DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

ORNL/TM-2025/4188

LLMs for Mfg.—On the State of Large Language Models and Applications to Manufacturing

William Halsey[1]
Michael Sprayberry[1]
Vincent Paquit

September 2025

[1] Equal contribution

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2025/4188

Manufacturing Sciences Division &
Electrification and Energy Infrastructure Division

LLMS FOR MFG.—ON THE STATE OF LARGE LANGUAGE MODELS AND APPLICATIONS TO MANUFACTURING

William Halsey
Michael Sprayberry
Vincent Paquit

September 2025

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831
managed by
UT-BATTELLE LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

ABBREVIATIONS

AM	Additive Manufacturing
LLM	Large Language Model
AI	Artificial Intelligence
ML	Machine Learning
SOTA	State of the Art
RAG	Retrieval Augmented Generation
NLP	Natural Language Processing
NN	Neural Network
RNN	Recurrent Neural Network
GPT	Generative Pretrained Transformer
AGI	Artificial General Intelligence
FFN	Feed Forward Network
LoRA	Low-Rank Adaptation
RL	Reinforcement Learning
RLHF	RL with Human Feedback
RLAIF	RL with AI Feedback
API	Application Program Interface
MOF	Metal-Organic Framework
KG	Knowledge Graph
MKG	Materials KG
LFAM	Large Format AM

ABSTRACT

Additive Manufacturing (AM), referred to as 3D printing, has emerged as a key pillar of Industry 4.0 enabling layer-by-layer fabrication of intricate geometries from CAD models. In parallel, Large Language Models (LLMs), deep learning models for natural language generation trained on vast text corpora, have demonstrated unprecedented capabilities in understanding and generating human-like text. The convergence of these trends opens new opportunities at the intersection of AM and AI/ML, where LLMs can assist engineers and researchers in design, manufacture planning, and knowledge discovery. Recent academic work has begun to explore LLM applications in AM and adjacent fields, such as material science, mechanical engineering, and design for additive manufacturing. This exploration ranges from intelligent process planning to domain-specific knowledge retrieval. This survey provides a comprehensive review of current developments, focusing on peer-reviewed literature contributions that apply, adapt, and advance LLMs in general and domain-specific domains. We analyze state-of-the-art (SOTA) techniques, such as fine-tuning foundational models for specific domains, retrieval-augmented generation (RAG) pipelines, knowledge graph integration, and delve into the architectures and evaluation methods employed. The goal of this survey is to inform researchers and practitioners of the current capabilities and limitations of LLMs in general and in domain-specific applications, and to outline how these models are being tailored to meet the requirements of these applications.

1. INTRODUCTION

1.1 Brief History

Large language models (LLMs) are the pinnacle of decades of work in language systems and natural language processing (NLP). However, these modern models differ significantly from early attempts and systems for computational linguistics. From the earliest days of computation, the Turing test posited that sufficiently complex computation could enable and exhibit apparent intelligence through language comprehension and generation. In the following decades several seminal demonstrations were performed. A chatbot named ELIZA was created that attempted to emulate a psychotherapist, and a system called SHRDLU demonstrated how a computer program could take in basic natural language prompts to control a virtual agent. An early and pervasive goal was to create a computer program that could perform automatic language translation, and in 1954 a set of experiments performed by IBM and Georgetown demonstrated the translation of several dozen Russian sentences to English. With the goal of computational translation in mind, research in computational linguistics grew through 1980s. These early efforts focused on defining formal and highly refined grammar and syntax rules that dictated how text should be translated from one language to the other. This paradigm was rigid and difficult to scale as rules and algorithms for translating text were manually and painstakingly curated and programmed.

During the 1990s, research began to shift towards statistical and probabilistic methods for translation. Instead of relying primarily on grammar and syntax, these methods began focusing on data—developing translation algorithms based on probabilities evidenced in corpora of translated texts. Later, advancements in neural networks (NN) led to yet another shift in NLP research. As compute capabilities grew and neural network architectures became more advanced, NLP researchers began to employ them more. These NN seemed able to capture the relationships between words as opposed to more basic bag-of-word approaches. Text could be broken down into small fragments, called tokens, that could be represented by vectors of numbers, and neural networks would learn relevant word associations and patterns from data during training—no more need for rigid, a priori rules and grammars or explicit statistical analysis. However, the class of NN architectures that was most amenable to language at the time, recurrent neural networks (RNNs), was also the slowest to train. The recurrence—meaning that an output or prediction from the NN depended upon the previous prediction—was simultaneously a useful feature and a stumbling block for scalability.

The next breakthrough was the advent of the Transformer architecture in 2017 which eliminated the burdensome recurrence paradigm while still capturing relationships between words. From then on, with those requisite techniques in place, the astounding progress in NLP has been an exercise in continual scaling up in compute power, training data, and model size. The release of ChatGPT on November 30, 2022 was a watershed moment, and large language models (LLMs) suddenly became part of common parlance. In the few short years since there has been a proliferation of research into LLMs and other related large models as evidenced by survey papers on LLM research[1], [2], [3], their applications broadly[4], [5], [6], and specifically to manufacturing[7], [8].

1.2 Key Concepts

The most prevalent LLMs belong to a group called generative pre-trained transformers (GPT). Each of those terms denote an important concept for understanding how LLMs function; each will be described in turn in reverse order. First, the transformer architecture was introduced by Vaswani et al.[9]. As previously stated, the advent of the transformer architecture was a significant development in NLP and towards the creation of LLMs, and today, the vast majority of LLMs are based on the transformer architecture, shown in Figure 1.

