Information Fusion 117 (2025) 102840

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Federated and edge learning for large language models

Francesco Piccialli [a], Diletta Chiaro [a,*], Pian Qi [a], Valerio Bellandi [b], Ernesto Damiani [b]

[a] Department of Mathematics and Applications "R. Caccioppoli", Mathematical mOdelling and Data AnaLysis (M.O.D.A.L.) research group, University of Naples Federico II, Naples, Italy

[b] Department of Computer Science, Università degli Studi di Milano, Milano, Italy

ARTICLE INFO

Keywords:
Edge learning
Edge computing
Federated learning
Large language models
Natural language processing

ABSTRACT

As the demand for sophisticated language models (LMs) continues to grow, the necessity to deploy them efficiently across federated and edge environments becomes increasingly evident. This survey explores the nuanced interplay between federated and edge learning for large language models (LLMs), considering the evolving landscape of distributed computing. We investigate how federated learning paradigms can be tailored to accommodate the unique characteristics of LMs, ensuring collaborative model training while respecting privacy constraints inherent in federated environments. Additionally, we scrutinize the challenges posed by resource constraints at the edge, reporting on relevant literature and established techniques within the realm of LLMs for edge deployments, such as model pruning or model quantization. The future holds the potential for LMs to leverage the collective intelligence of distributed networks while respecting the autonomy and privacy of individual edge devices. Through this survey, the objective is to provide an in-depth analysis of the current state of efficient and privacy-aware LLM training and deployment in federated and edge environments, with the aim of offering valuable insights and guidance to researchers shaping the ongoing discussion in this field.

1. Introduction

> "Language is a process of free creation; its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation".
>
> *Noam Chomsky*

Expressing and communicating through language is a fundamental human ability that begins to develop in early childhood and continues to evolve throughout a lifetime [1]. Unlike humans, machines lack the innate capacity to comprehend and communicate in human language unless empowered with sophisticated artificial intelligence (AI) algorithms. Since the proposal of the Turing Test in the 1950s, overcoming this challenge has been a longstanding pursuit in research, aiming to equip machines with the capability to read, write, and communicate like humans [2]. Language modeling, a crucial task in natural language processing (NLP), stands out as a significant approach in enhancing the language intelligence of machines by enabling them to predict the next word or character in a given text sequence [3,4], thus allowing the model to produce new text and complete sentences, among its diverse applications.

Language models (LMs) can be mainly categorized into four categories, as depicted in Fig. 1: statistical language models, machine learning (ML) models, deep learning (DL) models, and transformer-based models. Early language models relied on basic statistical methods that estimated word sequence probabilities through frequency counts [5]. Examples of probability-based LMs include n-grams [6], Hidden Markov Models (HMMs) [7], and Maximum Entropy Models [8]. N-grams, as an example, are sequences of neighboring words or tokens utilized to predict the likelihood of the subsequent word based on preceding ones [9]. Although regarded as basic by modern criteria, these models represented a crucial initiation in comprehending natural language, enabling fundamental text generation and word prediction but having constraints in grasping intricate contextual associations [10–12]. Then a shift toward data-driven methodologies occurred [13], and researchers explored ML algorithms to enhance language understanding [14]. Models such as Support Vector Machines (SVMs) exemplify this shift [15]. ML models brought a more sophisticated approach to NLP tasks, enabling the development of applications like spam detection [16] and sentiment analysis [17]. The availability of large-scale Twitter datasets, in particular, revolutionized real-time sentiment analysis [18]. The rise of DL, along with the availability of extensive public datasets [19], and powerful computing devices [20] capable of processing large amounts of complex data, represented a crucial juncture in the advancement of LMs [21]. Notably, neural networks, specifically Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, capable to capture intricate

* Corresponding author.
E-mail address: diletta.chiaro@unina.it (D. Chiaro).

https://doi.org/10.1016/j.inffus.2024.102840
Received 5 November 2024; Accepted 28 November 2024
Available online 16 December 2024

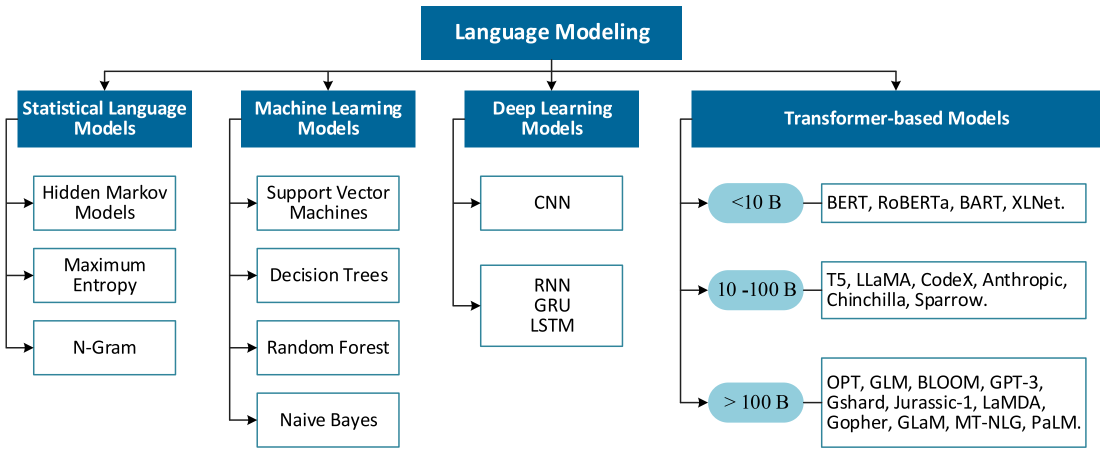

Fig. 1. A hierarchical representation of LMs, illustrating the commonly adopted classification, reporting some solutions ranging from statistical-based models to traditional machine and deep learning-based ones, and finally to the latest transformer-based models.

features and long-range dependencies within textual content, gained prominence in this era [22]. This phase significantly enhanced the models' capacity to comprehend context, rendering them well-suited for tasks such as machine translation and speech recognition [23,24]. Nonetheless, DL encountered challenges related to vanishing gradients [25] and long-term dependencies [26], thereby constraining its overall effectiveness.

However, it was not until the introduction of the Transformer architecture in the influential work "Attention is All You Need" in 2017 [27] that a truly groundbreaking leap occurred in the realm of LLMs. Founded on the self-attention mechanism [28] and frequently pre-trained on extensive text corpora, Transformers empowered models to encompass the entire context of a sentence or document, fostering genuine contextual understanding [29], and leading to a revolution in applications like chatbots [30], text summarization [31], and language translation [32]. To distinguish LMs based on parameter scales, the research community has coined the term "large language models" (LLMs) for pre-trained language models (PLMs) with substantial sizes, often containing tens or hundreds of billions of parameters. An essential characteristic of LLMs is their capacity to handle vast amounts of data, including unstructured text, and capture semantic relationships between words and phrases [33]. Furthermore, these models can process various types of data, such as visual [34], audio [35], audiovisual [36], and multi-modal data [37], learning the semantic connections among them.

This breakthrough paved the way for the development of state-of-the-art models, exemplified by OpenAI's GPT (Generative Pre-trained Transformer) series [38], PaLM (Pathways Language Model) [39], and Google's Gemini [40] and Gemma [41]. Additionally, LLM developments have expanded into more specialized fields [42,43], with models created for tasks including code production [44], scientific research [45], website building [46], and medical language processing [47]. The process of crafting and tuning prompts in NL to optimize the performance of LLMs for specific tasks is termed prompt engineering. It involves strategically constructing input prompts to guide AI models towards producing more accurate, pertinent, and valuable responses. Effective prompt engineering can substantially enhance LLMs' effectiveness in particular tasks by furnishing clear instructions and contextual cues to steer the model's output. Moreover, prompt engineering aids in mitigating "catastrophic forgetting", wherein an LLM may lose previously acquired knowledge when fine-tuning for a new task, as well as the occurrence of hallucinations, wherein AI models, particularly LLMs, produce irrelevant, implausible, or nonsensical outputs. While hallucinations and prompting might indeed be fascinating topics, this survey does not delve into the specifics of these aspects. In order to guarantee responsible and equitable use, efforts have also been made to address ethical concerns [48], interpretability [49], and minimizing biases in LLMs [50].

