\phi}(\mathcal{D}_{\mathrm{cand}}, \rho_{\mathrm{O}}, \mathcal{F}_{\mathrm{O}}),$$

where $\mathcal{D}^{\star}$ denotes an optimized data pool or data-selection decision. The optimizer may act as a selector, ranker, rewriter, reweighter, debiaser, or semantic enricher. The optimized output $\mathcal{D}^{\star}$ may contribute to the adopted set $\mathcal{D}_{\mathrm{adp}}$, but it should be understood as a role-level pool decision rather than the complete data-stage state.

What distinguishes O_{D}^{ϕ} from A_{D}^{ϕ} and S_{D}^{ϕ} is that its feedback $\mathcal{F}_{\mathrm{O}}$ estimates prospective downstream utility rather than retrospective surface quality. This requirement appears at three granularities. At the transformation level, the candidates are augmentation operations themselves. DISCO [147] filters counterfactual perturbations through teacher consistency, while LLM-AutoDA [136] searches augmentation policies tailored to long-tailed learning.

At the sample level, utility is estimated for individual instances. Self-Filter [148] scores vision-language instructions by target-VLM difficulty, and Li et al. [149] contrast responses with and without instruction context to isolate informativeness. At the auxiliary-field level, the target moves beyond the artifact d to its semantic or knowledge fields. RLMRec [150] and KAR [95] align LLM-derived user or item descriptors with downstream recommendation objectives. Across these settings, $\mathcal{F}_{\mathrm{O}}$ functions as a utility estimator, and its reliability bounds how aggressively the data pool can be selected, weighted, or revised.

*4) **Evaluator**:* Before artifacts enter training or benchmarking, evaluators assess whether they are reliable, valid, and useful enough for downstream use. Unlike optimizers, which revise the data pool toward an objective, evaluators produce diagnostic judgments as their direct outputs. Their targets include label correctness, factual faithfulness, schema

validity, semantic consistency, toxicity, duplication, benchmark leakage, and distributional coverage.

Data-stage evaluation returns a reliability score, diagnostic label, and optional explanation for artifact d, conditioned on ρ_{E} and $\mathcal{R}_d$:

$$(q_d, \delta_d, e_d^{\mathrm{E}}) = E_{\mathrm{D}}^{\phi}(d, \rho_{\mathrm{E}}, \mathcal{R}_d),$$

where q_d denotes a quality or reliability score, δ_d denotes a diagnostic label, and e_d^{E} denotes an optional explanation. The diagnostic label may indicate whether an artifact is faithful, inconsistent, ambiguous, noisy, contaminated, redundant, or otherwise unsuitable. These judgments can be used to filter artifacts, trigger human review, guide synthesis, or provide feedback to the optimizer.

Data-stage evaluators address two failure modes that surface before artifacts enter training or benchmarking. The first is localized artifact failure, where a sample, summary, or tool-use trajectory may be unsupported, invalid, unrelated, or inconsistent with its reference. Multi-News+ [151], FIZZ [152], and Iskander et al. [153] target this mode through unrelated-document detection, atomic-fact verification, and synthetic-trajectory validation.

The second is systemic pool or benchmark failure. Here, individual artifacts may pass local inspection while the aggregate still exhibits coverage gaps, contamination, or leakage. Data Advisor [132] surfaces under-covered topics and safety dimensions in alignment data, while Sainz et al. [154] and Deng et al. [155] expose benchmark contamination patterns that are invisible at the instance level. Both modes belong to E_{D}^{ϕ} because the adopted output assesses the validity of data or benchmark assets rather than model behavior.

B. LLMs for Model Design and Optimization

Model design and optimization is the stage in which LLM outputs are turned into executable or semi-executable model-development decisions. These decisions span architectures, modules, prompts, hyperparameter configurations, loss and reward definitions, training recipes, adaptation strategies, and implementation code. Since their value cannot be inferred from surface form, this stage is validation-bound. LLM-generated proposals can be adopted only after they are executed, trained, ablated, or evaluated downstream.

We denote the initial model-development state by $\mathcal{M}_0$, which may contain existing model components, design constraints, candidate architectures, task requirements, resource budgets, or partial implementations. We abstract the model-stage transformation as

$$\mathcal{M}_{\mathrm{dev}} = \mathcal{T}_{\mathrm{M}}^{\phi}\left(\mathcal{M}_0, \rho_{\mathrm{M}}, \mathcal{H}_{\mathrm{M}}\right). \tag{2}$$

The signal $\mathcal{H}_{\mathrm{M}}$ may originate from humans, empirical trials, execution traces, validation results, or external tools. Here, $\mathcal{M}_{\mathrm{dev}}$ denotes the model-development state produced by LLM-mediated proposal, feedback-driven refinement, search, or procedural planning.

In this validation-bound stage, the roles differ by when feedback enters the workflow. S_{M}^{ϕ} proposes executable candidates before empirical behavior is observed. O_{M}^{ϕ} revises or selects instantiated candidates after feedback becomes available. P_{M}^{ϕ} schedules generation, validation, and revision.

1) ***Synthesizer****:* At the model stage, synthesis turns natural-language specifications, partial designs, or task requirements into candidate artifacts that can be executed or tested. These artifacts may include architectures, modules, prompts, reward or objective definitions, training procedures, and implementation code. The role of S_{M}^{ϕ} is therefore to enlarge the candidate space before empirical feedback is available.

Model-stage synthesis maps $\mathcal{M}_{\mathrm{seed}}$, ρ_{S}, and $\mathcal{C}_{\mathrm{S}}$ to candidate model-development artifacts:

$$\tilde{\mathcal{M}} = S_{\mathrm{M}}^{\phi}(\mathcal{M}_{\mathrm{seed}}, \rho_{\mathrm{S}}, \mathcal{C}_{\mathrm{S}}),$$

where $\tilde{\mathcal{M}}$ denotes synthesized model-development artifacts. Their validity typically depends on empirical behavior rather than surface form. Generated code is expected to compile, architectures to remain compatible with the training pipeline, and proposed objectives or reward functions to elicit measurable behavior under validation. In most reported settings, such candidates enter $\mathcal{M}_{\mathrm{dev}}$ only after execution-based checking or downstream optimization.

Model-stage synthesis is governed by what counts as a testable candidate before empirical feedback is available. In the most direct setting, the candidate is a standalone design intended for trial. Arch-LLM [124] decodes discrete architecture codes for neural-architecture generation, and ModelGPT [139] maps task descriptions or user data to tailored model designs. In search-assisted settings, the candidate is instead one step in a broader exploration process. GENIUS [126], GPT-NAS [156], and graph-aware or evolutionary NAS variants [130], [157] use LLMs to propose structures, while downstream feedback handles ranking and selection.

A boundary case arises when the LLM-mediated artifact is not a full model candidate but an interface for representation transfer. BLIP-2 [69] and Flamingo [68] bridge visual encoders to frozen LLMs through lightweight connecting modules. ELLA [72] and LLM4GEN [73] inject LLM-derived semantic features into diffusion backbones. Graph learning and recommendation studies similarly align graph, user, or item representations with LLM representation spaces or LLM-derived semantics [88], [93], [94], [96], [97], [105], [158], [159]. We assign such work to S_{M}^{ϕ} when the adopted artifact is the interface itself, and reserve O_{M}^{ϕ} for cases where empirical feedback selects, ranks, or revises that interface.

2) ***Optimizer****:* Optimization starts once a candidate can be compared under explicit criteria. It uses logs, validation scores, execution traces, proxy metrics, or human preferences to revise, rank, select, or recombine model-development artifacts.

The optimizer receives a candidate pool $\mathcal{M}_{\mathrm{cand}}$, the requirement ρ_{O}, and feedback $\mathcal{F}_{\mathrm{O}}$, and returns

$$\mathcal{M}^{\star} = O_{\mathrm{M}}^{\phi}(\mathcal{M}_{\mathrm{cand}}, \rho_{\mathrm{O}}, \mathcal{F}_{\mathrm{O}}),$$

where $\mathcal{M}^{\star}$ denotes the optimized artifact pool, a selected design decision, or the next candidate configuration to evaluate. The candidate pool may contain many alternatives, as in architecture search or hyperparameter optimization, or it may degenerate to a singleton, as in iterative refinement of a prompt, module, objective, or training recipe.

The unifying mechanism across O^{ϕ}_{M} work is to verbalize prior outcomes so that they can condition the next model-stage decision. What varies is the optimized object. OPRO [122] and LLAMBO [121] instantiate the configuration-level case, where serialized hyperparameter settings and scores guide the next sampling step in black-box or Bayesian search. A second case treats the optimization target itself as an executable textual object. DiscoPOP [160] searches for preference-optimization algorithms, and Auto MC-Reward [161] revises dense reward functions through code-level checks and trajectory feedback.

The same feedback-conditioned mechanism also applies to prompt optimization [75], [162] and to training-state revision through verbalized model feedback [90], [163]. Across these cases, $\mathcal{F}_{\mathrm{O}}$ drives the loop's adaptivity rather than serving merely as logged history. This is why optimized artifacts should be separated from LLM-derived encoders, adapters, or semantic bridges that are adopted without empirical selection.

3) ***Orchestrator****:* The orchestrator organizes model development at the procedural level. Rather than producing a candidate architecture or objective, it specifies how candidate generation, training, validation, ablation, resource allocation, and revision should be sequenced.

Let $\mathcal{M}_{\mathrm{state}}$ collect candidate artifacts, training logs, validation outcomes, available resources, and unresolved design choices. The model-development policy is then expressed as:

$$\pi_{\mathrm{M}} = P^{\phi}_{\mathrm{M}}(\mathcal{M}_{\mathrm{state}}, \rho_{\mathrm{P}}, \mathcal{C}_{\mathrm{P}}),$$

where π_{M} denotes a model-development plan, search policy, experimental protocol, or sequence of design actions. The orchestrator does not replace empirical validation. It determines which validation actions are triggered, when they are executed, and how their outcomes are routed back to synthesis or optimization.