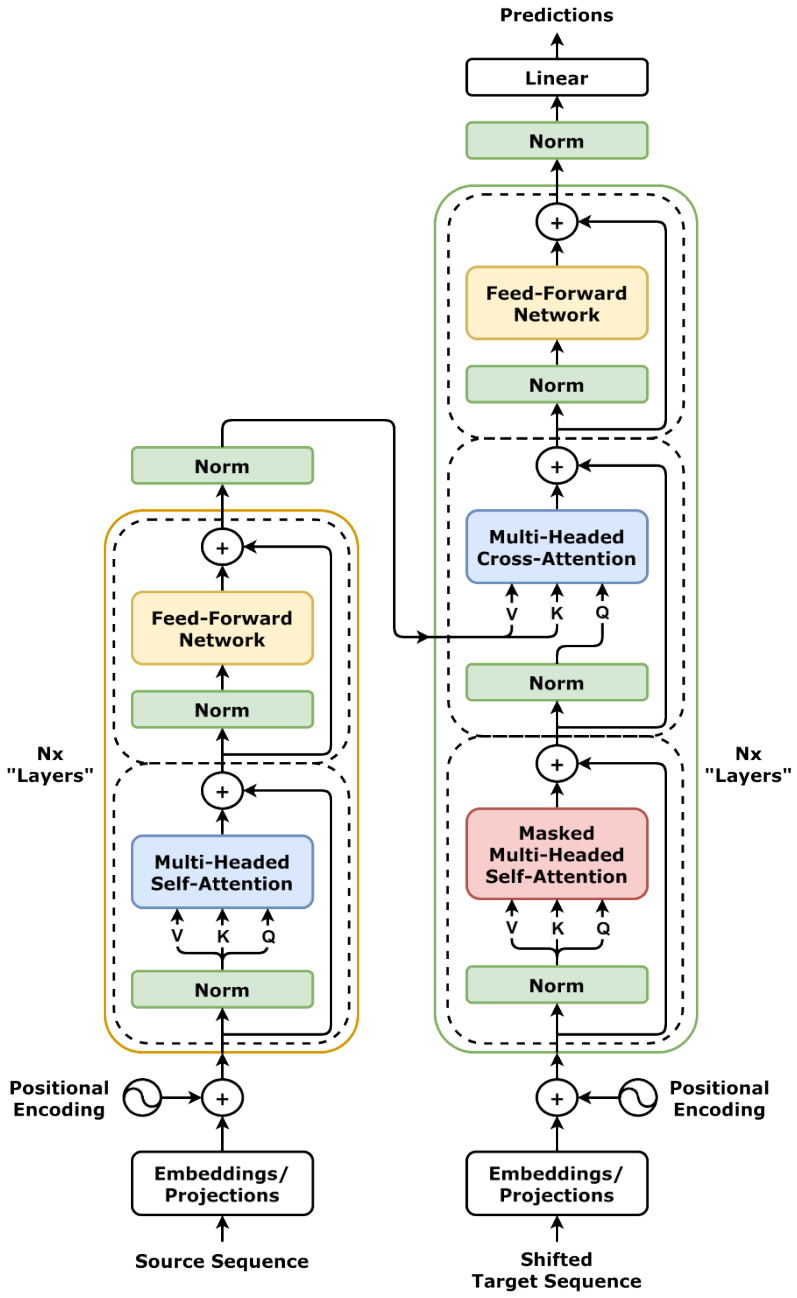

Figure 1: Transformer architecture as introduced in Viswani et al.[9].

The *transformer* modules, outlined in green and yellow in the figure, consist of attention modules and dense, feed-forward layers. The attentions mechanism was first introduced Bahdanau et al.[10], and at a high level, it allows the network to learn important relationships within a sequence: between different words, or tokens, in the input and between the abstract encodings in the deeper layers. With the relevant relationships between tokens in the series emphasized from the attention mechanism, the feed-forward layers roughly serve to allow the network to learn to encode the meaning of the string of input data. Increasing the numbers of weights within the attention and feed-forward layers within the transformer module and stacking many transformer modules in series, effectively allows the network to learn relationships across longer strings of text as well as learn and encode more complex and varied concepts.

The architecture grants LLMs robust capabilities to seemingly understand complex and varied concepts; however, this only happens after an LLM has learned them through a *pretraining* process. Modern LLMs are trained on a vast amount of textual data from a wide variety of sources and disciplines. These sources can include books, news articles, social media, academic literature, source code, etc. that are collected from publicly available sources on the internet as well as proprietary datasets. In general, the pretraining process happens in a self-supervised fashion. Simply put, the model is iteratively passed sections of text from the training set with the goal of predicting the next word in the sequence. This process is repeated an extraordinary number of times, and through leveraging traditional NN training techniques such a gradient descent and back propagation, the network learns to encode information within a sequence of text to maximize the ability to predict the next word in the sequence. The output of the network is a probability distribution across the model's vocabulary, and there are different methods for calculating the prediction performance.

Once trained, the models may be employed in various ways. Based on how they are trained, as text predicters, they are readily able to *generate* new text. Given new or arbitrary text, an LLM will predict the probable next word in the sequence. The model can then be given the original prompting text and its previous prediction and predict the next word and so on. To add some variety and stochasticity to these generative models, the model does not always select the most probable word in the sequence. Instead, it chooses the next word by randomly sampling across the probability distribution of the vocabulary based on the input text. LLMs are susceptible to a phenomenon called hallucination since the underlying mechanisms of text generation are probability predictions of next words based on the input and sampling from the vocabulary based on those probabilities. The output they generate is not explicitly oriented toward creating factual statements but are instead based on creating output that seems probable based on the data that it was trained on. Hallucinations, then, are generated responses from LLMs that appear plausible and seem coherent but that contain inaccurate or false information.

Generative pretrained transformers, based on their underlying principles, have demonstrated astounding emergent properties that, with only a few additions, enable them to appear to understand complex natural language queries, remember some facts from training, generate novel content, and synthesize responses across a diversity of disciplines. As such, research to both further their development and apply them has grown exponentially. In the remainder of this report, we will explore these research trends, applications of LLMs for manufacturing, as well as potential impacts and risks.

2. CURRENT LLM TRENDS AND RESEARCH

2.1 LLM KNOWLEDGE INTEGRATION AND DOMAIN ADAPTATION METHODS

Using an LLM off the shelf may or may not yield desired results for specific tasks or domain-specific applications. This could be due to a mismatch in the "knowledge" of the LLM—what information it was trained on and how well it learned—and the desired task. As such, much research has focused on methods to integrate new or relevant knowledge into LLMs. A survey by Feng et al.[2] breaks down the knowledge integration research into two categories: knowledge editing and retrieval augmentation. We also add a third category: model scaling as shown in Figure 2. In this section we survey the most common trends, methods, and augmentations.

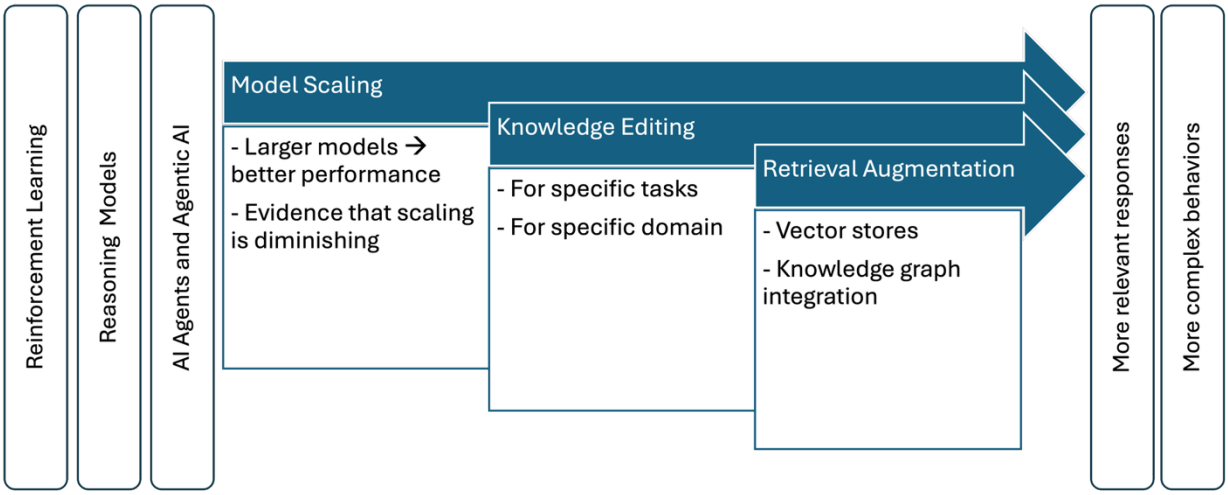

Figure 2: Methods for knowledge integration and adapting LLMs for specific tasks or domains (center) and augmenting LLMs (left) to produce more relevant responses and exhibit more complex behaviors.

2.1.1 Model scaling

The capabilities that an off-the-shelf model possesses for a specific domain is significantly affected by the size of the model. A general and important trend in LLM research is that models tend to exhibit better performance for both general and domain-specific applications as the size of the model and the number of resources used to train and run it increases[11]. The "size" of the model refers to the number of learned parameters the model has. Early advances in the LLM capabilities were realized, in large part, simply by releasing increasingly large models. Figure 3 shows the number of estimated parameters for each generation of GPT models from OpenAI.

However, the extent and prospects of this trend have been called into question due to both philosophical and practical limitations. The implication is that the benefits of scaling may have plateaued and that continuing to increase model size will provide diminishing benefits to model capability[12]. Furthermore, several reports and practitioners question whether increasing model scale alone can yield artificial general intelligence (AGI) since increasing model size has not alleviated problems when providing answers to queries that require logic or reasoning[13], [14].