However, LLMs' enormous resource requirements pose obstacles. A paradigm shift has been spurred by the cost of training on cloud servers with strong Graphical Processing Units (GPU) clusters and the latency of cloud-based inference. There are many different reasons to move LLM inference to the edge [51], and these reasons are influenced by variables unique to LLM, original equipment manufacturer considerations, and industry dynamics. The primary factor pushing LLMs to the edge is the decrease in reliance on connectivity [51]. Edge-deployed LLMs, in contrast to their cloud-based counterparts, can operate without any or very little network access. As an essential component of an ideal user experience in LLM-based apps, latency drives edge migration as well. Reaction times can be significantly reduced by locally conducted inference, which offers far better user experience than relying on the reliability and speed of a network connection. Also, by reducing the need to send sensitive data via networks, this method reduces the possibility of data breaches and gives consumers more control over their personal information. Customization appears as a driving force behind edge deployments, impacting both inference and training [52]. An LLM can, on the verge, deeply understand a user's speech patterns, writing style, and more. Enhancing privacy is coupled with the capability for devices to customize models to align with specific personalities and behaviors, thereby crafting a uniquely tailored user experience. Scalability represents another key driver, as the widespread distribution of applications across a diverse array of devices is facilitated, avoiding the burden of overwhelming centralized servers, thanks to the growing prevalence of edge devices. With the in-depth exploration of parallel, distributed, and federated learning (FL) in recent years, numerous solutions in the realms of edge learning (EL) and federated learning have been suggested. These solutions aim to train, fine-tune, or facilitate the deployment of LLMs [53,54].

In light of these transformative trends, our survey becomes instrumental in comprehensively examining the current landscape and future trajectories of federated and edge learning within the domain of LLMs. The survey aims to delve into the nuances of how FL and EL are shaping the evolution of LLMs, exploring their impact on scalability, privacy, and user experience. By synthesizing insights from industry developments, research advancements, and emerging trends, this survey endeavors to provide a roadmap for the ongoing integration of federated and EL in the realm of LLMs. The remainder of this survey is structured as follows: Section 2 provides an introduction to LLMs, encompassing a concise history of state-of-the-art solutions and the prevalent approach to dealing with them, namely pre-training and fine-tuning. Additionally, it delves into the reasons and challenges

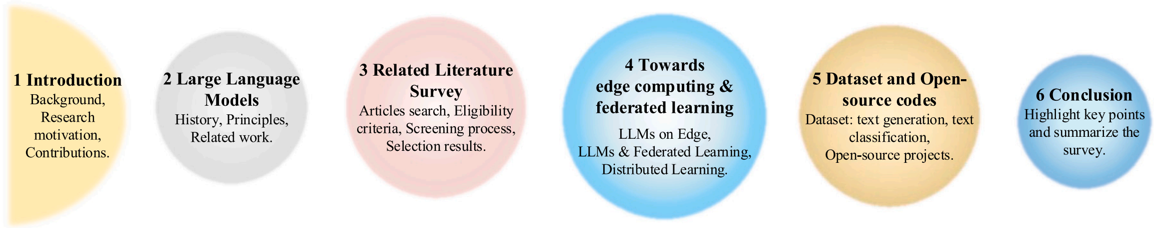

Fig. 2. The structure of the survey.

associated with LLMs in both EL and FL. Section 3 outlines the methodology employed in this survey, detailing the retrieval strategy for the discussed and reported papers. In Section 4, a deeper examination is undertaken, focusing on scrutinized papers related to LLMs and EL and/or FL. This section describes the solutions adopted to facilitate model deployment on the edge, as well as techniques within the federated realm. This section aims to shed light on the innovative approaches and practical strategies employed to address the unique constraints and opportunities presented by deploying LLMs in edge and federated environments. In Section 5, we ventured into the realm of available datasets and open-source codes closely aligned with the research domain and pertinent to the surveyed papers. Through the presentation of numerous tables summarizing content and providing direct links to repositories, we aimed to furnish readers with a comprehensive resource base for further exploration and utilization in their own endeavors. Conclusions close the survey by summarizing the key findings and insights gleaned from our investigation. Fig. 2 illustrates the structure of this survey.

2. Large language models

2.1. LLMs: then and now

The evolution of LLMs can be traced back to recent years, witnessing notable progress and breakthroughs with the introduction of the Transformer architecture and the launch of the GPT series. In 2017, Google proposed the Transformer model [27], leveraging the attention mechanism to learn longer-term dependencies in language and enabling parallel training on multiple GPUs [55]. This innovation facilitated the training of significantly larger models [56].

Following this development, in 2018, OpenAI adopted the novel neural network architecture for language modeling tasks, unveiling the inaugural GPT model, GPT-1 [57]. GPT-1 showcased substantial enhancements in commonsense reasoning, question answering, and text entailment compared to existing pre-trained language models. Although its limitations, it laid the foundation for subsequent, more potent models, ushering in a new era of AI research and highly competitive exploration in LLMs.

In 2019, OpenAI released GPT-2 [58], boasting a parameter size ten times larger than GPT-1, totaling 1.5 billion parameters. By 2020, GPT-3 [59] was launched, standing out as one of the largest language models to date with an impressive 175 billion parameters. The GPT-3 family, particularly ChatGPT [60], gained widespread attention and popularity across various industries since its November 2022 release. In March 2023, GPT-4 [61] was unveiled, extending text input to fused multimodal inputs. GPT-4 demonstrated enhanced capabilities in handling complex tasks, exhibiting significant performance improvements and the ability to generate even more coherent and natural-sounding text compared to its predecessor. Simultaneously, other outstanding LLMs emerged during this period. Google's BERT [62], released in 2018 with 1.1 billion parameters, achieved SOTA results across 11 NLP tasks. In 2019, Facebook AI developed BART [63] and RoBERTa [64], which are improved versions of the BERT model. In the same year, Google developed XLNet [65] and T5 [66]. XLNet is a generalized autoregressive pre-training model that performs well on multiple NLP tasks. T5, or Text-to-Text Transfer Transformer, achieves impressive results on various benchmarks.

In June 2020, GShard [67] is tailored for distributed training of massive models, enabling efficient processing across multiple accelerators (such as GPUs or TPUs). In the year 2021, EleutherAI developed both GPT-Neo [68] and GPT-J [69]. GPT-Neo, a community-driven project, aims to create accessible and powerful language models, available in various sizes such as GPT-Neo 1.3B and GPT-NeoX with billions of parameters [70]. Notable LLMs from the same year include CodeX [71], Jurassic-1 [72], AnthropicLM [73], GLaM [74], and Gopher [75].

The year 2022 witnessed an explosion in the development of LLMs, with models like MT-NLG [76], InstructGPT [77], LaMDA [78], Chinchilla [79], PaLM [39], OPT [80], BLOOM/BLOOMZ [81,82], Minerva [83], Sparrow [84], Flan-PaLM [85], Galactica [86], AnthropicLM v4-s3 [73], OPT-IML [87], etc. These models ranged from 10B to 100B parameters, with pre-training data sizes reaching up to 1.4 trillion tokens.

In 2023, LLaMA was introduced by Touvron et al. [88], featuring a parameter range from 7 billion to 65 billion. LLaMA demonstrated outstanding performance in instruction-following tasks, with LLaMA-13B outperforming GPT-3 in various benchmarks. The same year also saw the launch of chatbots like Bard [89], Claude [73], and Gemini [40], along with further improved LLMs such as Vicuna [90], Jurassic-2 [91], Falcon 40B/180B [92], Gorilla [93], Orca/Orca-2 [94,95], among others. In early 2024, the Google DeepMind team unveiled Gemma [41], the newest iteration of a lightweight open model.

In 2024, research related to LLMs remained popular, with many teams focusing on developing multimodal models to create more robust foundational systems. For instance, Stable LM 2 [96] was introduced for multilingual tasks and trained on data from seven languages. The Google DeepMind team also launched a new version of their lightweight open model, Gemma [41]. In March, Inflection-2.5 [97] was released, which enhanced the functionality of personal AI assistants while optimizing resource use during training. That same month, Claude 3 debuted, offering significant improvements over its predecessor, particularly in various cognitive tasks. Following this, LLaMA 3 was released in April, and GPT-4o [98] arrived in May as a multimodal AI capable of processing and generating text, audio, and visual content in real time. In July, Qwen2 [99] was released, building on the original Qwen model and incorporating several enhancements, including improved performance in chat applications and stronger multilingual capabilities. In August, the xAI team introduced Grok 2 [100], which is tailored specifically for users of the X platform. Most recently, in October, Gemini 1.5 Flash-8B [101] went into production, boasting enhanced speed and efficiency compared to earlier versions. Fig. 3 provides a timeline overview of both open and proprietary LLMs.