The defining feature of P^{ϕ}_{M} is that the adopted artifact is no longer a single candidate, but a procedure that decides which synthesis, execution, validation, or revision step should run next. This procedural role appears at two scopes. Within a single generation pipeline, LLMs decompose complex design or generation requests into layout plans, region-wise constraints, verification steps, or diffusion-pipeline schedules [76]–[79]. Across heterogeneous tools, they coordinate graph utilities, chat-based graph workflows, data-management pipelines, or action-feedback agents [92], [164]–[167]. The dividing line with S^{ϕ}_{M} is procedural. An output that only specifies candidate artifacts belongs in synthesis, whereas an output that specifies how candidates are validated and revised belongs here.

C. LLMs for Model Evaluation

Model evaluation turns model behavior into actionable assessment evidence. Its artifacts include benchmark instances, test cases, model responses, reference answers, rubrics, pairwise comparison records, metric descriptions, error reports, and diagnostic summaries. In this stage, LLMs contribute in three separable ways. They judge model behavior, construct what will be tested, and organize how assessment evidence is combined.

We write $\mathcal{V}_0$ for the initial evaluation state, which may contain available benchmarks, model outputs, task rubrics, metric definitions, reference materials, or prior evaluation records. The evaluation-stage transformation is defined as

$$\mathcal{V}_{\mathrm{eval}} = \mathcal{T}^{\phi}_{\mathrm{V}}\left(\mathcal{V}_0, \rho_{\mathrm{V}}, \mathcal{H}_{\mathrm{V}}\right). \tag{3}$$

The feedback term $\mathcal{H}_{\mathrm{V}}$ may include human judgments, external checks, or prior evaluation feedback. Here, $\mathcal{V}_{\mathrm{eval}}$ denotes the resulting evaluation state after LLM-mediated judging, test construction, diagnosis, or protocol organization. We use the subscript V for the evaluation stage and reserve E^{ϕ} for the evaluator role. The boundary again follows the primary LLM-mediated artifact. E^{ϕ}_{V} produces scores, preferences, or diagnostic labels, S^{ϕ}_{V} produces benchmark cases, rubrics, references, or probes, and P^{ϕ}_{V} routes instances, assigns judges, or aggregates evidence.

1) ***Evaluator****:* The evaluator maps model outputs or behaviors to task-specific scores, preferences, and diagnostic judgments. It may support LLM-as-a-judge evaluation, rubric-based assessment, preference comparison, factuality checking, safety diagnosis, reasoning assessment, and error analysis. Its distinction from data-stage evaluation lies in the target. Data-stage evaluation assesses artifact usability, whereas model evaluation assesses whether model behavior satisfies semantic, factual, safety, utility, robustness, or alignment requirements.

An evaluation unit v may be a model response, a response pair, or a test instance bundled with references, rubrics, or retrieved evidence. Formally, the evaluator operator E^{ϕ}_{V} takes v, requirement ρ_{E}, and reference information $\mathcal{R}_v$ as input, and returns a judgment triple:

$$(q_v, \delta_v, e^{\mathrm{E}}_v) = E^{\phi}_{\mathrm{V}}(v, \rho_{\mathrm{E}}, \mathcal{R}_v),$$

where q_v denotes a score, rating, or preference signal, δ_v denotes a diagnostic judgment, and e^{E}_v denotes an optional explanation. The reference $\mathcal{R}_v$ may include gold answers, source documents, rubrics, retrieved evidence, tool outputs, human judgments, or comparison baselines.

Two instantiations of the judgment function in E^{ϕ}_{V} reflect a trade-off between flexibility and specialization. Prompt-based evaluators prioritize flexibility by eliciting judgments from general-purpose LLMs through rubrics, reasoning steps, probability-based scoring, or fine-grained feedback, as illustrated by early ChatGPT-style NLG evaluation, G-EVAL, GPTScore, and InstructScore [19]–[21], [59], [168], [169]. Trained judges prioritize specialization by fine-tuning evaluator language models for customized rubrics, direct scoring, or pairwise ranking, as in JudgeLM, Prometheus, and Prometheus 2 [170]–[172]. Both routes produce the same type of adopted output, namely a score, preference, explanation, or rubric-grounded diagnostic label.

Beyond the judgment function itself, the evaluation unit v and its reference $\mathcal{R}_v$ are shaped by what counts as evidence in a given domain. RAG settings decompose assessment into faithfulness, answer relevance, context relevance, and retrieval quality because none of these properties can be inferred from the response alone [53], [54]. Conversational recommendation, machine translation, and image-text retrieval adapt the criteria to their own notions of relevance, quality, and error type [173]–[175]. For factuality and multimodal faithfulness, the evidence

grain itself becomes the unit of evaluation. Atomic facts, QA probes, visual evidence, or scene-level units replace whole-response judgments to localize semantic mismatches [46], [81], [82], [176], [177]. By contrast, capability benchmarks such as those in [83], [84] supply test instances rather than an E_{V}^{ϕ} judgment. It becomes an evaluator instance only when an LLM produces the adopted judgment.

A recurring concern across both judgment forms and evidence structures is whether E_{V}^{ϕ} outputs can be trusted as scalable substitutes for human assessment. MT-Bench and Chatbot Arena made scalable LLM-based comparison widely used, but subsequent audits documented position bias, unfair preferences, adversarial susceptibility, and other systematic biases in LLM judges [178]–[181]. Validator-alignment work further shows that even purpose-built evaluation functions require human calibration and iterative validation [182]. The practical implication is that outputs of E_{V}^{ϕ} should be treated as assessment evidence rather than terminal judgments. Their reliability must be established through references, calibration, multiple judges, human review, or task-specific metrics.

*2) **Synthesizer**:* Evaluation-stage synthesis is useful when fixed benchmarks are too narrow, stale, or potentially contaminated. It generates test cases, adversarial prompts, rubrics, reference answers, diagnostic probes, or task instructions. The primary artifact is not a judgment, but evaluation material on which later judgment can be performed.

Evaluation synthesis maps $\mathcal{V}_{\mathrm{seed}}$, ρ_{S}, and $\mathcal{C}_{\mathrm{S}}$ to generated evaluation artifacts:

$$\tilde{\mathcal{V}} = S_{\mathrm{V}}^{\phi}(\mathcal{V}_{\mathrm{seed}}, \rho_{\mathrm{S}}, \mathcal{C}_{\mathrm{S}}),$$

where $\tilde{\mathcal{V}}$ denotes synthesized evaluation artifacts. After generation, these artifacts require reference checking, difficulty and diversity control, and leakage detection before they can serve as reliable evaluation materials.

S_{V}^{ϕ} responds to two limitations of static benchmarks that become more severe as models improve. The first is distributional staleness. Dynamically generated questions and exam-style items refresh the test pool and reduce reliance on fixed labels, as in exam-based relevance evaluation and Language-Model-as-an-Examiner [52], [183]. This addresses staleness but does not by itself make broad abilities measurable.

The second limitation is criterion ambiguity. Intermediate diagnostic units decompose broad judgments into atomic facts, QA probes, or scene-level units, as instantiated by FActScore, TIFA, and Davidsonian Scene Graph [46], [81], [177]. In both cases, synthesized artifacts become reliable evaluation materials only after checks for validity, coverage, difficulty, and leakage. This dependence on later checking links S_{V}^{ϕ} to evaluator and orchestrator roles rather than leaving it as unconstrained test generation.

*3) **Orchestrator**:* When assessment depends on multiple criteria, judges, or evidence sources, the central artifact is no longer a single score but an assessment workflow. The orchestrator assigns rubrics, chooses pointwise or pairwise assessments, routes instances to judges, aggregates evidence, and triggers additional diagnostics.

With $\mathcal{V}_{\mathrm{state}}$ summarizing candidate benchmarks, model outputs, scoring rules, judge results, uncertainty signals, and resource constraints, the evaluation protocol is generated as:

$$\pi_{\mathrm{V}} = P_{\mathrm{V}}^{\phi}(\mathcal{V}_{\mathrm{state}}, \rho_{\mathrm{P}}, \mathcal{C}_{\mathrm{P}}),$$

where π_{V} denotes an evaluation protocol, routing policy, aggregation strategy, or diagnostic plan. The orchestrator complements the evaluator. E_{V}^{ϕ} produces unit-level judgments, whereas P_{V}^{ϕ} specifies how units, judges, criteria, uncertainty signals, and feedback are combined into an assessment workflow.

P_{V}^{ϕ} arises when a single judgment from E_{V}^{ϕ} is too narrow or too brittle to support a defensible verdict. One orchestration pattern composes multiple judges into a deliberative protocol. ChatEval and PRD use debate, peer-ranking, or discussion procedures to aggregate evidence rather than define a new scalar metric [47], [184]. Another pattern composes multiple steps into a diagnostic procedure. SocREval decomposes reference-free reasoning evaluation into Socratic diagnostic stages, while Visual Programming embeds text-to-image evaluation in an executable program that interleaves generation and checking [80], [185]. The common pattern is that the adopted artifact is the protocol that schedules judges, rubrics, evidence, and feedback into a coherent assessment process.

D. LLMs for Closing the AIR Loop

The preceding stages describe transformations within individual phases of the ML-centered AIR pipeline. Closing the loop requires carrying such information across iterations, so that evaluation results, data-quality signals, model-development decisions, execution logs, and human feedback can inform subsequent actions. At the loop level, LLMs operate on an evolving workflow state rather than isolated stage artifacts. The objective is to use accumulated evidence to guide the next research iteration while preserving traceability and adaptivity.