Possible philosophical limits aside, larger models require more data for training and compute resources for training and inference. From the data perspective, current LLMs are already pre-trained on large swathes of the publicly available internet as well as other large proprietary datasets[15]. Additionally, finding other data sources with sufficient quality and quantity is non-trivial and can be costly. Some estimate that a "data cliff"—the point where no additional, high-quality data is available—

may be reached before 2030[16], [15]. Furthermore, attempts at leveraging synthetic data—data that is generated by other LLMs—has yielded inconclusive results[17], [18], [19].

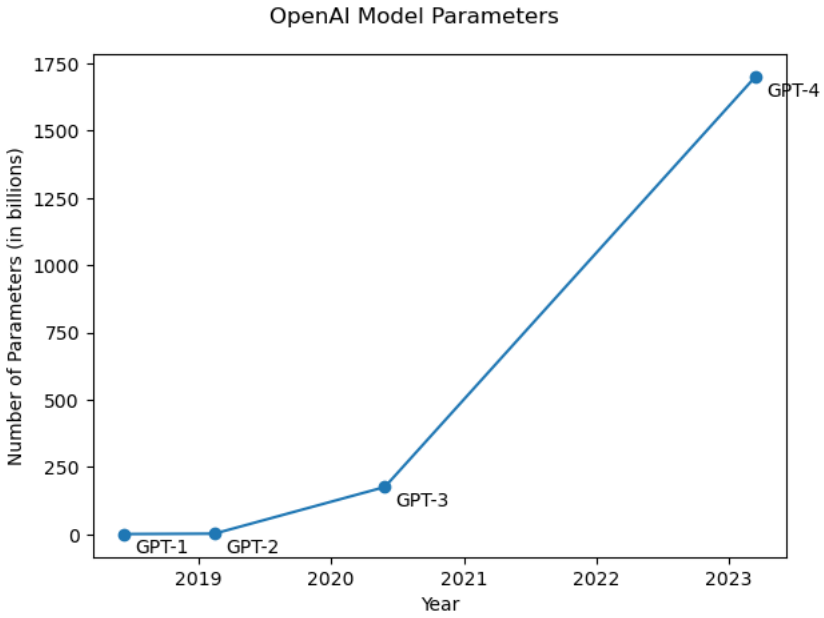

Figure 3: Approximate number of parameters of different versions of OpenAI's GPT models over the year.

2.1.2 Knowledge Editing

Knowledge editing seeks to introduce new and relevant knowledge to a model to improve the relevance and accuracy of responses. Due to the scale and number of parameters in LLMs, knowledge editing research is largely focused on methods for updating minimal or no model parameters[2]. In their survey of knowledge integration methods, Feng et al. outline two broad knowledge editing categories: input editing and model editing.

Input editing does not require any updates to the model and instead relies on additional information that is provided by either the user or a software tool. This information is added directly to the prompt and allows for the integration of previously unknown knowledge into the model. This approach is particularly valuable due to the low overhead needed to implement and its flexibility. However, knowledge that is introduced in this fashion is not persistent and may have to be reintroduced multiple times once the context gets too large. Input editing is not only valuable for introducing new information to LLMs but also for inducing more complex reasoning behaviors. In-context learning methods consist of a constructing prompts that contains one or several example queries and correct responses that are similar to the specific query of interest[20]. Wei et al.[21] demonstrated how a form of input editing that they called "chain of thought" significantly improved performance on math and symbolic reasoning problems—surpassing state of the art performance for several benchmarks.

Model editing is the second form of knowledge editing, which aims to integrate new knowledge and remove irrelevant and incorrect knowledge from a model by updating the model's parameters. Feng et al. introduce three broad categories for knowledge editing: knowledge plug-in, locate-then-edit, and overall editing[2]. Knowledge plug-in methods aim to integrate knowledge by adding and training small, modular accessory networks. These networks are often inserted either before or after feed forward network (FFN) layers of an LLM and served to update those outputs, based on training from a domain-specific datasets, to generate more relevant responses. Locate-then-edit techniques are more complex and involve locating the LLM parameters that represent certain knowledge and updates those during retraining. Finally, overall editing comprises methods that either update some or all of a model's

parameters or train an accessory network that, when combined with the base network, integrates updated knowledge all without attempting to explicitly determine where knowledge is located. Fine tuning is the most straightforward approach and is a type of transfer learning. It seeks to tune an off-the-shelf model's parameters to make them more suitable for specific tasks or domains by employing additional training iterations on relevant datasets. While this is a viable option for smaller models, it can be too cumbersome for larger models in practice. Hu et al.[22] introduced low-rank adaption (LoRA) that represents a method of training an accessory network, while leaving the base model parameters unchanged, for knowledge integration. The benefits of this method are reduced memory and training burden as well as the ability to switch out the accessory network as needed for different domain-specific tasks. See Feng et al.[2] for more information about these categories.

2.1.3 Retrieval augmentation

Retrieval augmentation, or retrieval augmented generation (RAG), entails coupling an LLM with external data sources using a retrieval system and often leaving the base LLM unchanged. Knowledge integration occurs by supplying the LLM with additional information, or context, to assist with synthesizing the final response. Since the model parameters are left unchanged RAG can be considered a sophisticated form of input editing; however, the number of considerations for RAG systems entails that they warrant further exposition. Feng et al.[2] outline several questions that must be addressed for effective implementations of RAG systems.

- When is retrieval augmentation required?
- How are relevant documents retrieved?
- How are documents utilized by the LLM? And
- How are knowledge conflicts between retrieved documents resolved?

Each of these questions has garnered its own research. Interestingly, research related to each of the first two questions consists of two types: methods that employ traditional modeling or metrics and methods that employ LLMs to accomplish the task. Table 1 below gives a description of the approaches with some accompanying example implementations that are discussed in Feng et al.[2].

Table 1: Description of methods for 1) determining if retrieval is required and 2) retrieving information and examples.

		Retrieval required?	**How to Retrieve?**
Metric-based	*Description*	Calculate a metric based on input (e.g., subject popularity or prevalence) or model output (e.g., uncertainty) to determine if retrieval is needed based on a threshold	Encode and calculate distance between query and available documents to determine relevance during retrieval
	Methods	*FLARE*[23], popularity metric[24]	*Query2doc*[25], *Dense retriever*[26], *TF-IDF*
LLM-based	*Description*	Ask the LLM directly to either 1) determine if it can answer a given query or if it requires additional information or 2) if retrieved information is relevant	Ask an LLM to either 1) choose relevant documents by generating identifier strings or 2) generate the relevant document(s) to serve as context
	Methods	1) Self knowledge[27], [28] 2) NLI[29]	1) *GENRE*[30], *DSI*[31], *SEAL*[32] 2) Generate-then-read[33], *RECITE*[34]

Once additional context is retrieved it may be utilized in several distinct ways by an LLM. A common approach is to supply the LLM with the retrieved context from the RAG system along with the original query. The LLM then uses the information from the additional context to synthesize more relevant and accurate responses. This approach is both a simple and effective way to integrate knowledge into a pre-trained LLM. Alternatively, the retrieved context may be used to verify and update an LLM's response. This posteriori verification approach may be used for both simple LLM implementations as well as reasoning LLMs. Finally, when implementing RAG systems, one must be aware of the possibility of knowledge conflicts—conflict between either the LLM's learned knowledge and context and between documents that are provided as context. Feng et al. note that most published research focuses more on identifying knowledge conflicts as opposed to resolving knowledge conflicts[2].

In addition to the theoretical considerations listed above, other practical considerations exist when implementing RAG systems. These include the source and structure of external data, if the data requires any preprocessing, and constraints on length of context that can be processed by the LLM. Possible data sources and interfaces can include knowledge graphs, document stores, search engines and even the internet. Document preprocessing can include choosing schemes to section them into chunks and how to index or encode them for retrieval. The number of design decisions for RAG systems entails that no one-size-fits-all solution exists.