2.2. LLMs: pre-training then fine-tuning

LLMs share a common procedure in their training process, which involves pre-training on large text data corpora followed by task-specific fine-tuning [4]. The models are exposed to a variety of online texts during pre-training to acquire facts, grammar, reasoning skills, and a

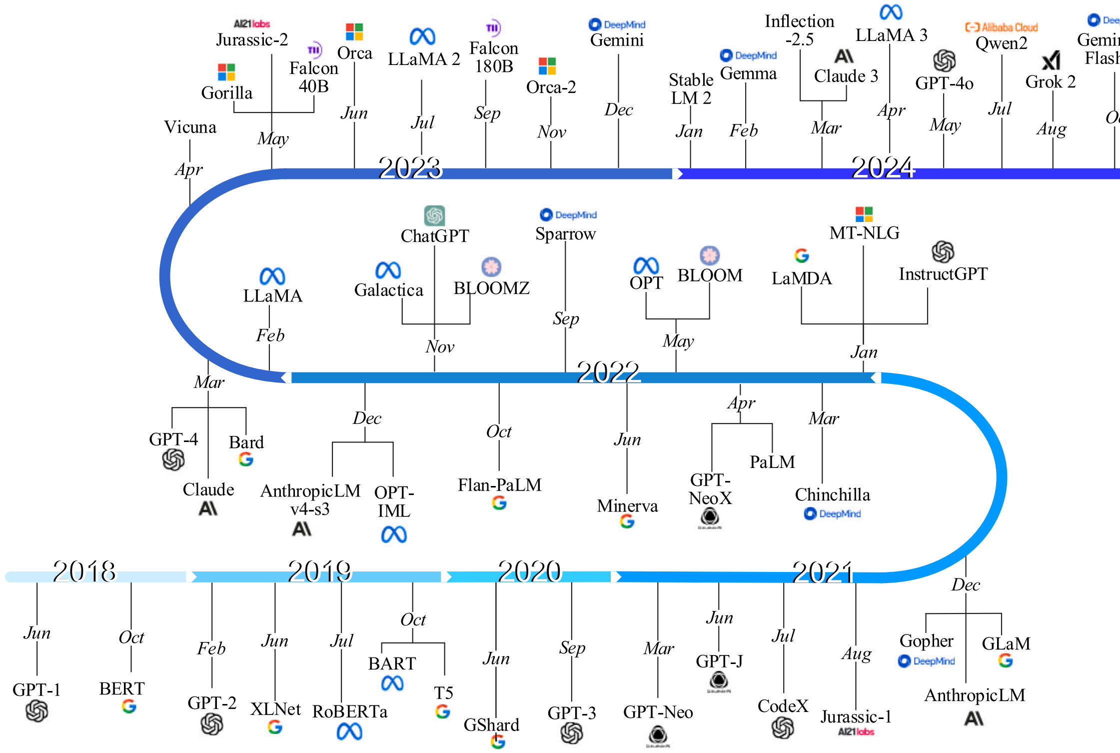

Fig. 3. Timeline of LLMs in open source and proprietary domains.

certain amount of common sense knowledge [102,103]. They acquire a wide comprehension of language as a result of this process. The models are then fine-tuned on smaller datasets to customize them for specific uses. For example, ChatGPT is optimized for conversational settings, making it suitable for virtual assistants and chatbots [90,104]. Though less well-known, Llama and Falcon are potential developments or specialized versions, perhaps created for particular use cases or research goals. Collectively, these models showcase cutting-edge advancements in NLP, enabling enhanced comprehension and human-like interactions attributed to the prowess of AI-driven language models [105–107]. The training process for models like ChatGPT, Llama, and Falcon [108,109] encompasses several crucial phases. The initial stage is pre-training, wherein these models undergo training on an extensive and diverse dataset of online text. This phase aims to instill grammar, vocabulary, context, and general knowledge [110] into the models' learning framework. The foundational architecture of the Transformer model plays a pivotal role in understanding the relationships between words within sentences. Following the pre-training phase, models undergo refinement using task-specific datasets tailored for particular objectives, such as text creation or discussion in the case of ChatGPT. Their proficiency in these specific tasks is honed through fine-tuning, employing hyperparameter optimization to maximize performance. Ethical considerations are integral to this process, aiming to mitigate unfavorable or biased outcomes. Furthermore, the training is a resource-intensive and iterative endeavor, subject to continuous monitoring and adjustments to enhance both performance and safety [111,112].

LLMs have progressed through various developmental phases, witnessing an evolution in both size and complexity. The GPT series, comprising GPT-1, GPT-2, and GPT-3 [113], has exhibited successive growth in the number of parameters. Beginning with a scale in the hundreds of millions for GPT-1, it has now reached a staggering 1.7 trillion parameters for GPT-4 [114]. This substantial increase in parameters facilitates enhanced language understanding and generation capabilities [115]. In a parallel vein, models inspired by BERT have also undergone advancements in pre-training strategies. Notable examples include ALBERT (A Lite BERT) [116] and RoBERTa [64], which have contributed to significant improvements in both performance and efficiency. Furthermore, in terms of training mode, there has been a notable trend towards embracing multi-modality. Take Gemini [40], for instance, which is designed to process various data types simultaneously, including text, images, audio, video, and even code, rather than exclusively relying on textual corpora. This transition has garnered substantial interest within the community. As illustrated in Fig. 4, the escalation in the size of LLMs correlates with a corresponding rise in hardware requirements. Absolutely, **GPU** and **RAM** are crucial hardware components for running LLMs. During both training and inference, LLMs typically rely on GPUs or tensor processing units (TPUs) [117,118]. These processors are particularly well-suited for handling the computational demands of transformer-based models, which are commonly used in LLMs. To function, they need a sizeable quantity of memory [118], indispensable for efficiently managing large datasets and model parameters during training. LLM training setups often require significant amounts of RAM, with DDR4 or DDR5 RAM, known for their high bandwidth and capacity, being recommended to prevent memory-related bottlenecks. For this reason, key considerations when selecting GPUs include factors such as memory capacity (VRAM), memory bandwidth, and CUDA cores (processing power), with high-end options like NVIDIA's Tesla series or GeForce RTX series being preferred for LLM training. Fast and high-capacity **storage** is crucial for managing the extensive data involved in LLM training, with Solid State Drives (SSDs), particularly NVMe SSDs, being favored over Hard Disk Drives (HDDs) due to their superior read and write speeds. Proper **cooling solutions**, such as high-performance fans or liquid cooling systems, are necessary [119] to prevent overheating resulting from the intense computational load of LLM training. A robust **power supply unit** (PSU) ensures consistent and sufficient power flow to all components. Sometimes, for training very large LLMs, **distributed computing setups** involving multiple GPUs or machines collaborating on training become essential, requiring networking infrastructure, specialized software frameworks (e.g., Horovod), and synchronization techniques to ensure efficient parallel processing. Central Processing Units (**CPUs**) remain crucial for data preprocessing, model setup, and

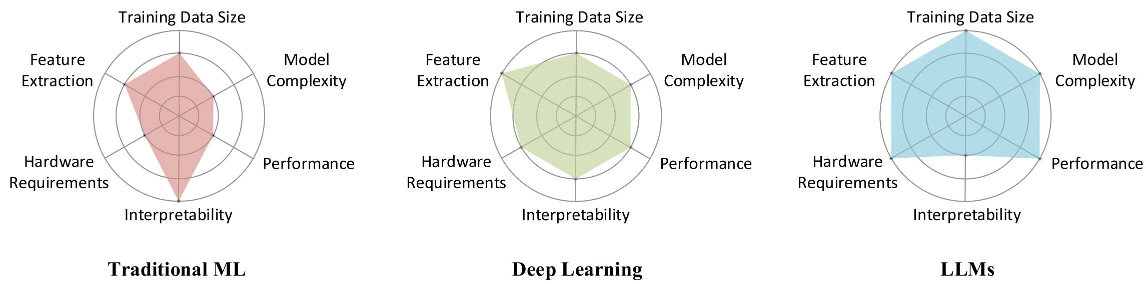

Fig. 4. Comparison of traditional machine learning, deep learning, and LLMs for language modeling. LLMs emerge as more demanding across various dimensions, necessitating huge data for training, extensive feature extraction, and exhibiting greater complexity, hardware demands, and reduced interpretability compared to traditional approaches.

coordination, playing a significant role in tasks such as data loading and preprocessing. While a powerful multi-core CPU can accelerate these tasks, the actual training phase heavily relies on the parallel processing capabilities of GPUs.

2.3. LLMs deployment on the edge

In recent years, LLMs have made significant strides in AI, showcasing advanced capabilities in NLP. However, the resource requirements for training these models on cloud servers, particularly those equipped with extensive GPU clusters, incur substantial costs. Additionally, the inference of these models on cloud servers presents challenges such as notable latency, impacting the overall user experience and raising concerns about privacy and security [120,121].