At iteration t, the AIR workflow state $\mathcal{W}^t$ contains current data, model, and evaluation states, action histories, execution logs, resource records, and accumulated feedback. Unlike the preceding stage-level transformations, the loop-level transition operates across iterations:

$$\mathcal{W}^{t+1} = \mathcal{T}_{\mathrm{L}}^{\phi}\left(\mathcal{W}^t, \rho_{\mathrm{L}}, \mathcal{H}_{\mathrm{L}}\right). \tag{4}$$

Here, $\mathcal{T}_{\mathrm{L}}^{\phi}$ represents the loop-level workflow transition induced by LLM-mediated reasoning, tool interaction, and feedback integration. It may trigger data construction, model revision, additional evaluation, human review, resource reallocation, or termination. The retained histories and logs also make the loop diagnosable and auditable. The following roles instantiate this transition through orchestration, optimization, and evaluation at the workflow level.

*1) **Orchestrator**:* At the loop level, the orchestrator links data, model, and evaluation stages into a coherent sequence of research actions. It decides whether the next step should involve data construction, model revision, additional evaluation, tool use, human feedback, resource reallocation, or

termination. The orchestrator role is therefore responsible for maintaining procedural coherence across stages.

From the current workflow state $\mathcal{W}^t$, the loop orchestrator produces a plan or action policy conditioned on ρ_{P} and $\mathcal{C}_{\mathrm{P}}$:

$$\pi_{\mathrm{L}} = P_{\mathrm{L}}^{\phi}(\mathcal{W}^t, \rho_{\mathrm{P}}, \mathcal{C}_{\mathrm{P}}),$$

where π_{L} denotes a loop-level plan, workflow policy, experiment schedule, tool-use policy, or next-stage action. The output π_{L} does not itself constitute the next workflow state; rather, it specifies how the current state should be acted upon. The actual transition to the next workflow state is governed by $\mathcal{T}_{\mathrm{L}}^{\phi}$, not by π_{L}.

At the shorter workflow horizon, loop-level orchestration can be viewed as a request-to-pipeline translation problem. AutoML-GPT [117], AutoM^{3}L [186], AutoMMLab [187], and AutoML-Agent [188] take high-level requirements and emit coordinated sequences over data preparation, model selection, training, hyperparameter tuning, verification, and deployment. What these systems contribute to P_{L}^{ϕ} is not an individual data or model artifact. It is the workflow policy that keeps stage-level procedures coherent under user requirements and resource constraints.

At a longer research horizon, the problem shifts from translation to durable state control because a research trajectory cannot be reduced to a single forward pipeline. AiScientist [189] illustrates this shift by maintaining project state across paper comprehension, environment setup, implementation, experimentation, debugging, and refinement. This setting fits more directly the cross-iteration semantics of Eq. (4), since the orchestrator carries $\mathcal{W}^t$ forward across iterations rather than merely sequencing tool calls within one run.

*2) **Optimizer**:* Loop-level optimization uses accumulated feedback to revise the workflow candidate or decision policy. Loop-level optimizers act on the configuration of the research process itself, rather than on a specific artifact pool as in the data or model stages. They may adjust data-selection strategies, model-development priorities, evaluation protocols, resource allocation, stopping criteria, feedback-routing policies, or cross-stage dependencies.

Loop-level optimization returns a revised workflow candidate based on $\mathcal{W}^t$, ρ_{O}, and accumulated feedback $\mathcal{F}_{\mathrm{O}}$:

$$\mathcal{W}^{\star} = O_{\mathrm{L}}^{\phi}(\mathcal{W}^t, \rho_{\mathrm{O}}, \mathcal{F}_{\mathrm{O}}),$$

where $\mathcal{W}^{\star}$ denotes a revised workflow candidate rather than the next workflow state itself. Adoption, execution, and validation through $\mathcal{T}_{\mathrm{L}}^{\phi}$ may still intervene before $\mathcal{W}^{\star}$ contributes to $\mathcal{W}^{t+1}$.

A first loop-level optimization problem is deciding what the workflow should try next after partial execution evidence has been observed. AgentHPO [120] makes this problem concrete at the hyperparameter level by letting prior trials condition later configurations. DS-Agent [190] and Text-to-ML [191] extend the same conditioning to data-science cases and complete ML workflow programs. What unifies these systems is that past executions are not only logged for retrospective inspection. They actively shape the next configuration, case, or program to explore. In this sense, $\mathcal{F}_{\mathrm{O}}$ is drawn from accumulated workflow evidence and consumed as a revision signal for the loop-level optimizer.

A harder problem is to optimize the solution path itself. Here, the target is represented by code, components, or research traces rather than by a single configuration. AIDE [192] formalizes machine learning engineering as tree search over solution scripts, while MLE-STAR [193] initializes from external solutions and refines selected components under ablation feedback. These systems belong to O_{L}^{ϕ} rather than to stage-level optimization because the revised object $\mathcal{W}^{\star}$ bundles implementation choices, data-processing decisions, model components, validation results, and next-step priorities into one composite workflow candidate.

*3) **Evaluator**:* The loop-level evaluator targets the workflow itself and examines its progress, reliability, and bottlenecks. Unlike data- or model-stage evaluators, it does not focus on a single artifact, response, or benchmark case; instead, it diagnoses whether the workflow is improving, stagnating, overfitting to a benchmark, accumulating unreliable data, exhausting resources, or requiring human intervention.

Workflow diagnosis is modeled as a mapping from $\mathcal{W}^t$, ρ_{E}, and loop-level reference information $\mathcal{R}_{\mathrm{L}}$ to progress, reliability, and diagnostic outputs:

$$(q_{\mathrm{L}}, \delta_{\mathrm{L}}, e_{\mathrm{L}}^{\mathrm{E}}) = E_{\mathrm{L}}^{\phi}(\mathcal{W}^t, \rho_{\mathrm{E}}, \mathcal{R}_{\mathrm{L}}),$$

where q_{L} denotes a progress, reliability, or utility estimate, δ_{L} denotes a workflow-level diagnosis, and $e_{\mathrm{L}}^{\mathrm{E}}$ denotes an optional explanation. The diagnosis may identify the failure source, the need for additional data or evaluation, or whether the loop should continue, branch, or stop. These judgments provide feedback to both the orchestrator and the optimizer, thereby supporting an adaptive AIR process.

The challenge specific to E_{L}^{ϕ} is that no single artifact carries the signal to be evaluated. The reference task must therefore be defined at the workflow level. MLAgentBench [194] and MLE-Bench [195] provide this scaffold through machine-learning experimentation and engineering tasks, where agents modify files, run experiments, inspect logs, prepare submissions, and improve performance under task-level scoring. Under these benchmarks, q_{L} measures experimental progress or engineering utility, which cannot be derived from response quality on any individual instance.

A more demanding setting evaluates long-horizon AI research rather than bounded engineering tasks. PaperBench [196] grades paper replication through hierarchically structured subtasks, while RE-Bench [197] compares open-ended research-engineering trajectories against human expert attempts. These benchmarks do not necessarily instantiate E_{L}^{ϕ}. Instead, they supply $\mathcal{R}_{\mathrm{L}}$ and workflow-level targets against which LLM-based diagnostics, explanations, and stopping decisions can be calibrated. Their value to AI4AIR is that defensible claims about autonomous AIR require process-level evidence about progress, reproducibility, resource use, and failure localization. Such evidence cannot be captured by stage-level evaluators in isolation.

Across these three roles, P_{L}^{ϕ} selects the next workflow action, O_{L}^{ϕ} revises the policy or configuration under which

actions are executed, and E_{L}^{ϕ} supplies the progress and reliability signal on which both depend.

E. Summary and Discussion

This section examined the AIR pipeline through five recurring roles rather than four stages. Three observations follow. First, the roles distribute unevenly across the pipeline: the annotator appears only in data engineering, where supervision attaches to discrete artifacts, while the orchestrator gains weight only as single-step validation becomes intractable. Second, the synthesizer and optimizer recur across stages, but what they produce and what feedback they consume shifts as the validation cost of the underlying artifact rises, from inspectable samples in S_{D}^{ϕ} to executable specifications in S_{M}^{ϕ} and test materials in S_{V}^{ϕ}. Third, within each stage the evaluator E_s^{ϕ}, optimizer O_s^{ϕ}, and orchestrator P_s^{ϕ} form a recurring triangle, which reappears at the loop level and drives the transition $\mathcal{T}_{\mathrm{L}}^{\phi}$.

The same patterns also locate where the pipeline becomes brittle. Roles whose artifacts admit linguistic inspection or external grounding, such as the annotator and most uses of the synthesizer, are constrained mainly by faithfulness, coverage, and bias control, since their outputs can be checked before adoption. Roles whose artifacts require empirical validation, such as the optimizer and orchestrator at the model and loop levels, are constrained instead by feedback fidelity, credit assignment, and long-horizon stability, since errors propagate through execution rather than surface in inspection.

This split suggests that reliability investments cannot be uniform across roles, that workflow-level diagnostics deserve more attention than they currently receive, and that the triangle formed by the evaluator, optimizer, and orchestrator is itself a unit worth engineering.

V. Potential Future Directions

While the AI4AIR paradigm is advancing rapidly, its current capabilities remain fragmented across isolated tasks and pipeline stages. We outline three interconnected directions, from model-level LLM capabilities to pipeline-level integration and then to reproducibility and auditability in resource-limited and long-tail settings. Across all three, the core question is whether LLM-mediated operations can be verified, compared, and reused under realistic research budgets.

A. Advancing LLM Capabilities for AIR

1) ***Intrinsic Enhancements within LLMs****:* A working assumption of AI4AIR is that stronger foundation models may provide more reliable support across the ML lifecycle. While contemporary LLMs can generate intermediate reasoning steps fluently [10], [198], [199], these steps are not always externally checkable in rigorous AIR contexts, especially under multi-hop inference and scientific deduction [200]. Verifiable reasoning therefore depends on coupling generation with self-verification and critique mechanisms [201], [202], and on grounding mathematical or formal reasoning in tool-augmented computation, such as retrieval-augmented theorem proving [3]. Numerical and scientific reasoning remain bottlenecks [203]. Practical deployment also requires parameter-efficient adaptation and compression under realistic compute and latency budgets [34], [36].