2.2 OTHER AUGMENTATIONS

In addition to the core trends in knowledge integration previously discussed, several other methods and augmentations have been developed and employed to produce more relevant outputs and more complex behaviors from LLMs and LLM-based systems that will briefly be introduced. While other topics could surely be discussed here—such as multimodal models and alternative NN architectures, distillation and quantization—we have included reinforcement learning, reasoning models, and AI agents and agentic AI as essential to understanding some of the latest trends in LLM research. Each research topic is dynamic and technical, so the following descriptions aim to give a high-level and intuitive understanding.

2.2.1 Reinforcement learning

Starting in 2022, techniques from reinforcement learning (RL) have been employed to improve the performance of LLMs, and generally this is utilized after the initial pretraining of the model. The variety of methods and open problems are already captured in several surveys[35], [36]. In general, RL is a set of methods for training systems to perform actions in order to maximize rewards. Implementing RL for LLMs can be quite complex. Rewards are based on the quality of responses and serve to align responses with user preferences and expectations across multiple possible aspects such as relevance, factuality, readability, etc.

The process entails having an LLM generate multiple responses to prompts. A human can then score or rank those responses. Once a sufficient dataset is generated, a reward model may be trained to predict the quality scores of future responses. With a trained reward model, the system is able to 1) take a query, 2) generate several responses, 3) automatically receive scores and rankings for those responses from the reward model, and 4) update the weights of the LLM to make the preferred response more likely. When the quality determinations are made by humans, the process is referred to a RL with human feedback (RLHF).

However, the need for human annotations and rankings requires significant costs, scales poorly, and is subjective, so research also covers how to replace human feedback with feedback from other LLMs known as RL from AI feedback (RLAIF). External, pretrained LLMs may be used to generate the original prompts and evaluate and rank the quality of responses that facilitate the training of a reward model—a form of *distillation*—or they are sometimes used directly as the reward model. The added complexity of

employing a reward model has prompted yet another line of research into techniques that avoid utilizing a reward model altogether[35].

2.2.2 Reasoning models

As the name implies, reasoning models are models that have been specially tailored to emulate human reasoning when responding to a query as opposed to naively and simply generating a probabilistic response. This line of research grew out of earlier chain-of-thought work that demonstrated that LLMs could mimic and apply examples of reasoning to provide more accurate and clear answers to reasoning and mathematical tasks[21]. Again, the research in this topic is vast and varied; however, several distinct categories of techniques have emerged: model fine-tuning, reinforcement learning, and inference methods (also referred to as test-time scaling)[37], [38].

After initial pretraining, reasoning models are often finetuned on datasets that include queries and responses that explicitly outline a thought process. In this respect, some reasoning capabilities can emerge as a matter of task adaptation—namely for reasoning itself. However, this behavior is limited by the size of the model with models containing fewer than ten billion trainable parameters struggling to manifest any emergent reasoning abilities[38]. Like other techniques, the dataset used for finetuning may either be generated and annotated by humans or by other LLMs.

As previously discussed, reinforcement learning can be employed to improve an LLM's responses to align with expectations and certain characteristics. To help elicit reasoning behaviors in LLMs, RL is used to align their responses for factuality and interpretability—that is the response contains a description of a thought process that was followed to arrive at an answer. This alignment allows an LLM to more readily manifest seemingly reasoned responses and handle queries that require reasoning. Both RLHF and RLAIF may be employed.

Finally, reasoning models often employ additional methods at inference, or test, time when they are generating responses. These methods involve either parallel scaling, sequential scaling, search-based methods, or some combination thereof[37]. Parallel scaling methods involve the LLM generating multiple independent responses to a query, and the intermediate or final response consists of the "best" one or of a combination of the individual responses. Thus, it requires the LLM or some other mechanism to evaluate the answers. Sequential scaling is comprised of the LLM generating a response, evaluating it, and refining it until some criteria is met. Search methods are an amalgam of the two whereby the LLM generates partial responses, discards the least appropriate ones, builds on the more appropriate ones, and iterates in that fashion until a suitable answer is generated.

Reasoning models have been shown often to perform markedly better than non-reasoning models. However, they can incur significant overhead while generating responses, especially those that employ test-time scaling methods. This is due to the additional compute required to generate full or partial responses that will ultimately get discarded. Additionally, recent work has shown that these models may actually underperform non-reasoning LLMs at some basic tasks as well as fail to generate accurate responses for sufficiently complex tasks[39].

2.2.3 AI agents and agentic AI

Beyond leveraging LLMs for question answering, research is evolving in two distinct yet complementary directions—namely development of AI agents and agentic AI. Initially, research evolved around the concept of AI agents which is defined by Sapkota et al.[40] as "modular systems driven and enabled by LLMs … for task-specific automation." AI agents are not new, however. In fact, the goal of artificial intelligence research has always been to design algorithms, that may or may not leverage machine learning models, for task automation. The key shift recently is the apparent emergent properties of LLMs for natural language processing which provides flexibility and adaptability to changing context and inputs. These emergent properties and the breadth of information that LLMs are trained on also mean that they are much more generalizable—tunable to many tasks. This makes them powerful engines for AI

agents by simplifying automation algorithms and allowing for adaptability. Also, by design they generate output in the form of text, which means they may be readily leveraged to interface with software tools. This is beneficial in two ways, the LLM is able to query application program interfaces (APIs) either to get additional information to accomplish a task or to execute a process that is part of a larger automation pipeline leveraging libraries such as LangChain[41] and Easytool[42].

More recently, agentic AI has a growing share of interest and "[represents] a paradigm shift marked by multi-agent collaboration, dynamic task decomposition, persistent memory, and coordinated autonomy."[40] Research in this area explores how multiple LLMs may be coordinated and utilized in conjunction to accomplish more complex tasks, and it is a natural progression built upon the concept of AI agents. Each LLM is given a role that they will be responsible for within the team. For example, when a command is given to the team of agents, one will be responsible for understanding the meaning of the command and determining what subtasks must be done to accomplish the overall goal. It will then generate commands for each subtask and send each one to other "agents" that are responsible for executing the instruction and communicating the results. As such, much research in this area pertains to develop methods for coordination and communication among the agents.

3. MANUFACTURING APPLICATIONS

3.1 LLM APPLICATIONS IN ADDITIVE MANUFACTURING AND ENGINEERING DOMAINS

Current peer-reviewed literature demonstrates a variety of fields LLMs are being leveraged in, including engineering, medicine, and material science fields. A recent survey by Li et al. [43] shows a proliferation of works leveraging LLMs in these areas over the last few years. Table 2 provides an overview of representative research works, their focus domains, LLM approaches, and key findings. In this section, we discuss these applications in thematic groups, including knowledge assistance, design and planning, process monitoring and control, code generation and debugging, and materials science knowledge management. We differentiate between foundation models, general purpose LLMs like GPT-4o or Llama, and application-specific implementations (models or pipelines adapted or fine-tuned for domain-specific tasks).