To address these issues, there is a growing trend to prioritize edge inference deployments for LLMs in upcoming platforms. This shift towards edge computing aims to mitigate the drawbacks associated with centralized cloud-based approaches, focusing on optimizing resource utilization, minimizing latency, and addressing privacy and security considerations in the practical implementation of LLMs [122]. Nevertheless, the inherent constraints of edge devices, including restricted processing power, memory, and storage, pose considerable obstacles to the seamless integration of resource-intensive LLMs [123]. In this context, addressing the limitations of edge deployment becomes crucial for unlocking the full potential of LLMs in diverse applications [124].

Following, we delve into main drivers for deploying LLMs on the edge, while also addressing the complex challenges linked to such deployments, such as resource constraints, energy efficiency considerations, security implications, and compatibility issues.

2.3.1. The whys

- **Reduced Connectivity Dependency:** Cloud-based LLMs have typically relied on a steady network connection for smooth inference. However, shifting LLM inference to the edge allows applications to function seamlessly in environments with unreliable or no network connectivity. This not only addresses operational hurdles but also enables the deployment of applications in resource-limited settings.
- **Low Latency for Enhanced User Experience:** Many LLM-based applications rely on swift responses to ensure top-notch user experiences. The speed and reliability of the network connection are crucial factors influencing the responsiveness of cloud-based LLMs. By moving inference tasks to the edge, response times are substantially reduced, enhancing user experience, especially in applications requiring real-time interactions.
- **Privacy and Data Security Through Edge Computing:** Edge computing emerges as a pivotal enabler for augmenting privacy and data security in LLM applications. By processing data locally on the device, attack surfaces are substantially reduced compared to traditional cloud-based systems. This mitigates the risk of data breaches, as sensitive information is not transmitted over the network to remote servers. The incorporation of FL further bolsters these privacy-centric measures, ushering in a new era of secure and decentralized data processing.
- **Personalization:** Edge computing facilitates a higher degree of personalization in LLM applications. Devices gain the capability to finely tune models according to individual user personalities and habits. This tailored approach ensures a more tailored and engaging user experience, as applications adapt dynamically to user preferences without reliance on centralized servers.
- **Scalability in Edge Deployment:** The scalability of edge devices plays a pivotal role in the widespread distribution of LLM applications. With edge devices deployed at scale, the distribution of applications across a diverse array of devices becomes feasible. This not only prevents overloading of central servers but also optimizes resource utilization, ensuring seamless scalability in response to growing user demands.

2.3.2. The challenges

- **Resource Constraints:** Efficiently running LLMs on edge devices presents a significant technical challenge due to inherent limitations in processing power, memory, and storage compared to robust cloud servers. Shrinking the size of LLMs without sacrificing performance is complex and requires sophisticated optimization and quantization techniques. Despite significant efforts in the AI industry, reducing LLM size is not just a preference but a necessity for successful deployment on the edge. This need is underscored by the incorporation of Neural Processing Units (NPUs), tailored for specific use cases, which play a vital role in the intricate landscape of edge computing.
- **Energy Efficiency:** [119] Using resource-intensive models like LLMs on battery-powered edge devices raises a crucial concern: rapid battery drainage. Developers and chip architects must meticulously optimize their designs to ensure energy efficiency [125]. The primary aim is to minimize any noticeable negative impacts on battery life, acknowledging the delicate equilibrium between computational demands and sustainable device operation. Achieving this balance requires a collaborative effort to improve algorithms, hardware architectures, and power management strategies [126].
- **Security:** The transition to edge computing offers the promise of improved data privacy compared to cloud-based models but also brings forth a unique set of challenges regarding data security on edge devices. The decentralized nature of edge computing necessitates strong measures to protect sensitive information processed locally. Therefore, implementing secure data storage protocols and encryption mechanisms becomes essential to counter potential threats and vulnerabilities in this distributed computing paradigm [127].
- **Compatibility:** The compatibility landscape poses a significant hurdle in the deployment of LLMs on edge devices. It is not guaranteed that LLMs will seamlessly integrate with all edge devices due to variations in hardware and software configurations. Developers play a pivotal role in ensuring compatibility by either crafting models capable of running on diverse configurations or by collaborating with hardware and software providers to

offer tailored solutions. The need for standardized approaches or customized adaptations becomes apparent to facilitate the widespread and effective deployment of LLMs across diverse edge computing environments.

2.4. LLMs within FL context

As technology continues to advance, FL is becoming increasingly important in enhancing the effectiveness, adaptability, and security of LLMs across various applications and industries. The collaborative nature of FL, along with its streamlined training methods and creative problem-solving capabilities, sets the stage for a transformative shift. The synergy between FL and LLMs not only helps them reach their full potential but also lays the foundation for a future where the seamless integration of these technologies plays a key role in advancing language processing and understanding.

2.4.1. The whys

Among the many reasons why LLMs may benefit from the federated approach, one notable advantage lies in the ability to create more customized models. For organizations seeking to fine-tune foundational models, accessing the necessary data is often a challenge due to its distribution across various departments, companies, and geographic regions. The scattered nature of this data, coupled with regulatory constraints on centralized data pooling, poses obstacles to traditional model refinement. However, FL presents a viable solution to this challenge. When used in conjunction with privacy technologies, FL allows organizations to access distributed data through a FL platform. This approach empowers organizations to drive better, more personalized models without the need to centralize or move the data. Importantly, it guarantees privacy to each data owner, fostering collaboration without compromising data security. Additionally, FL offers the added benefit of reducing the time and costs associated with centralizing data and establishing complex data sharing agreements. In particular, the key advantages of FL may be summarized as follows:

- **Advanced security:** FL prioritizes user privacy by sending only model updates, not raw data, to a central server. This decentralized approach aligns with privacy regulations, mitigates the risk of data breaches, and ensures data security. As data privacy gains significance, FL provides a solution that enables organizations to comply with data protection laws while harnessing the capabilities of LLMs [128].
- **Scalability and convenience:** FL's decentralized training spreads the computational workload, enhancing scalability and yielding substantial cost savings. By leveraging the computational power of diverse devices, FL makes fine-tuning a manageable and economically efficient process. This democratizes access to LLM benefits, particularly beneficial for organizations with limited resources.
- **Adaptability:** FL seamlessly addresses the challenge of continuously expanding datasets by integrating newly collected data into existing models. This ensures continuous improvement and adaptability to changing environments, making FL essential for the evolution of LLMs. In dynamic sectors like healthcare and finance, FL ensures LLMs stay relevant and practical, keeping pace with the latest information.
- **Optimized user experience:** FL tackles privacy and scalability concerns, enhancing the user experience by deploying models directly to edge devices. This speeds up model responses, minimizes latency, and ensures quick answers for users. Local deployment is particularly relevant in applications where immediate responses are critical, such as virtual assistants and interactive customer service, offering a practical solution to address user needs.

2.4.2. The challenges

While FL holds immense potential, its development for LLMs is still in a preliminary stage, primarily due to the following challenges:

- **High demands:** LLMs impose significant demands on memory, communication, and computational resources [129]. Traditional FL methods involve transmitting and training the entire LLM across multiple clients using their local data. However, the substantial size of LLMs introduces complexities related to storage, model transmission, and the computational resources needed for training or fine-tuning [130]. This challenge becomes particularly pronounced in scenarios with limited storage and computational capabilities, especially in cross-device FL.
- **Proprietary LLMs:** Proprietary LLMs pose challenges as clients do not own them. Allowing federated fine-tuning without accessing the entire model becomes necessary, particularly in closed-source LLMs. The ongoing debate about open-sourcing generative AI models has gained traction, especially following an incident where researchers instructed a proprietary generative AI system called MegaSyn [131] to create toxic molecules, some resembling known nerve agents. This raises a critical issue: opponents argue that open-sourcing generative AI may lead to misuse, while proponents believe that proprietary models concentrate too much power in the hands of a select few.

Despite these challenges, FL has the potential to overcome obstacles associated with using LLMs. Collaborative pre-training and fine-tuning enhance the robustness of LLMs, and efficient algorithms address memory, communication, and computation challenges [132]. Designing FL systems tailored to LLMs and harnessing decentralized data present exciting opportunities for the future.

3. Related literature survey

A thorough examination of existing literature focusing on articles introducing LLMs within edge and FL areas was conducted. Following, we elaborate on the methodology employed for retrieving relevant literature and delineate the process for selecting articles.