2) ***External Augmentation and Integration****:* Beyond intrinsic reasoning, an unresolved question is how LLMs can interact with external environments over long horizons while limiting the propagation of early errors. AI research produces executable artifacts, including code, logs, and tool outputs. This makes robust tool use, standardized interfaces, and long-horizon planning important requirements for more reliable AIR automation. Agentic systems provide initial evidence for multi-step orchestration [64], and LLM-based ML workflow systems show how high-level objectives can be decomposed into executable subtasks [117], [118]. A promising direction is to develop failure-aware orchestrators that can retain recoverable traces across stages. AIR support would also benefit from stronger structured and multimodal grounding. Potential directions include visual knowledge graph integration [204], multimodal judge protocols for visual-textual assessment [205], and retrieval-augmented generation as a broadly applicable grounding mechanism [33].

B. Imbalanced LLM Integration in AIR Pipelines

1) ***Underexplored Roles in Model Design and Optimization****:* The maturity of LLM integration varies significantly across AIR pipeline stages. LLMs are now common in data preparation and evaluation, but they are less established in architecture design and training optimization. This disparity stems from the costly and delayed feedback inherent in training runs, the challenges of high-dimensional search spaces, and the difficulty of assigning credit to individual design decisions [30], [206]. Existing attempts in this direction, such as reward-design loops with rollout feedback [123] and code-level evolutionary search [129], suggest that semantic proposals become more reliable when grounded in executable feedback. Rigorous early verification remains underused, although reflective zero-cost strategies [127] offer one way to screen LLM-proposed interventions before expensive training runs. Executable co-design should be paired with low-cost diagnostics, so that semantic proposals can be screened before full training.

2) ***Open Challenges in the Data and Evaluation Stages****:* In the data and evaluation stages, where LLM integration is more mature, the central bottleneck shifts from capability scaling to reliability and interpretability [13], [182]. Data-stage failures often arise silently through bias amplification, contamination, and weak provenance tracking [155], [207]. These failures can propagate into downstream training and evaluation. Human-LLM verification can help check LLM-produced labels [15], but cross-model agreement, leakage auditing, and distributional-artifact detection still lack common protocols. Evaluation-stage use of LLM-as-a-Judge offers scalable assessment, yet it remains sensitive to prompt phrasing, position bias, and superficial cues [179], [180]. Existing remedies include multi-agent debate [47] and decompositional evaluation based on atomic claims or retrieval-grounded

checks [46], [54]. The open problem is how to calibrate these mechanisms across tasks, models, and time, rather than deploying them as isolated checks. Bridging this gap requires treating LLM interventions as executable and auditable pipeline components, each coupled with low-cost verification and logged evidence. Beyond these component-level checks, AI4AIR also needs workflow-level meta-evaluation. Future studies should test whether LLM-mediated pipelines produce better models, datasets, or evaluation protocols than LLM-free or human-designed workflows. Such comparisons should use the same data, compute, and human-review budget, with role-level ablations and execution logs for attribution.

C. Inclusive AI4AIR in Constrained Settings

1) ***Empowering Resource-Limited Research Participation****:* Resource-limited participation is better assessed by operational access than by API availability alone. Constrained groups would benefit from AIR workflows that can be rerun, inspected, and compared under explicit data, compute, and evaluation conditions. For these groups, parameter-efficient fine-tuning and model compression can help reduce adaptation costs and hardware requirements [34], [36]. For AI4AIR, however, such lightweight deployments become more reusable when they also preserve audit trails, evaluation logs, and reproducible adaptation recipes. Standardized pipelines that package automated data checks, evaluation rubrics, and audit logs as reusable components would help resource-limited groups validate and extend prior work through shared artifacts rather than proprietary full stacks.

2) ***Unlocking Long-Tail AIR Scenarios****:* Long-tail AIR scenarios remain underexplored, partly because they often combine limited data, scarce expertise, and weak evaluation infrastructure. Early work on LLM-driven data augmentation for long-tailed problems [136] and long-tail visual recognition [65] suggests that generated content can improve tail-class coverage under controlled settings. At the same time, attributed data synthesis reveals bias risks in synthetic supervision [13]. A more difficult open question is whether such tools can support comparable evaluations, rather than only producing more data, in settings where validation is itself under-resourced [106]. Progress is therefore better defined by reproducible task formulation, controllable dataset synthesis, and comparable evaluation protocols than by raw resource scaling. Such a framing would shift inclusion from a broad aspiration toward a workflow property that can be checked and compared.

VI. Conclusion

In this survey, we present AI4AIR as a systematic framework for characterizing how large language models are integrated into ML-centered AI research. We organize existing studies from two complementary perspectives: a domain-oriented view covering major AI sub-domains, and a pipeline-oriented view covering data engineering, model design and optimization, model evaluation, and closing the AIR loop. The reviewed literature suggests that LLMs are moving beyond peripheral assistance in many ML-centered workflows. Instead, they increasingly act as annotators, synthesizers, optimizers, evaluators, and orchestrators across the AI research lifecycle. Through these roles, LLMs help construct and refine research artifacts, support model-development decisions, assist evaluation, and coordinate iterative feedback. These capabilities may accelerate experimentation and make complex model-development workflows more accessible. At the same time, their broader adoption depends on addressing persistent challenges in reliability, interpretability, bias, and controllability. Future AI4AIR systems should therefore emphasize verifiability, adaptivity, and human-controllable workflow design.

References

[1] Y. Li, L. Chen, A. Liu, K. Yu, and L. Wen, "Chatcite: Llm agent with human workflow guidance for comparative literature summary," in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 3613–3630.

[2] Z. Yang, X. Du, J. Li, J. Zheng, S. Poria, and E. Cambria, "Large language models for automated open-domain scientific hypotheses discovery," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 13 545–13 565.

[3] K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. J. Prenger, and A. Anandkumar, "Leandojo: Theorem proving with retrieval-augmented language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 21 573–21 612, 2023.

[4] Y. Zhang, S. A. Khan, A. Mahmud, H. Yang, A. Lavin, M. Levin, J. Frey, J. Dunnmon, J. Evans, A. Bundy *et al.*, "Exploring the role of large language models in the scientific method: from hypothesis to discovery," *npj Artificial Intelligence*, vol. 1, no. 1, p. 14, 2025.

[5] T. Burki, "Nobel prizes honour ai pioneers and pioneering ai," *The Lancet Digital Health*, vol. 7, no. 1, pp. e11–e12, 2025.

[6] H. Wang, T. Fu, Y. Du, W. Gao, K. Huang, Z. Liu, P. Chandak, S. Liu, P. Van Katwyk, A. Deac *et al.*, "Scientific discovery in the age of artificial intelligence," *Nature*, vol. 620, no. 7972, pp. 47–60, 2023.

[7] Q. Chen, M. Yang, L. Qin, J. Liu, Z. Yan, J. Guan, D. Peng, Y. Ji, H. Li, M. Hu *et al.*, "Ai4research: A survey of artificial intelligence for scientific research," *arXiv preprint arXiv:2507.01903*, 2025.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] Y. Deldjoo, Z. He, J. McAuley, A. Korikov, S. Sanner, A. Ramisa, R. Vidal, M. Sathiamoorthy, A. Kasirzadeh, and S. Milano, "A review of modern recommender systems using generative models (gen-recsys)," in *Proceedings of the 30th ACM SIGKDD conference on Knowledge Discovery and Data Mining*, 2024, pp. 6448–6458.

[10] 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*, vol. 35, pp. 24 824–24 837, 2022.

[11] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language models with self-generated instructions," in *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, 2023, pp. 13 484–13 508.

[12] R. He, S. Sun, X. Yu, C. Xue, W. Zhang, P. Torr, S. Bai, and X. Qi, "Is synthetic data from generative models ready for image recognition?" *arXiv preprint arXiv:2210.07574*, 2022.

[13] Y. Yu, Y. Zhuang, J. Zhang, Y. Meng, A. J. Ratner, R. Krishna, J. Shen, and C. Zhang, "Large language model as attributed training data generator: A tale of diversity and bias," *Advances in neural information processing systems*, vol. 36, pp. 55 734–55 784, 2023.

[14] F. Gilardi, M. Alizadeh, and M. Kubli, "Chatgpt outperforms crowd workers for text-annotation tasks," *Proceedings of the National Academy of Sciences*, vol. 120, no. 30, p. e2305016120, 2023.

[15] X. Wang, H. Kim, S. Rahman, K. Mitra, and Z. Miao, "Human-llm collaborative annotation through effective verification of llm labels," in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–21.

[16] 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*, vol. 33, pp. 1877–1901, 2020.

[17] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, "Large language models are zero-shot time series forecasters," *Advances in Neural Information Processing Systems*, vol. 36, pp. 19 622–19 635, 2023.

[18] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[19] J. Fu, S. K. Ng, Z. Jiang, and P. Liu, "Gptscore: Evaluate as you desire," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 6556–6576.

[20] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: Nlg evaluation using gpt-4 with better human alignment," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 2511–2522.

[21] M. Gao, J. Ruan, R. Sun, X. Yin, S. Yang, and X. Wan, "Human-like summarization evaluation with chatgpt," *arXiv preprint arXiv:2304.02554*, 2023.

[22] M. Awais, M. Naseer, S. Khan, R. M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, and F. S. Khan, "Foundation models defining a new era in vision: a survey and outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[23] 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*, vol. 46, no. 8, pp. 5625–5644, 2024.