3.2 KNOWLEDGE ASSISTANCE AND INFORMATION RETRIEVAL

A common use of LLMs is as an intelligent assistant to aid in navigating technical information. In the AM domain, Chandrasekhar et al.[44] introduced AMGPT, a specialized language model for contextual querying in additive manufacturing. Rather than developing an LLM from scratch, AMGPT employs a RAG approach: a lightweight LLM (Llama2-7B) that is coupled with a domain-specific document database to answer queries with up-to-date, detailed information. In the implementation approximately 50 authoritative AM papers and textbooks were indexed, using LlamaIndex[45], so that when a query is posed relevant text embeddings are retrieved and fed into the LLM for informed text generation[44]. The design injects specialized knowledge (e.g. material properties of novel alloys, process parameters for specific technologies and systems) into the generative process. Evaluation from domain knowledge users reported that LLM+RAG model yielded more specific and accurate answer for AM queries than the general model like GPT-4 without domain-specific adaptation[44]. By grounding the LLM in curated AM literature, AMGPT, was able to provide detailed manufacturing instructions and materials insights that GPT-4 could not provide[44]. This highlights a key strategy for domain adaption, rather than, or in addition to, fine-tuning a foundation model on specialist text, one can use retrieval of relevant context to guide a general model.

Similar LLM-based knowledge aids are appearing in adjacent fields. For example, materials science where researchers have constructed large knowledge graphs from literature and used LLMs to query them in natural language[46]. An et al.[47] developed KGQA4MAT, a benchmark, and interface where ChatGPT translates user questions into formal queries on a material knowledge graph, specifically for metal-organic frameworks[47]. This system allows domain experts to ask complex questions, in natural language, about the material structures or properties and get answers by querying an underlying knowledge graph, essentially combining the flexibility of an LLM with precision of a structured database. In experiments, GPT-4 could successfully interpret 161 challenging questions (with comparisons, aggregations, etc.) into SPARQL queries on the metal-organic framework (MOF) knowledge graph (KG), demonstrating the feasibility of natural-language-to-database interaction for materials engineering[47]. These examples illustrate how LLMs serve as bridges to technical knowledge, by either directly encoding a literature corpus or by acting as an interface to external knowledge bases (documents, graphs, etc.).

The integration of knowledge graphs with LLMs is particularly promising for engineering domains. Knowledge graphs provide factual precision and coverage of known data, while LLMs provide flexible understanding of user intents and language[48]. Ye et al.[49] took this further by using LLM-based text mining to build a Materials Knowledge Graph (MKG) from scratch[49]. The authors processed a decade of materials science papers with NLP techniques (named entity recognition, relation extraction), enhanced by an LLM to improve extraction of complex relationships, to populate a graph

Table 2: LLM applications to manufacturing.

Reference (Year)	Domain / Task	LLM & Methodology	Data / Knowledge Base	Key Results
Chandra-sekhar et al. (2024)[50]	Metal AM literature Q&A (knowledge navigation)	Llama2-7B with Retrieval-Augmented Generation	~50 AM papers & textbooks (domain corpus)	RAG-specialized model produced more specific answers to AM queries than a general GPT-4, leveraging up-to-date domain knowledge. Accelerated researcher access to detailed materials/process info.
Badini et al. (2023)[48]	FFF process troubleshooting (G-code optimization)	ChatGPT (GPT-3.5) via prompt-based queries	None (zero-shot; existing G-code knowledge)	Demonstrated ChatGPT can analyze and suggest optimizations for G-code parameters (e.g. temperature, speed) to mitigate issues like warping and stringing in polymer 3D printing. Showed potential for real-time print parameter tuning.
Xie et al. (2024)[51]	LFAM print strategy design (thermal planning)	Custom Transformer model (fine-tuned)	Historical print temperature data (sensor logs)	Transformer-based model accurately predicted layer-wise temperature profiles for new geometries and optimized inter-layer cooling times, improving print quality and efficiency beyond fixed scheduling methods.
Jadhav et al. (2024) [52] – *LLM-3DPrint*	FDM process monitoring & control (defect auto-correction)	Multi-agent GPT-4 (vision + text)	Real-time layer images; printer sensor data	A multi-agent LLM framework detected common print defects (e.g. extrusion errors, warping) from images and printer logs and generated corrective actions autonomously. Matched or surpassed human experts in identifying and fixing failures without intervention.
Fang et al. (2024)[53]	AM defect detection (few-shot learning)	GPT-3 (few-shot prompt as classifier)	Small labeled dataset of build defects	Treated defect detection as a language task, allowing an LLM to classify defect types with only a few examples prompts. Achieved quality inspection with minimal training data, highlighting LLM generalization in identifying anomalies (as reported in China Automation Congress 2024).
Makatura et al. (2023)[8]	Design & manufacturing planning (CAD/CAM pipeline)	GPT-4 (general foundation model)	None (zero-shot on design text prompts)	Found GPT-4 can reason about high-level design requirements and incorporate textual instructions into generating manufacturing process plans. Showed promise in automating parts of CAD/CAM workflows (e.g. suggesting design modifications from functional descriptions).
Ye et al. (2024)[49]	Materials science knowledge graph construction	GPT-3-based NLP pipeline (LLM-assisted IE)	10 years of materials papers (text corpus)	Built a Materials Knowledge Graph (MKG) with 162k nodes and 732k edges by using LLM-driven extraction of entities/relations from literature. The MKG enables link prediction and querying, accelerating new material discovery by integrating dispersed knowledge.

Table 2 continued

Reference (Year)	Domain / Task	LLM & Methodology	Data / Knowledge Base	Key Results
An et al. (2024)[47]	Querying materials knowledge (MOF domain Q&A)	ChatGPT (GPT-4) translating NL to KG queries	MOF-KG (integrated database + literature KG)	Developed a benchmark of 644 questions on MOF materials and showed ChatGPT can translate natural language questions into structured queries (SPARQL) on the knowledge graph with high accuracy. Enabled a natural language interface for material scientists to query complex data.
Jignasu et al. (2023)[54]	G-code debugging and editing (code comprehension)	Multiple LLMs (GPT-4, GPT-3.5, Claude 2, etc.)	Simulated G-code error scenarios (evaluation set)	Extensive evaluation found GPT-4 identified G-code errors immediately with minimal prompt guidance, outperforming other models (Claude-2, Llama2-70B, etc.) in correctness. However, all models struggled to generate complete correct G-code for complex shapes (e.g. incomplete infill patterns) due to context and reasoning limits.

with hundreds of thousands of entities and links[49]. The resulting MKG encodes information on material names, formulas, properties, and applications in a structured form. This graph can then be used in conjunction with LLMs for tasks like hypothesis generation or targeted information retrieval. The synergy of LLMs and knowledge graphs exemplifies a cutting-edge technique for knowledge-rich domains, LLMs can help populate and query structured knowledge bases, which in turn can augment LLMs to produce more accurate and traceable outputs.

3.3 DESIGN AND PLANNING IN ENGINEERING

In mechanical engineering and design for manufacturing, LLMs are being explored as co-pilots for the creative and planning stages for the workflow. Makatura et al.[8] conducted an extensive study and using GPT-4 to automate parts of the computational design and manufacturing pipeline. Their work demonstrates a powerful foundation model like ChatGPT-4 can interpret high-level design specifications and reason about design, intent constraints, and processes. For instance, ChatGPT-4 was able to suggest design modifications given a textual description of a product's functional requirements, and incorporate manufacturability considerations, like recommending fillets for sharp corners, in its output. It could also generate pseudo-code or scripts to interface with CAD tools, indicating an ability to span from natural language specs to machine-interpretable instructions[8]. These findings highlight that modern LLMs possess a form of embedded engineering knowledge, likely gleaned from public design documents and manuals in their training data, which can be elicited with proper prompting. However, they also note limitations: ChatGPT-4, while good at high-level reasoning, may lack accuracy in detailed numeric computations or geometric reasoning without additional help[8]. To address such gaps, researchers are investigating how LLMs might work in tandem with other models or tools in the design process.