3.1. Retrieval strategy

To systematically clarify the methodology used in this article's literature review, we strictly adhered to PRISMA standards [133]. PRISMA has emerged as the gold standard for systematic reviews and meta-analyses, providing a robust framework that ensures transparency, reliability, and reproducibility in our research pursuits. Initially, we meticulously determined search terms, search time horizon, and search scope, aligning them with the thematic focus of the literature review. Subsequent phases involved the scrutiny of titles and abstracts to identify articles meeting eligibility criteria, followed by comprehensive reviews of full texts for further assessment. Ultimately, the inclusion of articles was contingent on their alignment with the review's topic and their provision of valuable solutions or insights to address the research questions at hand.

3.1.1. Articles search

We searched articles based on the occurrence of terms in titles, abstracts, and keywords. Initially, our focus was on articles specifically addressing LLMs. Building on this, we broadened our search criteria to include terms associated with edge computing, such as edge learning, edge computing, and mobile edge computing. Simultaneously, we incorporated terms relevant to federated/distributed learning (DistL). The formulated search keywords comprised a combination of these terms (including their plural forms) and were structured as follows: `TITLE-ABS-KEY: (("Large language models"`

OR "Large language model") AND ("edge" OR "edge learning" OR "edge-learning" OR "edge computing" OR "edge-computing" OR "mobile edge computing" OR "mobile devices" OR "IoT device" OR "on-device" OR "distributed learning" OR "distributed machine learning" OR "federated learning")).

Our search strategy involved a targeted exploration of published papers in *Scopus*, a comprehensive database, obviating the need for redundant searches in databases such as *ACM Library* and *IEEE Xplore*. Recognizing the novelty of our survey topic, which may result in a significant portion of pertinent research being available solely in preprint form, we systematically gathered all relevant preprint papers from *arXiv*, a comprehensive repository with a primary focus on computer science preprints. This dual approach, encompassing both established databases and preprint sources, was employed to guarantee an exhaustive examination of the existing literature on our subject matter.

3.1.2. Eligibility criteria

Following the survey topic and PRISMA guidelines, we established specific criteria for the literature review.

We excluded papers that: (i) were not in English; (ii) constituted duplicates; and (iii) belonged to the categories of "Review" or "Conference Review" articles.

Included papers met the following criteria: (i) contained at least one search term in the title, abstract, or keywords; (ii) demonstrated relevance to the deployment or training of LLMs at the edge through a careful examination of the abstract and full text; (iii) exhibited relevance to the deployment of LLMs in FL or other distributed environments through a thorough assessment of the abstract and full text.

3.1.3. Screening process

The article selection process adhered to a systematic methodology aimed at ensuring the inclusion of high-quality articles. Initially, searches were conducted in the *Scopus* and *arXiv* databases using formulated keywords. Subsequently, the search was refined by restricting the language to "English" and excluding document types such as "Conference review", "Review", "Editorial", and "Letter" that did not meet the specified criteria. A secondary screening was then implemented using Python for efficient and rapid execution. This phase involved utilizing code to eliminate articles with duplicate titles and to filter out articles containing "review" and "survey" in their titles. In the final screening stage, two reviewers among the authors meticulously and independently evaluated abstracts and full texts by the aforementioned eligibility criteria to identify articles that met the inclusion criteria. This progressive approach was designed to ensure a rigorous and thorough selection process. Fig. 5 illustrates the paper count at each stage of the selection process. Initially, a total of 282 papers were identified through the search query, and 4 papers were identified through a manual search. Following the elimination of duplicate and review articles using Python, 265 papers remained in consideration. During the screening step, where two independent reviewers assessed titles, abstracts, and full texts, a total of 114 papers were selected.

3.2. Selection results

The selection results reveal a distribution of papers by year, with a noticeably higher concentration in 2023 and 2024. This observation underscores the novelty of the research direction pursued in this article, affirming the increasing trend in studies exploring the integration of LLMs into edge computing or FL settings in recent years. We have further categorized it into three groups: edge learning, federated learning (FL), and distributed learning (DistL), based on the framework outlined in the papers. Furthermore, we meticulously documented the hardware configurations specified in each of the included articles, listed in Table 1. This aspect holds particular significance when training LLMs. As discussed above, the choice of hardware directly influences the training speed and overall performance of the model. Consequently, recording the hardware environments utilized affords us more interpretation of the experimental outcomes.

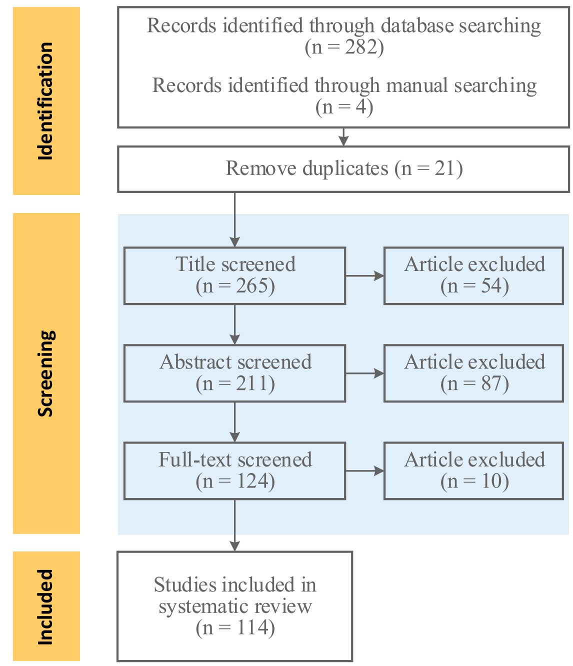

Fig. 5. The research article retrieval methodology comprises three steps: identification, screening and eligibility, and inclusion. In the identification stage, we chose pertinent keywords to fetch research articles pertaining to the investigated research topic. During the screening and eligibility stage, we established specific criteria to sift through literature that did not align with the research needs and objectives. Finally, in the inclusion step, we determined and incorporated literature that met the study's criteria and requirements.

4. Towards edge computing & federated learning

4.1. LLMs on edge

Rapid advances in AI technology, especially in LLMs, are dramatically reshaping the way we interact with external information [134, 135]. However, along with the widespread popularity of progress like ChatGPT comes the discussion around their personalization, security, and other aspects. While LLMs are trained on large databases, there has been a trend toward deploying them to edge devices, including smartphones, laptops, and IoT devices [136–138]. This strategic shift allows models to run locally, increasing processing speeds, and reducing data transfer latency and location awareness, thereby significantly enhancing the user experience [139]. In addition, deploying edge devices reduces the pressure on cloud computing. By offloading some of the computation and processing tasks to these edge devices, the reliance on cloud servers can be reduced, thereby increasing the efficiency of the overall system [140]. The transition of LLM from data centers to edge devices introduces a distinct set of challenges [51]. Owing to the constraints of edge devices concerning memory, capacity, and computational capabilities, directly deploying complete LLMs onto these devices is impractical. In recent years, numerous research endeavors have proposed solutions to facilitate LLM deployment at the edge. The most mainstream solution is to encourage compression of the model to reduce its parameter size, thereby enhancing its applicability to edge devices. In Section 4.1.2, such methods are introduced. By eliminating redundant and unnecessary parameters and getting rid of the burden of heavy computational parameters, the researchers successfully developed a lightweight model tailored for edge computing. Furthermore, specialized inference engines and operational frameworks optimized for edge devices have emerged, aiming to efficiently harness the limited resources available on these devices. In Section 4.1.1, we outline the primary approaches designed for training and deploying LLMs on the edge, along with a discussion of their respective limitations.

Table 1
Hardware configurations for experimental studies in the literature.

Hardware configurations	Relevant studies
NVIDIA GTX 1080 GPUs	[141]
NVIDIA RTX 2080 Ti GPUs	[142]
NVIDIA RTX 3080 GPUs	[143]
NVIDIA RTX 3090 GPUs	[142,144–146]
NVIDIA RTX 4070 GPUs	[147]
NVIDIA RTX 4090 GPUs	[147,148]
NVIDIA Tesla T4 GPUs	[149]
NVIDIA Tesla V100 GPUs	[142,150–152]
NVIDIA A10 GPUs	[153]
NVIDIA A100 GPUs	[141,142,149,154–171]
NVIDIA RTX A6000 GPUs	[163]
NVIDIA Titan RTX GPUs	[172]
Azure NDv5 H100 GPUs	[173]
NVIDIA Jetson AGX	[162]
NVIDIA Jetson NX	[147,170,174]
NVIDIA Jetson TX2	[169,174,175]
NVIDIA Jetson Nano	[176]
Xiaomi 10 and Xiaomi 12	[174]
Redmi 10X Pro, Redmi K50, Mi 10 Lite	[177]
Samsung S23	[178]
Google Pixel 7 Pro, 8 Pro	[169,170,178–180]
Raspberry Pi 4B	[175,176]
Android devices	[181]
Snapdragon CPU and DSP, Apple M1, Microcontrollers	[176]
CUBOT X30	[179]
OPPO Reno 6	[182]

4.1.1. Edge fine-tuning vs. edge inference

Deploying LLMs at the edge offers a promising solution, enabling models to take advantage of the data proximity at the edge and mitigating various risks, such as latency and potential data leakage associated with cloud transmission. There are some research efforts dedicated to enabling running LLMs on edge devices, which can be categorized as edge fine-tuning and edge inference.