[24] D. Liu, M. Yang, X. Qu, P. Zhou, Y. Cheng, and W. Hu, "A survey of attacks on large vision–language models: Resources, advances, and future trends," *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[25] J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, H. Zhang, Y. Liu, C. Wu, X. Li, C. Zhu *et al.*, "How can recommender systems benefit from large language models: A survey," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–47, 2025.

[26] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang *et al.*, "Recommender systems in the era of large language models (llms)," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 11, pp. 6889–6907, 2024.

[27] X. Ren, J. Tang, D. Yin, N. Chawla, and C. Huang, "A survey of large language models for graphs," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 6616–6626.

[28] B. Jin, G. Liu, C. Han, M. Jiang, H. Ji, and J. Han, "Large language models on graphs: A comprehensive survey," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[29] J. Liu, C. Yang, Z. Lu, J. Chen, Y. Li, M. Zhang, T. Bai, Y. Fang, L. Sun, P. S. Yu *et al.*, "Graph foundation models: Concepts, opportunities and challenges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[30] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[31] R. Barbudo, S. Ventura, and J. R. Romero, "Eight years of automl: categorisation, review and trends," *Knowledge and Information Systems*, vol. 65, no. 12, pp. 5097–5149, 2023.

[32] Y. Gu, H. You, J. Cao, M. Yu, H. Fan, and S. Qian, "Large language models for constructing and optimizing machine learning workflows: A survey," *ACM Transactions on Software Engineering and Methodology*, 2025.

[33] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[34] 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.

[35] W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang, "A survey on mixture of experts in large language models," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[36] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[37] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[38] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, 2022.

[39] W. Wang, Y. Yang, and F. Wu, "Towards data-and knowledge-driven ai: a survey on neuro-symbolic computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[40] L. Böttcher, N. Antulov-Fantulin, and T. Asikis, "Ai pontryagin or how artificial neural networks learn to control dynamical systems," *Nature communications*, vol. 13, no. 1, p. 333, 2022.

[41] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[42] Z. Yang, X. Zeng, Y. Zhao, and R. Chen, "Alphafold2 and its applications in the fields of biology and medicine," *Signal Transduction and Targeted Therapy*, vol. 8, no. 1, p. 115, 2023.

[43] K. Guo, Z. Yang, C.-H. Yu, and M. J. Buehler, "Artificial intelligence and machine learning in design of mechanical materials," *Materials Horizons*, vol. 8, no. 4, pp. 1153–1172, 2021.

[44] A. Blanco-Gonzalez, A. Cabezon, A. Seco-Gonzalez, D. Conde-Torres, P. Antelo-Riveiro, A. Pineiro, and R. Garcia-Fandino, "The role of ai in drug discovery: challenges, opportunities, and strategies," *Pharmaceuticals*, vol. 16, no. 6, p. 891, 2023.

[45] B. Peng, C. Li, P. He, M. Galley, and J. Gao, "Instruction tuning with gpt-4," *arXiv preprint arXiv:2304.03277*, 2023.

[46] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi, "Factscore: Fine-grained atomic evaluation of factual precision in long form text generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 12 076–12 100.

[47] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, "Chateval: Towards better llm-based evaluators through multi-agent debate," *arXiv preprint arXiv:2308.07201*, 2023.

[48] Y. Liu, K. Shi, K. He, L. Ye, A. R. Fabbri, P. Liu, D. Radev, and A. Cohan, "On learning to summarize with large language models as references," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 8647–8664.

[49] J. Gao, R. Pi, Y. Lin, H. Xu, J. Ye, Z. Wu, W. Zhang, X. Liang, Z. Li, and L. Kong, "Self-guided noise-free data generation for efficient zero-shot learning," *arXiv preprint arXiv:2205.12679*, 2022.

[50] Z. Li, W. Chen, S. Li, H. Wang, J. Qian, and X. Yan, "Controllable dialogue simulation with in-context learning," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022, pp. 4330–4347.

[51] J. Cegin, B. Pecher, J. Simko, I. Srba, M. Bieliková, and P. Brusilovsky, "Effects of diversity incentives on sample diversity and downstream model performance in llm-based text augmentation," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 13 148–13 171.

[52] Y. Bai, J. Ying, Y. Cao, X. Lv, Y. He, X. Wang, J. Yu, K. Zeng, Y. Xiao, H. Lyu *et al.*, "Benchmarking foundation models with language-model-as-an-examiner," *Advances in Neural Information Processing Systems*, vol. 36, pp. 78 142–78 167, 2023.

[53] A. Muhamed, "Ccrs: A zero-shot llm-as-a-judge framework for comprehensive rag evaluation," *arXiv preprint arXiv:2506.20128*, 2025.

[54] S. Es, J. James, L. E. Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024, pp. 150–158.

[55] Z. Li, S. Fan, Y. Gu, X. Li, Z. Duan, B. Dong, N. Liu, and J. Wang, "Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 17, 2024, pp. 18 608–18 616.

[56] M. Enis and M. Hopkins, "From llm to nmt: Advancing low-resource machine translation with claude," *arXiv preprint arXiv:2404.13813*, 2024.

[57] S. Pan, Z. Tian, L. Ding, H. Zheng, Z. Huang, Z. Wen, and D. Li, "Pomp: Probability-driven meta-graph prompter for llms in low-resource unsupervised neural machine translation," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 9976–9992.

[58] A. El Mekki and M. Abdul-Mageed, "Effective self-mining of in-context examples for unsupervised machine translation with llms," in *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025, pp. 4229–4256.

[59] C.-H. Chiang and H.-Y. Lee, "Can large language models be an alternative to human evaluations?" in *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023, pp. 15 607–15 631.

[60] S. Jin, X. Jiang, J. Huang, L. Lu, and S. Lu, "Llms meet vlms: Boost open vocabulary object detection with fine-grained descriptors," *arXiv preprint arXiv:2402.04630*, 2024.

[61] Y. Zhou, S. Zhao, Y. Chen, Z. Wang, C. Jin, and D. N. Metaxas, "Led: Llm enhanced open-vocabulary object detection without human curated data generation," *arXiv preprint arXiv:2503.13794*, 2025.

[62] S. Fu, Q. Yang, Q. Mo, J. Yan, X. Wei, J. Meng, X. Xie, and W.-S. Zheng, "Llmdet: Learning strong open-vocabulary object detectors under the supervision of large language models," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 14 987–14 997.

[63] H. Tang, Y. Zhao, Y. Huang, M. Xu, J. Wang, and Q. Wu, "Lmseg: Unleashing the power of large-scale models for open-vocabulary semantic segmentation," *arXiv preprint arXiv:2412.00364*, 2024.

[64] C. Wu, S. Yin, W. Qi, X. Wang, Z. Tang, and N. Duan, "Visual chatgpt: Talking, drawing and editing with visual foundation models," *arXiv preprint arXiv:2303.04671*, 2023.

[65] Q. Zhao, Y. Dai, H. Li, W. Hu, F. Zhang, and J. Liu, "Ltgc: Long-tail recognition via leveraging llms-driven generated content," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 510–19 520.

[66] Q. Yu, X. Shen, and L.-C. Chen, "Towards open-ended visual recognition with large language models," in *European Conference on Computer Vision*. Springer, 2024, pp. 359–376.

[67] S. Changpinyo, D. Kukliansy, I. Szpektor, X. Chen, N. Ding, and R. Soricut, "All you may need for vqa are image captions," in *Proceedings of the 2022 conference of the north american chapter of the association for computational linguistics: human language technologies*, 2022, pp. 1947–1963.

[68] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.

[69] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *International conference on machine learning*. PMLR, 2023, pp. 19 730–19 742.

[70] X. Chen, Y. Lin, Y. Zhang, and W. Huang, "Autoeval-video: An automatic benchmark for assessing large vision language models in open-ended video question answering," in *European Conference on Computer Vision*. Springer, 2024, pp. 179–195.

[71] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.

[72] X. Hu, R. Wang, Y. Fang, B. Fu, P. Cheng, and G. Yu, "Ella: Equip diffusion models with llm for enhanced semantic alignment," *arXiv preprint arXiv:2403.05135*, 2024.

[73] M. Liu, Y. Ma, Z. Yang, J. Dan, Y. Yu, Z. Zhao, Z. Hu, B. Liu, and C. Fan, "Llm4gen: Leveraging semantic representation of llms for text-to-image generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 5, 2025, pp. 5523–5531.

[74] S. Zhong, Z. Huang, W. Wen, J. Qin, and L. Lin, "Sur-adapter: Enhancing text-to-image pre-trained diffusion models with large language models," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 567–578.

[75] Y. Hao, Z. Chi, L. Dong, and F. Wei, "Optimizing prompts for text-to-image generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 66 923–66 939, 2023.

[76] W. Feng, W. Zhu, T.-j. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang, "Layoutgpt: Compositional visual planning and generation with large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 18 225–18 250, 2023.

[77] Z. Wang, E. Xie, A. Li, Z. Wang, X. Liu, and Z. Li, "Divide and conquer: Language models can plan and self-correct for compositional text-to-image generation," *arXiv preprint arXiv:2401.15688*, 2024.

[78] L. Lian, B. Li, A. Yala, and T. Darrell, "Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models," *arXiv preprint arXiv:2305.13655*, 2023.

[79] L. Yang, Z. Yu, C. Meng, M. Xu, S. Ermon, and B. Cui, "Mastering text-to-image diffusion: Recaptioning, planning, and generating with multimodal llms," in *Forty-first International Conference on Machine Learning*, 2024.

[80] J. Cho, A. Zala, and M. Bansal, "Visual programming for step-by-step text-to-image generation and evaluation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 6048–6069, 2023.

[81] Y. Hu, B. Liu, J. Kasai, Y. Wang, M. Ostendorf, R. Krishna, and N. A. Smith, "Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 20 406–20 417.