Another line of work in design optimization uses transformer-based architectures to model engineering physics and aid planning. Xie et al.[51] developed a transformer model for offline printing strategy design in Large-Format Additive Manufacturing (LFAM). In LFAM (which produces large parts via extrusion), one critical design decision is how long to wait between layers (i.e., layer time) to ensure proper cooling and bonding. This involves predicting the temperature evolution of the printed layers – a complex physics problem often tackled by case-specific simulations. Xie et al. leveraged a data-driven

approach: they trained a transformer on historical thermal data from previous prints so that it can predict temperature profiles for new part geometries in real-time. The transformer, effectively functioning as a specialized LLM that "speaks the language" of thermal behavior, outputs temperature predictions across the print surface. Those predictions are fed into an optimization algorithm to choose optimal layer times. This approach yielded a highly accurate predictor of layer temperatures and allowed optimizing cooling schedules far beyond traditional fixed-time heuristics. The result was improved print quality and efficiency, showcasing how advanced ML architectures can learn domain physics and support design decisions. While this transformer model is not generating human language, it shares technical kinship with LLMs and exemplifies fine-tuning foundational models for domain-specific tasks—in this case, adapting a transformer to capture manufacturing physics. It illustrates a broader trend: using the sequence modeling power of transformers to encode design or process sequences (such as toolpaths, thermal histories, etc.) and to make predictions or suggestions that aid engineering planning.

3.4 PROCESS MONITORING, QUALITY CONTROL, AND DEFECT MITIGATION

Ensuring quality in additive manufacturing is challenging, as printing processes can suffer from defects (e.g., filament mis-extrusion, warping, delamination) that require expert intervention. A novel application of LLMs in this arena is to make them part of the control loop for monitoring and correcting errors. Jadhav et al.[52] introduced LLM-3DPrint, a multi-agent system that deploys LLMs to monitor an FDM 3D printer and autonomously address print failures. In their setup, a camera captures images of the print after each layer and an LLM (with vision capabilities, e.g. a GPT-4 variant that accepts image input) analyzes these images in conjunction with printer telemetry. The LLM identifies any emerging defects by comparing the observed print to expected outcomes and by recognizing telltale visual patterns of failure (such as spaghetti-like filament indicative of layer detachment, or irregular lines indicating extrusion problems). If a potential defect is found, the LLM agent queries the printer for relevant parameters (temperatures, feed rates, etc.) and then generates a plan of corrective actions in natural language which is parsed into machine instructions. For example, if it detects warping at the edges, the LLM might suggest increasing the bed temperature or adding brim support and then issue those commands to the printer. This multi-agent framework actually involves multiple LLM instances or prompt stages—one could be focused on vision analysis, another on decision-making—coordinating via an agent orchestration. The authors validated their system by comparing it against human operators: impressively, the LLM-based agent accurately identified common print errors (inconsistent extrusion, stringing, etc.) and corrected them on the fly, matching or exceeding the performance of a group of AM experts. This work exemplifies a cutting-edge use of LLMs beyond text generation: as reasoning engines in a physical feedback loop, requiring multi-modal understanding (vision and text) and real-time action planning. It also highlights some unique challenges—the LLM must be guided to focus on critical features in an image (likely through careful prompt engineering or few-shot visual examples), and safety is paramount (wrong corrective actions could damage equipment). LLM-3DPrint did not involve fine-tuning on image data; instead, it leveraged a powerful foundation model (GPT-4V) and domain-specific prompting. The success suggests that even without explicit training on manufacturing data, large models have enough general reasoning ability to be repurposed for industrial monitoring when given the right context.

Quality control can also benefit from LLMs in less direct ways. Fang et al.[53] explored using LLMs as few-shot defect detectors in AM processes. In their approach, the LLM is not physically connected to a printer but is given descriptions or data about a manufacturing process and asked to classify or identify defects, with only a few examples provided as guidance. The idea is to exploit the LLM's broad knowledge (gained from training on diverse technical texts) so that it can generalize to recognizing anomalies without needing thousands of labeled training data—a boon for manufacturing, where compiling large defect datasets is time-consuming. Their results, presented at the China Automation Congress 2024, indicated that an LLM could indeed achieve reasonable accuracy in identifying defect types after seeing only a handful of annotated cases. This approach treats inspection as a language problem: for example, feeding the LLM a textual summary of sensor readings or process

events and asking if it “looks like” any known failure mode, with prompt examples like “If the nozzle temperature drops and extrusion stops, it's likely a clog.” Such few-shot prompts effectively transfer knowledge of failure patterns into the LLM’s query, yielding a classification without a dedicated training run. While performance might not match a specialized vision algorithm in every scenario, the flexibility and low data requirement are key advantages. It underlines an important point: LLMs excel at low-data regime tasks in new domains by virtue of their pre-trained knowledge. In manufacturing quality control, this means they can recognize patterns of machine faults or product defects described in text, even if they have not been explicitly trained on those specific machines or products.

In summary, LLMs are being applied to enhance quality and reliability in AM both directly—by acting as real-time decision-makers in process control—and indirectly—by providing analysis and diagnostic suggestions to human engineers. These applications demand a high level of trustworthiness and accuracy, as mistakes can be costly. Thus, a recurring theme is the need for evaluation by domain experts and often keeping a human-in-the-loop. For instance, in LLM-3DPrint the interventions were verified to ensure they would not cause harm, and Fang et al. emphasized using LLM output as advice to be confirmed by engineers. The requirement for precision and safety is a defining characteristic of engineering LLM applications, distinguishing them from more forgiving domains.

4. POTENTIAL IMPACTS AND RISKS

Sam Altman, the CEO of OpenAI, has indicated that he believes advances in LLMs will help make scientific and technical progress faster than progress seen today [11]. The advent of AI programming tools such as Github Copilot allow for quicker start up to programming tasks especially those tasks that are routine but not routinely coded. These AI tools allow for human language queries to be translated into coding languages using proper syntax accelerating coding project start times and demonstrate how LLM assistants are becoming ubiquitous. In the materials domain, Boyar et al. [55] developed a LLM assistant for new material discovery that uses multimodal data that has a higher accuracy than traditional method. However, challenges remain, and these copilots and assistants may still generate output containing incomplete code or outdated information. Yet, they do provide a framework or useful information that an individual with knowledge and experience in the domain could leverage. These examples illustrate the simultaneous potential for accelerating progress and risk of thoughtless application that proliferate with the use of LLMs. These considerations must remain at the fore as LLMs are applied to various scientific domains.

As LLMs and LLM-based systems become increasingly complex, evaluating their performance and quality of their responses will be paramount. Traditional ML evaluation approaches, such as prediction accuracy and precision and recall, are ill-equipped to capture the relevant performance metrics of generative models—especially LLMs[1]. Currently, several metrics and benchmarks exist evaluating the responses of basic LLMs across additional characteristics such as natural language abilities, bias, reasoning, and trustworthiness[1]. Benchmarks and metrics include ROUGE and BERTScore; RealToxicityPrompts; HELLASWAG; and TruthfulQA respectively. However, evaluating augmented LLMs, such as LLMs with RAG, and LLM-based systems—like those leveraging LLMs as AI agents—is considerably less standardized and, in some cases, more subjective. For instance, evaluations for LLMs with RAG must also account for faithfulness of generated responses to the retrieved content. Finally, evaluation methods for LLM-based AI agents and agentic AI systems are nascent and often still require manual human testing and evaluation.

Furthermore, LLMs trained on a large corpus of text still suffer from hallucination. Banerjee et al.[56] suggest that LLMs will always hallucinate and suggests that we should just accept that this is a part of the cost of their use. The challenge is not knowing that LLMs hallucinate, but how to detect hallucinations as false information being presented in a convincing tone. Yao et al.[57] suggest that hallucinations are not bugs of coding but are adversarial information being presented in a convincing tone. The adversarial information or hallucinations could be from outdated or incomplete knowledge being presented as fact in the training dataset. These findings present challenges in science where the current solution is fine-tune a foundational model on a more selective dataset similar to LLM-Fusion [55] to reduce hallucination chance but this also presents the challenge that can limit new discoveries in the sciences.