- **Edge fine-tuning.** Fine-tuning LLMs on edge devices, especially mobile platforms like smartphones and laptops, demands a substantial allocation of memory resources. According to the findings in [177], fine-tuning a BERT model with a batch size of 8 on a smartphone Redmi K10X Pro resulted in a memory utilization of approximately 5 GB. Given the typically limited RAM capacity of contemporary mobile devices, dedicating a significant portion of memory to LLMs may hinder the concurrent running of other applications.
  Despite the challenges that cannot be underestimated, on-device training is a promising solution. It allows pre-trained LLM to be personalized to the user's local data without sending it to the cloud [183]. PockEngine, as presented in [176], serves as an engine for edge training that can undergo fine-tuning on various edge devices. It incorporates a sparse backpropagation method for efficient low-latency edge training. PockEngine facilitates deployment on edge devices and ensures the capability to fine-tune models on such devices with the support of consumer-level GPUs. In the attempt to personalize LLMs for edge devices, [153] introduces a scheme that autonomously identifies and stores the most relevant data in a self-supervised manner, with the selected data having a reduced memory footprint suitable for user annotations during fine-tuning. Another approach, proposed by [141], establishes a collaborative fine-tuning framework where edge users leverage their local data to train the initial layers of the adapter, while the remaining layers remain frozen. The server then receives these trained parameters and updates the subsequent layers. Similarly, NetGPT [184] enables cloud–edge collaborative training by deploying smaller LLMs at the edges and larger LLMs in the cloud. In the work of [178], cascading is utilized to combine mobile agents with server models, and performance on text rewriting tasks is demonstrated through instruction tuning of on-device models. Recognizing the underutilization of consumer-level GPUs, [143] proposes a strategy where the workload of LLMs is decomposed and distributed across devices with restricted memory capacity. Experimental results indicate that 50 RTX 3080 GPUs can match the throughput of 4 H100 GPUs, yielding substantial cost savings.
- **Edge inference.** Model inference refers to using a trained model to predict or classify unseen data. Edge inference does not require a complicated training process and is easier to implement than edge fine-tuning. However, in edge inference, important indicators that must be paid attention to are memory usage and inference latency [185]. It is crucial to take corresponding measures to reduce the model size [186]. Moreover, to minimize inference latency, it is essential to integrate LLMs into devices situated near the end-user [187]. Existing programs have taken the lead in taking such measures. The `llama.cpp` [188] is a program developed in C++ that quantifies the original LLaMa model using 4 bit integers, enabling inference on edge devices with limited capacity such as MacBook. The program MLC LLM [189] leverages compilation technology to facilitate local development, optimization, and deployment of LLMs on personal devices. This allows the execution of quantized 7B models on smartphones.
  In this part of the research work, efficient inference engines for edge devices have been implemented and produced impressive results. EdgeMoE [175] serves as a device-side engine where non-expert weights reside in the device's memory, while expert weights are stored externally and loaded into memory only when activated. This model partitioning not only conserves memory but also enhances computing efficiency. Another device-side inference engine, LLMCAD [174], utilizes compact LLMs with smaller memory footprints to generate simple tokens, and a high-precision LLM is employed to verify the accuracy of those tokens. In the work of [148], a staged speculative decoding method is introduced to expedite on-device inference for small batches. Experimental results, using the GPT-2-L model with a parameter size of 762M, demonstrate a 3.16 times reduction in latency. [190] propose Agile-Quant, an activation-guided quantization framework to accelerate inference of LLMs on edge devices. It combines a simplified activation quantization with an activation-aware tag-trimming technique to reduce outliers and improve attention efficiency. Using SIMD-based 4-bit multipliers and optimized TRIP matrix multiplication, Agile-Quant delivers up to 2.55x speedup over FP16 models in 8-bit and 4-bit weight quantization scenarios on various edge devices.

The scenarios depicted in Fig. 6 highlight various possibilities for edge deployment. The evolution of LLMs towards EL presents substantial opportunities for enhancing domains such as smart cities [191], smart transportation [192], and smart manufacturing [193]. For example, in scenario 1, the edge device handles the training of initial layers, thereby reducing the need for local training [194]. In scenario 3, a range of model compression techniques are utilized to optimize the model, thus improving the efficiency of both edge training and edge inference [195].

4.1.2. Model compression: enabling edge computing

Presently, a prevalent strategy in research to tailor LLMs for edge computing involves compressing the model [196]. This compression is achieved through various methods, with model pruning, model quantization, and knowledge distillation (KD) being the most commonly utilized techniques.

- **Model quantization.** This process aims to reduce the size of Language Models (LLMs) by modifying how model weights are

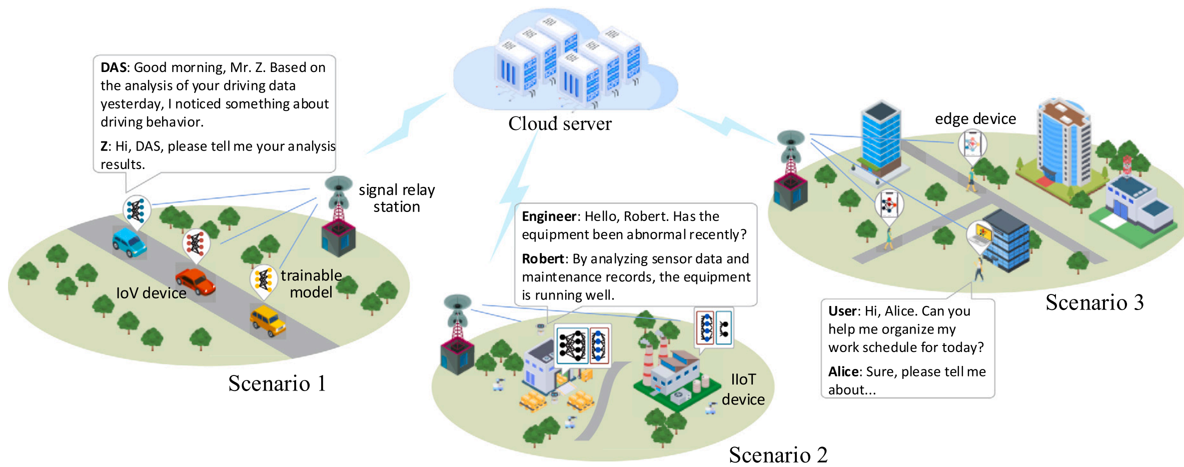

Fig. 6. Edge deployment and application scenes of LLMs in smart cities, smart transportation, and smart manufacturing. **Scenario 1**: Driving Analysis System (DAS) offers personalized driving behavior analysis by utilizing real-time data, and delivering tailored insights and recommendations to Mr. Z. **Scenario 2**: Robert reports to the engineer about the operation of the equipment by examining sensor data and maintenance records. **Scenario 3**: Alice responds to a user's request for assistance with the work schedule, prompting the user to provide additional details.

stored [197]. Typically, deep neural network weights are stored as 32-bit floating-point numbers. Quantization, as discussed in literature [198], involves using tensors with fewer bits, commonly reducing them to 16 bits, 8 bits, and 4 bits. Model quantization is further categorized into weight-only quantization and weight-activated quantization based on the quantization scope [159,199]. This distinction is essential because the activation function is more sensitive to quantization. However, retaining softmax without quantization or maintaining higher accuracy might introduce additional latency [200]. The quantization process is further detailed based on execution steps, with two main approaches: post-training quantization (PTQ) and quantization-aware training (QAT) [201]. PTQ involves converting a pre-trained model into a quantized version without additional training, making it faster and more cost-effective. In contrast, QAT employs an extended training process that simulates quantization effects, ensuring the model's adaptability to reduced precision without compromising performance.