[82] M. Yarom, Y. Bitton, S. Changpinyo, R. Aharoni, J. Herzig, O. Lang, E. Ofek, and I. Szpektor, "What you see is what you read? improving text-image alignment evaluation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 1601–1619, 2023.

[83] H. Wang, S. Feng, T. He, Z. Tan, X. Han, and Y. Tsvetkov, "Can language models solve graph problems in natural language?" *Advances in Neural Information Processing Systems*, vol. 36, pp. 30 840–30 861, 2023.

[84] J. Guo, L. Du, H. Liu, M. Zhou, X. He, and S. Han, "Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking," *arXiv preprint arXiv:2305.15066*, 2023.

[85] J. Zhao, L. Zhuo, Y. Shen, M. Qu, K. Liu, M. Bronstein, Z. Zhu, and J. Tang, "Graphtext: Graph reasoning in text space," *arXiv preprint arXiv:2310.01089*, 2023.

[86] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, "Language is all a graph needs," in *Findings of the association for computational linguistics: EACL 2024*, 2024, pp. 1955–1973.

[87] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. Huang, "Graphgpt: Graph instruction tuning for large language models," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 491–500.

[88] M. Zhang, M. Sun, P. Wang, S. Fan, Y. Mo, X. Xu, H. Liu, C. Yang, and C. Shi, "Graphtranslator: Aligning graph model to large language model for open-ended tasks," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 1003–1014.

[89] Z. Chen, H. Mao, H. Wen, H. Han, W. Jin, H. Zhang, H. Liu, and J. Tang, "Label-free node classification on graphs with large language models (llms)," *arXiv preprint arXiv:2310.04668*, 2023.

[90] Y. Qiao, X. Ao, Y. Liu, J. Xu, X. Sun, and Q. He, "Login: A large language model consulted graph neural network training framework," in *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 2025, pp. 232–241.

[91] Z. Zhang, X. Wang, H. Zhou, Y. Yu, M. Zhang, C. Yang, and C. Shi, "Can large language models improve the adversarial robustness of graph neural networks?" *arXiv preprint arXiv:2408.08685*, 2024.

[92] J. Zhang, "Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt," *arXiv preprint arXiv:2304.11116*, 2023.

[93] Y. Gong, X. Ding, Y. Su, K. Shen, Z. Liu, and G. Zhang, "An unified search and recommendation foundation model for cold-start scenario," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 4595–4601.

[94] W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Llmrec: Large language models with graph augmentation for recommendation," in *Proceedings of the 17th ACM international conference on web search and data mining*, 2024, pp. 806–815.

[95] Y. Xi, W. Liu, J. Lin, X. Cai, H. Zhu, J. Zhu, B. Chen, R. Tang, W. Zhang, and Y. Yu, "Towards open-world recommendation with knowledge augmentation from large language models," in *Proceedings of the 18th ACM Conference on Recommender Systems*, 2024, pp. 12–22.

[96] 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*, 2024, pp. 365–374.

[97] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He, "Tallrec: An effective and efficient tuning framework to align large language model with recommendation," in *Proceedings of the 17th ACM conference on recommender systems*, 2023, pp. 1007–1014.

[98] L. Wang, N. Yang, and F. Wei, "Query2doc: Query expansion with large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 9414–9423.

[99] F. Ye, M. Fang, S. Li, and E. Yilmaz, "Enhancing conversational search: Large language model-aided informative query rewriting," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 5985–6006.

[100] F. Mo, A. Ghaddar, K. Mao, M. Rezagholizadeh, B. Chen, Q. Liu, and J.-Y. Nie, "Chiq: Contextual history enhancement for improving query rewriting in conversational search," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 2253–2268.

[101] W. Peng, G. Li, Y. Jiang, Z. Wang, D. Ou, X. Zeng, D. Xu, T. Xu, and E. Chen, "Large language model based long-tail query rewriting in taobao search," in *Companion Proceedings of the ACM Web Conference 2024*, 2024, pp. 20–28.

[102] L. Bonifacio, H. Abonizio, M. Fadaee, and R. Nogueira, "Inpars: Unsupervised dataset generation for information retrieval," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 2387–2392.

[103] Z. Dai, V. Y. Zhao, J. Ma, Y. Luan, J. Ni, J. Lu, A. Bakalov, K. Guu, K. B. Hall, and M.-W. Chang, "Promptagator: Few-shot dense retrieval from 8 examples," *arXiv preprint arXiv:2209.11755*, 2022.

[104] W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, and Z. Ren, "Is chatgpt good at search? investigating large language models as re-ranking agents," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 14 918–14 937.

[105] F. Yang, Z. Chen, Z. Jiang, E. Cho, X. Huang, and Y. Lu, "Palr: Personalization aware llms for recommendation," *arXiv preprint arXiv:2305.07622*, 2023.

[106] H. A. Rahmani, C. Siro, M. Aliannejadi, N. Craswell, C. L. Clarke, G. Faggioli, B. Mitra, P. Thomas, and E. Yilmaz, "Llm4eval: Large language model for evaluation in ir," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 3040–3043.

[107] M. D. de Souza, F. Santos, and C. Zanchettin, "Evaluating the usefulness of large language models for synthetic samples generation via few-shot learning," in *Latinx in AI@ NeurIPS 2024*, 2024.

[108] C. Rousseau, T. Boschi, G. Cornacchia, D. Salwala, A. Pascale, and J. B. Moreno, "Forging time series with language: A large language model approach to synthetic data generation," *arXiv preprint arXiv:2505.17103*, 2025.

[109] A. Hota, S. Chatterjee, and S. Chakraborty, "Evaluating large language models as virtual annotators for time-series physical sensing data," *ACM Transactions on Intelligent Systems and Technology*, vol. 16, no. 6, pp. 1–25, 2025.

[110] S. Wu, D. Li, H. Ye, Z. Chen, J. Zhou, J. Lou, Z. Zheng, and S.-K. Ng, "Tsrating: Rating quality of diverse time series data by meta-learning from llm judgment," *arXiv preprint arXiv:2506.01290*, 2025.

[111] G. Sperlì and M. A. Sichinolfi, "Empowered stock market forecasting using large language model on social media content," *Engineering Applications of Artificial Intelligence*, vol. 162, p. 112727, 2025.

[112] T. Lan, H. D. Le, J. Li, W. He, M. Wang, C. Liu, and C. Zhang, "Axis: Explainable time series anomaly detection with large language models," *arXiv preprint arXiv:2509.24378*, 2025.

[113] D. Roy, X. Zhang, R. Bhave, C. Bansal, P. Las-Casas, R. Fonseca, and S. Rajmohan, "Exploring llm-based agents for root cause analysis," in *Companion proceedings of the 32nd ACM international conference on the foundations of software engineering*, 2024, pp. 208–219.

[114] J. Xu, Q. Zhang, Z. Zhong, S. He, C. Zhang, Q. Lin, D. Pei, P. He, D. Zhang, and Q. Zhang, "Openrca: Can large language models locate the root cause of software failures?" in *The Thirteenth International Conference on Learning Representations*, 2025.

[115] E. Fons, E. Kochkina, R. Kaur, Z. Zeng, B. Hlavaty, C. Smiley, S. Vyetrenko, and M. Veloso, "Ai analyst: Framework and comprehensive evaluation of large language models for financial time series report generation," *arXiv preprint arXiv:2507.00718*, 2025.

[116] C. Liu, Y. Ma, K. Kothur, A. Nikpour, and O. Kavehei, "Biosignal copilot: Leveraging the power of llms in drafting reports for biomedical signals," *medRxiv*, pp. 2023–06, 2023.

[117] S. Zhang, C. Gong, L. Wu, X. Liu, and M. Zhou, "Automl-gpt: Automatic machine learning with gpt," *arXiv preprint arXiv:2305.02499*, 2023.

[118] L. Zhang, Y. Zhang, K. Ren, D. Li, and Y. Yang, "Mlcopilot: Unleashing the power of large language models in solving machine learning tasks," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 2931–2959.

[119] K. Mahammadli and S. Ertekin, "Sequential large language model-based hyper-parameter optimization," *arXiv preprint arXiv:2410.20302*, 2024.

[120] S. Liu, C. Gao, and Y. Li, "Agenthpo: Large language model agent for hyper-parameter optimization," in *Conference on Parsimony and Learning*. PMLR, 2025, pp. 1146–1169.

[121] T. Liu, N. Astorga, N. Seedat, and M. van der Schaar, "Large language models to enhance bayesian optimization," *arXiv preprint arXiv:2402.03921*, 2024.

[122] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large language models as optimizers," *arXiv preprint arXiv:2309.03409*, 2023.

[123] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[124] D. G. Poddenige, S. Seneviratne, D. Senanayake, M. Niranjan, P. Suganthan, and S. Halgamuge, "Arch-llm: Taming llms for neural architecture generation via unsupervised discrete representation learning," *arXiv preprint arXiv:2503.22063*, 2025.

[125] Y. Hu, J. Liu, K. Wang, J. Zheng, W. Shi, M. Zhang, Q. Dou, R. Liu, A. Zhou, and H. Li, "Lm-searcher: Cross-domain neural architecture search with llms via unified numerical encoding," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 9419–9432.

[126] M. Zheng, X. Su, S. You, F. Wang, C. Qian, C. Xu, and S. Albanie, "Can gpt-4 perform neural architecture search?" *arXiv preprint arXiv:2304.10970*, 2023.

[127] Z. Ji, G. Zhu, C. Yuan, and Y. Huang, "Rz-nas: Enhancing llm-guided neural architecture search via reflective zero-cost strategy," in *Forty-second International Conference on Machine Learning*, 2025.

[128] G. Jawahar, M. Abdul-Mageed, L. Lakshmanan, and D. Ding, "Llm performance predictors are good initializers for architecture search," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 10 540–10 560.

[129] A. Chen, D. Dohan, and D. So, "Evoprompting: Language models for code-level neural architecture search," *Advances in neural information processing systems*, vol. 36, pp. 7787–7817, 2023.