Methods to limit hallucination would likely require a meticulously-curated and constantly-updated dataset of current scientific information that would present other challenges such as siloed information and questions about how to determine quality information to add to the LLM dataset. Maintaining such a store of data could easily lead to a lag in new discoveries because it would not allow the quickest ingestion of current scientific discoveries to be added to the dataset that the LLM uses. This could potentially lead to comparably inaccurate results based on outdated or incomplete information in the foundational LLMs. An understanding of how to curate scientific literature, data, analysis, and discoveries to provide the quickest and most up-to-date information for the LLM to draw from is still yet to be determined.

5. CONCLUSION

Unequivocally, LLMs represent a significant and transformative technological development. The explosion of interest about them and varied and dynamic research efforts surrounding them are indicative of the early stage of development and application that they are still in. While LLMs present significant advantages over other information retrieval methods, due to their ability to generate response from a diverse data corpus, there are challenges when using LLMs for domain specific areas. The biggest challenge is in improving the responses of the LLM with up-to-date and relevant data sources. Methods for improving the responses include using prompt engineering methods which can improve responses by providing specific context to the initial query, adding external datasets and implementing a retrieval method to augment the original query adding resources that improve trustworthiness of the response at the cost of computational time, and fine-tuning the LLM with new data which maintains the same computational query time at the cost of additional training time and possible loss of previous contextual functionality due to the fine-tuning on more specific data. Each method comes with benefits and costs and still face the common challenge of "hallucinations" by the LLM.

LLMs do provide a rich set of methods for synthesizing heterogenous data sources that allow domain experts to interact with information in previously unexplored ways. However, when applying LLMs to specific domains, a combination of the methods for improving responses will be required and will likely have to employ a human in the middle (HIM) feedback system. This HIM system would consist of experts in the domain that can curate the external dataset (vector store), develop queries and appropriate responses based on current information to be applied for supervised fine-tuning, and provide methods for improving the queries through prompt engineering. All of which are active areas of research in the LLM field. Unfortunately, advances in LLMs have not yet led to AGI and therefore still require more research as well as significant amounts of tailoring in order to apply to domain specific applications and to provide an appreciable augmentation to expertise in these domains.

REFERENCES

[1] Y. Chang *et al.*, "A Survey on Evaluation of Large Language Models," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, pp. 1–45, Jun. 2024, doi: 10.1145/3641289.

[2] Z. Feng *et al.*, "Trends in Integration of Knowledge and Large Language Models: A Survey and Taxonomy of Methods, Benchmarks, and Applications," Oct. 23, 2024, *arXiv*: arXiv:2311.05876. doi: 10.48550/arXiv.2311.05876.

[3] H. Naveed *et al.*, "A Comprehensive Overview of Large Language Models," Oct. 17, 2024, *arXiv*: arXiv:2307.06435. doi: 10.48550/arXiv.2307.06435.

[4] M. U. Hadi *et al.*, "Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects," Aug. 12, 2024, *Preprints*. doi: 10.36227/techrxiv.23589741.v6.

[5] J. Bi *et al.*, "Large AI Models and Their Applications: Classification, Limitations, and Potential Solutions," *Softw Pract Exp*, vol. 55, no. 6, pp. 1003–1017, Jun. 2025, doi: 10.1002/spe.3408.

[6] C. Zhang *et al.*, "A survey on potentials, pathways and challenges of large language models in new-generation intelligent manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 92, p. 102883, Apr. 2025, doi: 10.1016/j.rcim.2024.102883.

[7] Y. Li *et al.*, "Large Language Models for Manufacturing," Oct. 28, 2024, *arXiv*: arXiv:2410.21418. doi: 10.48550/arXiv.2410.21418.

[8] L. Makatura *et al.*, "How Can Large Language Models Help Humans in Design and Manufacturing?," Jul. 25, 2023, *arXiv*: arXiv:2307.14377. doi: 10.48550/arXiv.2307.14377.

[9] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," May 19, 2016, *arXiv*: arXiv:1409.0473. doi: 10.48550/arXiv.1409.0473.

[11] S. Altman, "Three Observations," Sam Altman. Accessed: Feb. 08, 2025. [Online]. Available: https://blog.samaltman.com/three-observations

[12] A. Lohn, "Scaling AI Cost and Performance of AI at the Leading Edge," Center for Security and Emerging Technology, Dec. 2023.

[13] J. Kahn, "The $19.6 billion pivot: How OpenAI's 2-year struggle to launch GPT-5 revealed that its core AI strategy has stopped working," Fortune. Accessed: Feb. 28, 2025. [Online]. Available: https://fortune.com/2025/02/25/what-happened-gpt-5-openai-orion-pivot-scaling-pre-training-llm-agi-reasoning/

[14] G. Marcus, "The most underreported and important story in AI right now is that pure scaling has failed to produce AGI," Fortune. [Online]. Available: https://fortune.com/2025/02/19/generative-ai-scaling-agi-deep-learning/

[15] P. Villalobos, A. Ho, J. Sevilla, T. Besiroglu, L. Heim, and M. Hobbhahn, "Will we run out of data? Limits of LLM scaling based on human-generated data," Jun. 04, 2024, *arXiv*: arXiv:2211.04325. doi: 10.48550/arXiv.2211.04325.

[16] F. Xue, Y. Fu, W. Zhou, Z. Zheng, and Y. You, "To Repeat or Not To Repeat: Insights from Scaling LLM under Token-Crisis".

[17] I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, and Y. Gal, "AI models collapse when trained on recursively generated data," *Nature*, vol. 631, no. 8022, pp. 755–759, Jul. 2024, doi: 10.1038/s41586-024-07566-y.

[18] K. Amin, S. Babakniya, A. Bie, W. Kong, U. Syed, and S. Vassilvitskii, "Escaping Collapse: The Strength of Weak Data for Large Language Model Training," Feb. 14, 2025, *arXiv*: arXiv:2502.08924. doi: 10.48550/arXiv.2502.08924.