The studies conducted by [105,202] demonstrate that Transformer models like BERT and GPT-3 can significantly enhance memory efficiency and speed by reducing the precision of weights and activations to INT8. Another innovative approach, presented in [163], introduces Sparse Quantized Representation (SpQR), which encodes and decodes weights during quantization, resulting in a 15% speedup for running LLMs. To further enhance the flexibility of quantization, [150] introduces the Quantization-Aware Low-Rank Adaptive (QA-LoRA) technique. This method involves quantizing LLM weights to INT4 during fine-tuning, incorporating both LLM and auxiliary weights into the quantized model without compromising accuracy. In the pursuit of optimizing weight quantization, [203] explores the use of low-precision floating-point (FP) numbers (FP8 and FP4) for LLMs. Interestingly, the study reveals that FP4 achieves equivalent performance to INT4, and in models with over 1 billion parameters, the performance advantage of FP8 over INT8 becomes more pronounced. Recognizing the complexity and diversity of tensor distribution in quantization, [204,205] highlights the importance of using different quantization formats for different layers. The research recommends adopting a mixed format approach for quantization to achieve optimal results.

Model quantization is presently one of the widely adopted compression methods, effectively mitigating both model storage requirements and inference overhead [173]. Nevertheless, reducing model parameters from floating-point representation to lower bit widths, this process may lead to a sacrifice in accuracy [206].

- **Model pruning.** The purpose of model pruning is achieved by shaping the weights of LLMs. Two prominent types of model pruning, namely structured pruning and semi-structured pruning, are outlined in [142]. Structured model pruning works by reducing the number of layers in the model or attention heads in a Transformer, as illustrated by [207]. On the other hand, semi-structured model pruning removes specific weights by setting them to zero, as explained by [208]. Typically, weight saliency metrics are employed in related studies to quantify the accuracy loss of accuracy resulting from model pruning.
  In their work, [142] performed unstructured weight pruning on the BERT model during both the pre-training and fine-tuning stages. They demonstrated a remarkable 10 times compression in model size, leading to a tenfold acceleration in CPU inference with only a 1% sacrifice in accuracy. While weight pruning proves effective for sparse LLMs, it often necessitates multiple rounds of fine-tuning or retraining to ensure optimal performance [209]. Given the substantial sizes of LLMs and the extensive datasets required for their training, the prospect of repeated retraining poses a challenge. In response, recent research efforts have shifted towards one-shot unstructured pruning without the need for additional fine-tuning [210–213]. Additionally, [156] introduces an iterative weight pruning method for sparse LLMs, aiming to minimize the reconstruction error between dense and sparse LLMs.
  Model pruning is a potentially effective technique, and utilizing model pruning requires a structural understanding of the model. However, some model architectures may not be suitable for model pruning because the complex structure and dependencies in the original model may be destroyed during the pruning process, resulting in performance degradation. Furthermore, determining the optimal pruning ratio is a challenge, as over- or under-pruning may affect model performance [214].
- **Knowledge distillation.** The concept of knowledge distillation (KD), initially introduced by [215], involves using a larger teacher model to guide the training of a smaller student model. In the context of challenges faced by LLMs in deployment on resource-constrained devices, KD has gained significant attention as an effective method for compressing models. Broadly, KD can take place in two stages: pre-training and fine-tuning of LLMs. Task-agnostic KD, as indicated by [216–218], refers to distillation performed in the pre-training stage, while task-specific KD, as mentioned by [219–221], involves first fine-tuning the LLM for downstream tasks and then distilling it. Both approaches typically

entail comparing the output distributions of the student and teacher models.

In a related study, [222] introduced a method named pQRNN, utilizing a pre-trained mBERT fine-tuned for semantic parsing tasks as a teacher model. The experimental results demonstrated a student model performance of 95.9% compared to the teacher model, accompanied by a reduction in model size by a factor of 350. Another approach proposed by [223] involves distilling knowledge from a BERT model into a single-layer BiLSTM, reducing model parameters by approximately 100 times. This resulted in a reduction of inference times by a factor of 15 on tasks such as paraphrasing, natural language reasoning, and sentiment classification.

It is crucial to acknowledge that while KD proves effective for model compression, it has certain limitations when implemented with LLMs. The need to calculate the difference in output distributions between the teacher and student models can lead to an increased computational burden during training, especially on edge devices with constrained resources. Additionally, due to the reduced capacity of the student model, it may not comprehensively inherit all the knowledge of LLMs, resulting in the loss of some information [224].

4.2. LLMs & federated learning

Modern distributed computing techniques, such as FL, offer a means of model training without the need for centralized data, thereby providing a level of privacy protection [225]. In the context of LM applications, FL is particularly attractive due to its effectiveness in addressing the challenges posed by distributed data [225,226]. Many real-world scenarios necessitate the use of data on edge devices instead of centralized servers, and this decentralized approach significantly enhances the efficiency of user data protection [227,228]. In Section 4.2.1, we delve into a detailed discussion of research efforts focused on preserving privacy in LLMs. In addition, a typical FL deployment strategy involves utilizing a large public dataset stored on a central server to initially fine-tune the pre-trained base LLM. Subsequently, pre-tuned models serve as the initialization for client models, with clients utilizing their private datasets for further fine-tuning [229]. In Section 4.2.2, we examine typical methods for parameter-efficient fine-tuning in LLM and explore how these methods are integrated with FL.

4.2.1. Privacy protection

The applications of LLM are interconnected with various aspects of our lives, necessitating a heightened focus on issues of data security and privacy protection [230,231]. For example, ChatDoctor [232] was developed for online medical consultation. In this scenario, users are required to transmit their medical information to a cloud system, inevitably risking the exposure of their data. Similarly, Chat-GPT is widely used that aid users in answering questions, providing solutions, and even generating code. However, this model necessitates users to submit their queries to the server, raising concerns, especially involving personal information or confidential commercial data [145]. LLM based on FL presents new solutions to address these privacy challenges in various sensitive scenarios. recent research indicates an emerging trend in integrating LLMs with the FL paradigm [227,233]. Horizontal FL is the most common form of FL, each client maintains the same model and forms a new global model by aggregating these models. [233] proposes the framework FewFedWeight, using BART-Base as a client and global model in FL, and experiments on 118 NLP tasks demonstrate its effectiveness in small-sample generalization and privacy preservation for multi-task learning. FedMLSecurity [168] is a benchmark dedicated to attacks and defenses in federated LLM. It consists of two main components, one simulates attacks injected during FL training, and another one simulates defense mechanisms to mitigate the impact of attacks. This research work demonstrates different security problems of a variety of LLMs faced in real-world applications. Vertical FL is supposed to safeguard user input and model knowledge by partitioning the model into bottom and top parts. Nevertheless, as highlighted in the article [145], there is a potential for privacy leaks through the reconstruction of input from intermediate embeddings. The article then presents a solution to address the input reconstruction attack specifically targeting vertically joint LLMs.

4.2.2. Parameter-efficient fine-tuning

In the field of FL for LLMs, optimizing model updates while considering limited communication and computing resources is crucial. To alleviate the burden of frequent model transmission, a common approach involves the use of Parameter-Efficient Fine-Tuning (PEFT) methods on client models [234]. These methods typically focus on minimizing the number of parameter updates rather than fine-tuning the entire model. This optimization reduces communication and computing costs while still preserving model quality [235]. Traditional fine-tuning involves extending the training of a pre-trained LLM to a specific task or dataset. However, complete parameter fine-tuning becomes impractical for LLMs due to the need to update all parameters and generate distinct instances for various tasks, leading to substantial memory consumption [236]. In response to these challenges, parameter-efficient fine-tuning selectively updates only a small subset of parameters while keeping the majority frozen [237]. This cost-effective strategy is favored by many researchers, especially in edge deployments where computational resources are constrained.

PEFT plays a significant role in fine-tuning parameters for LLMs and is widely applied across training contexts rooted in FL. Fig. 7 (a) illustrates FL employing PEFT where clients possess identical LLM architectures [238–240]. Furthermore, when devices exhibit heterogeneity resulting in models of varying sizes, the KD technique outlined in Section 4.1.2 can facilitate federated training of LLMs [166], as depicted in Fig. 7(b).

In [241], federated training was conducted on multiple clients that only updated adapters and classification headers. The evaluation results demonstrated a reduction in training time by about 20-40%, along with a more than 98% increase in transmission speed. Another notable approach is presented in [154], where PrivateLoRA integrates three low-rank matrices for weight adaptation. The two non-trainable metrics are deployed in the cloud, while the trainable metric resides on the edge device. This proposal guides the transformer toward personalized output and achieves privacy preservation. Furthermore, [172] introduces FedIT, a method that utilizes instructions stored on different local devices and performs instruction tuning via FL. This approach ensures privacy preservation and data security by leveraging instructions in the fine-tuning process.