[130] C. Morris, M. Jurado, and J. Zutty, "Llm guided evolution-the automation of models advancing models," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 377–384.

[131] M. U. Nasir, S. Earle, J. Togelius, S. James, and C. Cleghorn, "Llmatic: neural architecture search via large language models and quality diversity optimization," in *proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 1110–1118.

[132] F. Wang, N. Mehrabi, P. Goyal, R. Gupta, K.-W. Chang, and A. Galstyan, "Data advisor: Dynamic data curation for safety alignment of large language models," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 8089–8100.

[133] H. Zhou, Y. Tang, H. Qin, Y. Yang, R. Jin, D. Xiong, K. Han, and Y. Wang, "Star-agents: Automatic data optimization with llm agents for instruction tuning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 4575–4597, 2024.

[134] J. Wang, X. Lin, R. Qiao, P. W. Koh, C.-S. Foo, and B. K. H. Low, "Nice: Non-differentiable evaluation metric-based data selection for instruction tuning," in *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2025.

[135] S. Yu, L. Chen, S. Ahmadian, and M. Fadaee, "Diversify and conquer: Diversity-centric data selection with iterative refinement," *arXiv preprint arXiv:2409.11378*, 2024.

[136] P. Wang, Z. Zhao, H. Wen, F. Wang, B. Wang, Q. Zhang, and Y. Wang, "Llm-autoda: Large language model-driven automatic data augmentation for long-tailed problems," *Advances in Neural Information Processing Systems*, vol. 37, pp. 64 915–64 941, 2024.

[137] K. Choi, C. Cundy, S. Srivastava, and S. Ermon, "Lmpriors: Pre-trained language models as task-specific priors," *arXiv preprint arXiv:2210.12530*, 2022.

[138] D. P. Jeong, Z. C. Lipton, and P. Ravikumar, "Llm-select: Feature selection with large language models," *arXiv preprint arXiv:2407.02694*, 2024.

[139] Z. Tang, Z. Lv, S. Zhang, F. Wu, and K. Kuang, "Modelgpt: Unleashing llm's capabilities for tailored model generation," *arXiv preprint arXiv:2402.12408*, 2024.

[140] R. Zhang, Y. Li, Y. Ma, M. Zhou, and L. Zou, "Llmaaa: Making large language models as active annotators," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 13 088–13 103.

[141] J. Ni, M. Shi, D. Stammbach, M. Sachan, E. Ash, and M. Leippold, "Afacta: Assisting the annotation of factual claim detection with reliable llm annotators," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 1890–1912.

[142] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Improving text embeddings with large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 11 897–11 916.

[143] H. Dai, Z. Liu, W. Liao, X. Huang, Y. Cao, Z. Wu, L. Zhao, S. Xu, F. Zeng, W. Liu *et al.*, "Auggpt: Leveraging chatgpt for text data augmentation," *IEEE Transactions on Big Data*, vol. 11, no. 3, pp. 907–918, 2025.

[144] F. Piedboeuf and P. Langlais, "Is chatgpt the ultimate data augmentation algorithm?" in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 15 606–15 615.

[145] C. Whitehouse, M. Choudhury, and A. F. Aji, "Llm-powered data augmentation for enhanced cross-lingual performance," in *Proceedings of the 2023 conference on empirical methods in natural language processing*, 2023, pp. 671–686.

[146] J. Yu, Y. Ren, C. Gong, J. Tan, X. Li, and X. Zhang, "Leveraging large language models for node generation in few-shot learning on text-attributed graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 39, no. 12, 2025, pp. 13 087–13 095.

[147] Z. Chen, Q. Gao, A. Bosselut, A. Sabharwal, and K. Richardson, "Disco: Distilling counterfactuals with large language models," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 5514–5528.

[148] R. Chen, Y. Wu, L. Chen, G. Liu, Q. He, T. Xiong, C. Liu, J. Guo, and H. Huang, "Your vision-language model itself is a strong filter: Towards high-quality instruction tuning with data selection," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 4156–4172.

[149] M. Li, Y. Zhang, Z. Li, J. Chen, L. Chen, N. Cheng, J. Wang, T. Zhou, and J. Xiao, "From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 7602–7635.

[150] X. Ren, W. Wei, L. Xia, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Representation learning with large language models for recommendation," in *Proceedings of the ACM web conference 2024*, 2024, pp. 3464–3475.

[151] J. Choi, J. Yun, K. Jin, and Y. Kim, "Multi-news+: Cost-efficient dataset cleansing via llm-based data annotation," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 15–29.

[152] J. Yang, S. Yoon, B. Kim, and H. Lee, "Fizz: Factual inconsistency detection by zoom-in summary and zoom-out document," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 30–45.

[153] S. Iskander, S. Tolmach, O. Shapira, N. Cohen, and Z. Karnin, "Quality matters: Evaluating synthetic data for tool-using llms," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 4958–4976.

[154] O. Sainz, J. Campos, I. García-Ferrero, J. Etxaniz, O. L. de Lacalle, and E. Agirre, "Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 10 776–10 787.

[155] C. Deng, Y. Zhao, X. Tang, M. Gerstein, and A. Cohan, "Investigating data contamination in modern benchmarks for large language models," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 8706–8719.

[156] C. Yu, X. Liu, Y. Wang, Y. Liu, W. Feng, X. Deng, C. Tang, and J. Lv, "Gpt-nas: Neural architecture search meets generative pre-trained transformer model," *Big Data Mining and Analytics*, vol. 8, no. 1, pp. 45–64, 2024.

[157] H. Wang, Y. Gao, X. Zheng, P. Zhang, J. Bu, and P. S. Yu, "Graph neural architecture search with large language models," *Science China Information Sciences*, vol. 68, no. 12, p. 222103, 2025.

[158] H. Lyu, S. Jiang, H. Zeng, Y. Xia, Q. Wang, S. Zhang, R. Chen, C. Leung, J. Tang, and J. Luo, "Llm-rec: Personalized recommendation via prompting large language models," in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 583–612.

[159] R. Chen, T. Zhao, A. Jaiswal, N. Shah, and Z. Wang, "Llaga: large language and graph assistant," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 7809–7823.

[160] C. Lu, S. Holt, C. Fanconi, A. J. Chan, J. Foerster, M. van der Schaar, and R. T. Lange, "Discovering preference optimization algorithms with and for large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 86 528–86 573, 2024.

[161] H. Li, X. Yang, Z. Wang, X. Zhu, J. Zhou, Y. Qiao, X. Wang, H. Li, L. Lu, and J. Dai, "Auto mc-reward: Automated dense reward design with large language models for minecraft," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 426–16 435.

[162] W. Mao, J. Wu, W. Chen, C. Gao, X. Wang, and X. He, "Reinforced prompt personalization for recommendation with large language models," *ACM Transactions on Information Systems*, vol. 43, no. 3, pp. 1–27, 2025.

[163] T. Z. Xiao, R. Bamler, B. Schölkopf, and W. Liu, "Verbalized machine learning: Revisiting machine learning with language models," *arXiv preprint arXiv:2406.04344*, 2024.

[164] Y. Peng, S. Lin, Q. Chen, S. Wang, L. Xu, X. Ren, Y. Li, and J. Xu, "Chatgraph: Chat with your graphs," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 5445–5448.

[165] X. Zhou, X. Zhao, and G. Li, "Llm-enhanced data management," *arXiv preprint arXiv:2402.02643*, 2024.

[166] I. Dasgupta, C. Kaeser-Chen, K. Marino, A. Ahuja, S. Babayan, F. Hill, and R. Fergus, "Collaborating with language models for embodied reasoning," *arXiv preprint arXiv:2302.00763*, 2023.

[167] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," in *International conference on machine learning*. PMLR, 2023, pp. 3676–3713.

[168] J. Wang, Y. Liang, F. Meng, Z. Sun, H. Shi, Z. Li, J. Xu, J. Qu, and J. Zhou, "Is chatgpt a good nlg evaluator? a preliminary study," in *Proceedings of the 4th New Frontiers in Summarization Workshop*, 2023, pp. 1–11.

[169] W. Xu, D. Wang, L. Pan, Z. Song, M. Freitag, W. Wang, and L. Li, "Instructscore: Towards explainable text generation evaluation with automatic feedback," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 5967–5994.

[170] L. Zhu, X. Wang, and X. Wang, "Judgelm: Fine-tuned large language models are scalable judges," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 51 257–51 296.

[171] S. Kim, J. Shin, J. Jang, S. Longpre, H. Lee, S. Yun, R. Shin, S. Kim, J. Thorne, M. Seo *et al.*, "Prometheus: Inducing fine-grained evaluation capability in language models," in *International Conference on Learning Representations*, vol. 2024, 2024, pp. 29 927–29 962.

[172] S. Kim, J. Suk, S. Longpre, B. Y. Lin, J. Shin, S. Welleck, G. Neubig, M. Lee, K. Lee, and M. Seo, "Prometheus 2: An open source language model specialized in evaluating other language models," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 4334–4353.

[173] H. Joko and F. Hasibi, "Face: A fine-grained reference free evaluator for conversational recommender systems," *arXiv preprint arXiv:2506.00314*, 2025.

[174] Q. Lu, B. Qiu, L. Ding, K. Zhang, T. Kocmi, and D. Tao, "Error analysis prompting enables human-like translation evaluation in large language models," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 8801–8816.

[175] J.-H. Yang and J. Lin, "Toward automatic relevance judgment using vision–language models for image–text retrieval evaluation," *arXiv preprint arXiv:2408.01363*, 2024.

[176] P. Manakul, A. Liusie, and M. Gales, "Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models," in *Proceedings of the 2023 conference on empirical methods in natural language processing*, 2023, pp. 9004–9017.