[19] E. Bhardwaj, R. Alexander, and C. Becker, “Limits to AI Growth: The Ecological and Social Consequences of Scaling,” Jan. 31, 2025, *arXiv*: arXiv:2501.17980. doi: 10.48550/arXiv.2501.17980.
[20] Q. Dong *et al.*, “A Survey on In-context Learning,” Oct. 05, 2024, *arXiv*: arXiv:2301.00234. doi: 10.48550/arXiv.2301.00234.
[21] J. Wei *et al.*, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” Jan. 10, 2023, *arXiv*: arXiv:2201.11903. doi: 10.48550/arXiv.2201.11903.
[22] E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models,” Oct. 16, 2021, *arXiv*: arXiv:2106.09685. doi: 10.48550/arXiv.2106.09685.
[23] Z. Jiang *et al.*, “Active Retrieval Augmented Generation,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore: Association for Computational Linguistics, 2023, pp. 7969–7992. doi: 10.18653/v1/2023.emnlp-main.495.
[24] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, “When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories,” Jul. 02, 2023, *arXiv*: arXiv:2212.10511. doi: 10.48550/arXiv.2212.10511.
[25] L. Wang, N. Yang, and F. Wei, “Query2doc: Query Expansion with Large Language Models,” Oct. 11, 2023, *arXiv*: arXiv:2303.07678. doi: 10.48550/arXiv.2303.07678.
[26] V. Karpukhin *et al.*, “Dense Passage Retrieval for Open-Domain Question Answering,” Sep. 30, 2020, *arXiv*: arXiv:2004.04906. doi: 10.48550/arXiv.2004.04906.
[27] Z. Yin, Q. Sun, Q. Guo, J. Wu, X. Qiu, and X. Huang, “Do Large Language Models Know What They Don’t Know?,” May 30, 2023, *arXiv*: arXiv:2305.18153. doi: 10.48550/arXiv.2305.18153.
[28] S. Kadavath *et al.*, “Language Models (Mostly) Know What They Know,” Nov. 21, 2022, *arXiv*: arXiv:2207.05221. doi: 10.48550/arXiv.2207.05221.
[29] O. Yoran, T. Wolfson, O. Ram, and J. Berant, “Making Retrieval-Augmented Language Models Robust to Irrelevant Context,” May 05, 2024, *arXiv*: arXiv:2310.01558. doi: 10.48550/arXiv.2310.01558.
[30] N. D. Cao, G. Izacard, S. Riedel, and F. Petroni, “Autoregressive Entity Retrieval,” Mar. 24, 2021, *arXiv*: arXiv:2010.00904. doi: 10.48550/arXiv.2010.00904.
[31] Y. Tay *et al.*, “Transformer Memory as a Differentiable Search Index”.
[32] M. Bevilacqua, G. Ottaviano, P. Lewis, W. Yih, S. Riedel, and F. Petroni, “Autoregressive Search Engines: Generating Substrings as Document Identifiers”.
[33] W. Yu *et al.*, “Generate rather than Retrieve: Large Language Models are Strong Context Generators,” Jan. 25, 2023, *arXiv*: arXiv:2209.10063. doi: 10.48550/arXiv.2209.10063.
[34] Z. Sun, X. Wang, Y. Tay, Y. Yang, and D. Zhou, “Recitation-Augmented Language Models,” Feb. 16, 2023, *arXiv*: arXiv:2210.01296. doi: 10.48550/arXiv.2210.01296.
[35] S. Wang *et al.*, “Reinforcement Learning Enhanced LLMs: A Survey,” Feb. 24, 2025, *arXiv*: arXiv:2412.10400. doi: 10.48550/arXiv.2412.10400.
[36] S. Casper *et al.*, “Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback,” Sep. 11, 2023, *arXiv*: arXiv:2307.15217. doi: 10.48550/arXiv.2307.15217.
[37] K. Kumar *et al.*, “LLM Post-Training: A Deep Dive into Reasoning Large Language Models,” Mar. 24, 2025, *arXiv*: arXiv:2502.21321. doi: 10.48550/arXiv.2502.21321.
[38] D. Bandyopadhyay, S. Bhattacharjee, and A. Ekbal, “Thinking Machines: A Survey of LLM based Reasoning Strategies,” Mar. 13, 2025, *arXiv*: arXiv:2503.10814. doi: 10.48550/arXiv.2503.10814.
[39] P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar, “The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity,” Jun. 07, 2025, *arXiv*: arXiv:2506.06941. doi: 10.48550/arXiv.2506.06941.
[40] R. Sapkota, K. I. Roumeliotis, and M. Karkee, “AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges,” May 28, 2025, *arXiv*: arXiv:2505.10468. doi: 10.48550/arXiv.2505.10468.
[41] K. Pandya and D. M. Holia, “Automating Customer Service using LangChain,” 2023.
[42] S. Yuan *et al.*, “EASYTOOL: Enhancing LLM-based Agents with Concise Tool Instruction,” Mar. 27, 2024, *arXiv*: arXiv:2401.06201. doi: 10.48550/arXiv.2401.06201.

[43] Y. Li *et al.*, "Large Language Models for Manufacturing," Oct. 28, 2024, *arXiv*: arXiv:2410.21418. doi: 10.48550/arXiv.2410.21418.
[44] A. Chandrasekhar, J. Chan, F. Ogoke, O. Ajenifujah, and A. B. Farimani, "AMGPT: a Large Language Model for Contextual Querying in Additive Manufacturing," May 24, 2024, *arXiv*: arXiv:2406.00031. Accessed: Nov. 15, 2024. [Online]. Available: http://arxiv.org/abs/2406.00031
[45] B. Zirnstein, "Extended context for InstructGPT with LlamaIndex," 2023, doi: 10.13140/RG.2.2.17701.31204.
[46] X. Bai and X. Zhang, "Artificial Intelligence-Powered Materials Science," *Nano-Micro Lett.*, vol. 17, no. 1, p. 135, Dec. 2025, doi: 10.1007/s40820-024-01634-8.
[47] Y. An *et al.*, "Knowledge Graph Question Answering for Materials Science (KGQA4MAT)," in *Metadata and Semantic Research*, vol. 2048, E. Garoufallou and F. Sartori, Eds., in Communications in Computer and Information Science, vol. 2048. , Cham: Springer Nature Switzerland, 2024, pp. 18–29. doi: 10.1007/978-3-031-65990-4_2.
[48] S. Badini, S. Regondi, and R. Pugliese, "Unleashing the Power of Artificial Intelligence in Materials Design," *Materials*, vol. 16, no. 17, p. 5927, Aug. 2023, doi: 10.3390/ma16175927.
[49] Y. Ye *et al.*, "Construction and Application of Materials Knowledge Graph in Multidisciplinary Materials Science via Large Language Model".
[50] A. Chandrasekhar, J. Chan, F. Ogoke, O. Ajenifujah, and A. B. Farimani, "AMGPT: a Large Language Model for Contextual Querying in Additive Manufacturing," May 24, 2024, *arXiv*: arXiv:2406.00031. doi: 10.48550/arXiv.2406.00031.
[51] H. Xie, D. Hoskins, K. Rowe, and F. Ju, "Spatio-Temporal Transformer for Temperature Profiles Prediction in Large Format Additive Manufacturing," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, Bari, Italy: IEEE, Aug. 2024, pp. 1331–1336. doi: 10.1109/CASE59546.2024.10711728.
[52] Y. Jadhav, P. Pak, and A. B. Farimani, "LLM-3D Print: Large Language Models To Monitor and Control 3D Printing," May 07, 2025, *arXiv*: arXiv:2408.14307. doi: 10.48550/arXiv.2408.14307.
[53] Q. Fang, G. Xiong, F. Wang, Z. Shen, X. Dong, and F.-Y. Wang, "Large Language Models as Few-Shot Defect Detectors for Additive Manufacturing," in *2024 China Automation Congress (CAC)*, Qingdao, China: IEEE, Nov. 2024, pp. 6900–6905. doi: 10.1109/CAC63892.2024.10865554.
[54] A. Jignasu, K. Marshall, B. Ganapathysubramanian, A. Balu, C. Hegde, and A. Krishnamurthy, "Evaluating Large Language Models for G-Code Debugging, Manipulation, and Comprehension," in *2024 IEEE LLM Aided Design Workshop (LAD)*, San Jose, CA, USA: IEEE, Jun. 2024, pp. 1–5. doi: 10.1109/LAD62341.2024.10691700.
[55] O. Boyar, I. Priyadarsini, S. Takeda, and L. Hamada, "LLM-Fusion: A Novel Multimodal Fusion Model for Accelerated Material Discovery," Mar. 02, 2025, *arXiv*: arXiv:2503.01022. doi: 10.48550/arXiv.2503.01022.
[56] S. Banerjee, A. Agarwal, and S. Singla, "LLMs Will Always Hallucinate, and We Need to Live With This," Sep. 09, 2024, *arXiv*: arXiv:2409.05746. doi: 10.48550/arXiv.2409.05746.
[57] J.-Y. Yao, K.-P. Ning, Z.-H. Liu, M.-N. Ning, Y.-Y. Liu, and L. Yuan, "LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples," Aug. 04, 2024, *arXiv*: arXiv:2310.01469. doi: 10.48550/arXiv.2310.01469.