[177] J. Cho, Y. Hu, J. Baldridge, R. Garg, P. Anderson, R. Krishna, M. Bansal, J. Pont-Tuset, and S. Wang, "Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation," in *International conference on learning representations*, vol. 2024, 2024, pp. 15 625–15 645.

[178] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in neural information processing systems*, vol. 36, pp. 46 595–46 623, 2023.

[179] P. Wang, L. Li, L. Chen, Z. Cai, D. Zhu, B. Lin, Y. Cao, L. Kong, Q. Liu, T. Liu *et al.*, "Large language models are not fair evaluators," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 9440–9450.

[180] J. Ye, Y. Wang, Y. Huang, D. Chen, Q. Zhang, N. Moniz, T. Gao, W. Geyer, C. Huang, P.-Y. Chen *et al.*, "Justice or prejudice? quantifying biases in llm-as-a-judge," *arXiv preprint arXiv:2410.02736*, 2024.

[181] M. Alaofi, P. Thomas, F. Scholer, and M. Sanderson, "Llms can be fooled into labelling a document as relevant: best café near me; this paper is perfectly relevant," in *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, 2024, pp. 32–41.

[182] S. Shankar, J. Zamfirescu-Pereira, B. Hartmann, A. Parameswaran, and I. Arawjo, "Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences," in *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, 2024, pp. 1–14.

[183] N. Farzi and L. Dietz, "An exam-based evaluation approach beyond traditional relevance judgments," *arXiv preprint arXiv:2402.00309*, 2024.

[184] R. Li, T. Patel, and X. Du, "Prd: Peer rank and discussion improve large language model based evaluations," *arXiv preprint arXiv:2307.02762*, 2023.

[185] H. He, H. Zhang, and D. Roth, "Socreval: Large language models with the socratic method for reference-free reasoning evaluation," in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 2736–2764.

[186] D. Luo, C. Feng, Y. Nong, and Y. Shen, "Autom3l: An automated multimodal machine learning framework with large language models," in *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024, pp. 8586–8594.

[187] Z. Yang, W. Zeng, S. Jin, C. Qian, P. Luo, and W. Liu, "Autommlab: Automatically generating deployable models from language instructions for computer vision tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 21, 2025, pp. 22 056–22 064.

[188] P. Trirat, W. Jeong, and S. J. Hwang, "Automl-agent: A multi-agent llm framework for full-pipeline automl," in *International Conference on Machine Learning*. PMLR, 2025, pp. 60 099–60 146.

[189] G. Chen, J. Chen, L. Chen, J. Zhao, F. Meng, W. X. Zhao, R. Song, C. Chen, J.-R. Wen, and K. Jia, "Toward autonomous long-horizon engineering for ml research," *arXiv preprint arXiv:2604.13018*, 2026.

[190] S. Guo, C. Deng, Y. Wen, H. Chen, Y. Chang, and J. Wang, "Ds-agent: Automated data science by empowering large language models with case-based reasoning," in *International Conference on Machine Learning*. PMLR, 2024, pp. 16 813–16 848.

[191] J. Xu, J. Li, Z. Liu, N. A. V. Suryanarayanan, G. Zhou, J. Guo, H. Iba, and K. Tei, "Large language models synergize with automated machine learning," *arXiv preprint arXiv:2405.03727*, 2024.

[192] Z. Jiang, D. Schmidt, D. Srikanth, D. Xu, I. Kaplan, D. Jacenko, and Y. Wu, "Aide: Ai-driven exploration in the space of code," *arXiv preprint arXiv:2502.13138*, 2025.

[193] J. Nam, J. Yoon, J. Chen, J. Shin, S. Arik, and T. Pfister, "Mle-star: Machine learning engineering agent via search and targeted refinement," *Advances in Neural Information Processing Systems*, vol. 38, pp. 116 692–116 712, 2026.

[194] Q. Huang, J. Vora, P. Liang, and J. Leskovec, "Mlagentbench: evaluating language agents on machine learning experimentation," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 20 271–20 309.

[195] J. S. Chan, N. Chowdhury, O. Jaffe, J. Aung, D. Sherburn, E. Mays, G. Starace, K. Liu, L. Maksin, T. Patwardhan *et al.*, "Mle-bench: Evaluating machine learning agents on machine learning engineering," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 50 466–50 494.

[196] G. Starace, O. Jaffe, D. Sherburn, J. Aung, J. S. Chan, L. Maksin, R. Dias, E. Mays, B. Kinsella, W. Thompson *et al.*, "Paperbench: Evaluating ai's ability to replicate ai research," in *International Conference on Machine Learning*. PMLR, 2025, pp. 56 843–56 873.

[197] H. Wijk, T. R. Lin, J. Becker, S. Jawhar, N. Parikh, T. Broadley, L. Chan, M. Chen, J. M. Clymer, J. Dhyani *et al.*, "Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts," in *International Conference on Machine Learning*. PMLR, 2025, pp. 66 772–66 832.

[198] 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*, vol. 36, pp. 11 809–11 822, 2023.

[199] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 16, 2024, pp. 17 682–17 690.

[200] A. Wu, K. Kuang, M. Zhu, Y. Wang, Y. Zheng, K. Han, B. Li, G. Chen, F. Wu, and K. Zhang, "Causality for large language models," *arXiv preprint arXiv:2410.15319*, 2024.

[201] Y. Tian, B. Peng, L. Song, L. Jin, D. Yu, L. Han, H. Mi, and D. Yu, "Toward self-improvement of llms via imagination, searching, and criticizing," *Advances in Neural Information Processing Systems*, vol. 37, pp. 52 723–52 748, 2024.

[202] L. Zheng, C. Li, H. Jia, and X. Zhang, "Reasoning paths as signals: Augmenting multi-hop fact verification through structural reasoning progression," *arXiv preprint arXiv:2506.07075*, 2025.

[203] M. A. Ferrag, N. Tihanyi, and M. Debbah, "Reasoning beyond limits: Advances and open problems for llms," *ICT express*, 2025.

[204] Y. Deng, C. Ye, Z. Huang, M. D. Ma, Y. Kou, and W. Wang, "Graphvis: Boosting llms with visual knowledge graph integration," *Advances in Neural Information Processing Systems*, vol. 37, pp. 67 511–67 534, 2024.

[205] D. Chen, R. Chen, S. Zhang, Y. Wang, Y. Liu, H. Zhou, Q. Zhang, Y. Wan, P. Zhou, and L. Sun, "Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark," in *Forty-first International Conference on Machine Learning*, 2024.

[206] M. R. Zhang, N. Desai, J. Bae, J. Lorraine, and J. Ba, "Using large language models for hyperparameter optimization," *arXiv preprint arXiv:2312.04528*, 2023.

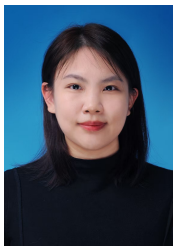
[207] J. Baumann, P. Röttger, A. Urman, A. Wendsjö, F. M. Plaza-del Arco, J. B. Gruber, and D. Hovy, "Large language model hacking: Quantifying the hidden risks of using llms for text annotation," *arXiv preprint arXiv:2509.08825*, 2025.

About the Authors

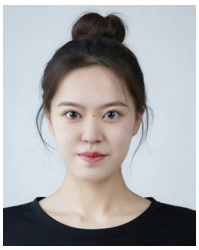
Xiang Ao is a professor at the State Key Laboratory of AI Safety, Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). He received the Ph.D. degree in computer science from ICT, CAS, in 2015, and the B.S. degree in computer science from Zhejiang University in 2010. His research interests include data mining, natural language processing, and trustworthy AI. He has published more than 100 papers in prestigious international conferences and IEEE/ACM Transactions.

Junhong Lian is pursuing the Ph.D. degree in computer science at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences (UCAS). He received the B.S. degrees from the Joint Program of Beijing University of Posts and Telecommunications (BUPT) and Queen Mary University of London (QMUL) in 2023. His research interests include personalized generation with LLMs, LLM-empowered AI research, and their applications in financial data mining.

Hanyang Li is pursuing the M.S. degree in computer engineering at the National University of Singapore (NUS). He received the B.S. degree in computer science from Harbin Institute of Technology (Shenzhen) in 2024. His research interests include LLM agents and AI for scientific research.



Siyi Wang is pursuing the M.S. degree in computer science at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences (UCAS). She received the B.S. degree in data science and big data technology from Shandong University in 2024. Her research interests include large language models and data mining.



Yiran Qiao is pursuing the Ph.D. degree in computer science at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences (UCAS). She received the B.Eng. degree in electronic engineering from Tsinghua University in 2022. Her research interests include LLM-augmented user modeling, risk assessment, and recommendation.

Yi Qiao is pursuing the M.S. degree in computer science at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences (UCAS). She received the B.S. degree in computer science from UCAS in 2023. Her research interests include user behavior modeling and recommender systems powered by LLMs.

Jiaqi Xu is pursuing the Ph.D. degree in computer science at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences (UCAS). He received the B.S. degree in computer science from East China Normal University (ECNU) in 2025. His research interests include LLM agents and graph machine learning.

Qing He received the B.S. degree in mathematics from Hebei Normal University, China, in 1985, the M.S. degree in mathematics from Zhengzhou University, China, in 1987, and the Ph.D. degree in fuzzy mathematics and artificial intelligence from Beijing Normal University, China, in 2000. He is currently a professor at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences (UCAS), China. His research interests include data mining and machine learning.

Xueqi Cheng is a professor at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). He received the Ph.D. degree from ICT, CAS, in 2006. He is currently the Director of the State Key Laboratory of AI Safety, ICT, CAS. His research interests include network data science, big data systems, social computing, Web search, and data mining. He has published more than 300 papers in premier international conferences and IEEE/ACM Transactions. His papers have received seven Best Paper Awards at top-tier international conferences.