4.3. Distributed learning

Distributed learning (DistL) involves distributing the computational workload of a ML task across multiple computers or network nodes, particularly when handling resource-intensive computations [242]. This approach encompasses two primary strategies: data parallelization and model parallelization. In data parallelization, the dataset is fragmented into smaller subsets, with each subset processed by a distinct machine or node [160]. On the other hand, model parallelization involves distributing various components of the model across multiple machines [243]. Each machine performs computations for a portion of the model's operations, and their outputs are combined to produce the final result. These methods enable the handling of larger models, quicker training times, and improved fault tolerance of compute node failures [171]. Fig. 8 depicts schematic diagrams illustrating efficient LLM training via model parallelization. In (a), the standard split learning approach in the distributed paradigm of LLM is shown, where the head sub-model is deployed on the edge device while the larger tail sub-model is trained in the cloud [244,245]. In (b), complete

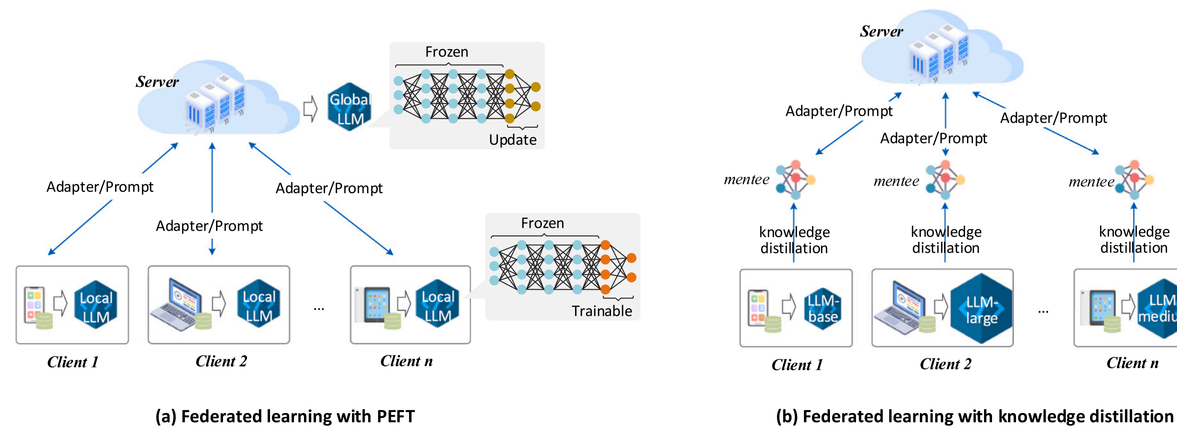

Fig. 7. Illustration of FL for LLMs, showcasing (a) the application of PEFT with clients possessing identical LLM architectures, and (b) the extension to federated training accommodating models of varying sizes through KD.

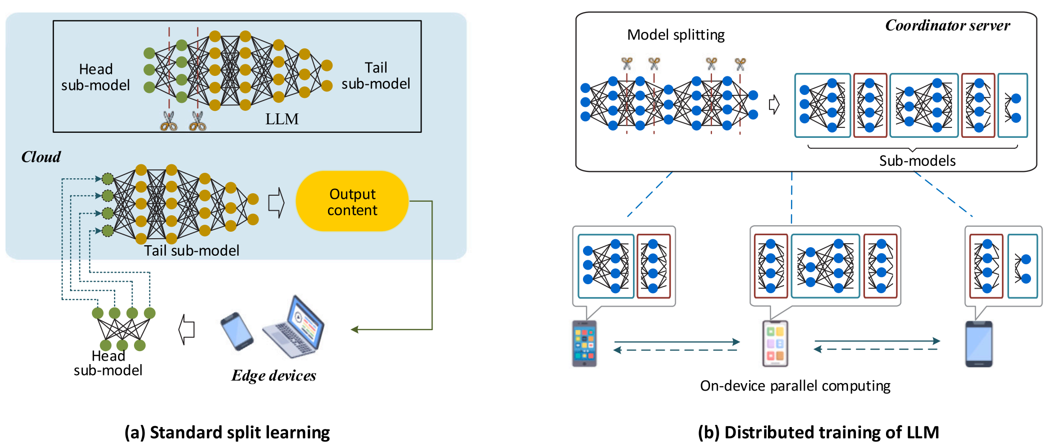

Fig. 8. Distributed learning for LLMs. (a) Standard split learning: head sub-model on edge for local processing, larger tail sub-model trained in cloud for computational resources. (b) Complete model parallel distributed training: model segments processed in parallel across distributed resources, enhancing training efficiency and resource use.

model parallel distributed training is shown [179]. DistL plays a crucial role in the training of LLMs, addressing the substantial computational challenges associated with processing extensive datasets and numerous parameters [246]. LLMs, especially those utilized in NLP tasks, often boast billions or even trillions of parameters, demanding sophisticated techniques to distribute the computational workload effectively. Attempting to train these massive models on a single machine becomes impractical due to the sheer computational requirements and memory constraints. Even the most powerful GPUs struggle to accommodate the parameters of these models in their memory. Moreover, the multitude of computing operations needed can lead to excessively long training times unless careful attention is given to optimizing the algorithms, software, and hardware stack collectively [247].

This is where frameworks like Megatron [248,249] and DeepSpeed [250] come into play, offering solutions to optimize the training of LLMs in distributed environments. Megatron (1, 2, and 3), developed by the Applied Deep Learning Research team at NVIDIA, is a large, powerful prominent framework specifically tailored for the distributed training of LLMs. It places a strong emphasis on model parallelism, enabling the distribution of model components across multiple GPUs. Megatron excels in handling models with trillions of parameters and is designed to make optimal use of GPU computing resources within heterogeneous clusters. By leveraging parallel processing, Megatron significantly accelerates the training process, making it feasible to train massive language models efficiently. On the other hand, DeepSpeed, developed by Microsoft, serves as a DistL optimization library with a focus on addressing challenges associated with training large models. While it complements distributed learning, DeepSpeed goes beyond by providing various optimization techniques such as model ZeRO, 3D-Parallelism, DeepSpeed-MoE, and ZeRO-Infinity, among others. These optimizations aim to enhance the efficiency of training large models, reduce memory requirements, and improve overall scalability. The collaborative utilization of Megatron's expertise in distributed training and model parallelism, combined with DeepSpeed's optimization capabilities, has enabled the training of the Megatron-Turing NLG 530B model [76]—the most extensive and potent monolithic transformer language model trained to date, boasting a staggering 530 billion parameters. Succeeding the Turing NLG 17B and Megatron-LM, MT-NLG surpasses its predecessors with three times the number of parameters, showcasing unparalleled accuracy across a diverse range of natural language tasks such as completion prediction, reading comprehension, commonsense reasoning, natural language inferences, and word sense disambiguation.

Nonetheless, these widely-used training frameworks encounter difficulties when training LLMs in a heterogeneous Network Interface Card (NIC) environment which involves communication between devices over a network, enabling data transfer and network connectivity. The challenge lies in optimizing GPU utilization within heterogeneous clusters, leading to suboptimal utilization of GPU computing resources [251]. This limitation hampers their efficiency in harnessing the full potential of GPU resources in diverse computing environments. In addressing this challenge, frameworks like Co-CoNet [152] and Holmes [165] have been specifically crafted to streamline the optimization of data, model, and pipeline parallel workloads

Table 2
Open datasets for text generation, text classification.

Type	Task	Dataset name	Relevant studies
Text generation	Natural language inference (Comprehensive)	General Language Understanding Evaluation (GLUE)	[144,175,176,253,253]
		Recognizing Textual Entailment	[254]
		Multi-Genre Natural Language Inference (MultiNLI)	[142,255]
		Natural Instructions	[256]
		Alpaca	[150,153]
		WikiText	[144,164,175,179,200,203,204,229]
		Databricks Dolly 15k	[153,155,172]
		Colossal Clean Crawled Corpus (C4)	[163,203,228]
		RedPajama	[163]
	Question answering	Stanford Question Answering Dataset (SQuAD)	[142,144,145,169,253]
		Question-answering NLI (QNLI)	[254]
		HellaSwag	[146,154,204]
		Physical Interaction: Question Answering (PIQA)	[146,204]
		Quora Question Pairs (QQP)	[142,255]
		Boolean Questions (BoolQ)	[154,254]
		GSM8K	[154,155]
		Stack Overflow Dataset	[228,229,241]
		Reddit Corpus	[229]
	Semantic textual similarity	Microsoft Research Paraphrase Corpus (MRPC)	[254]
		Corpus of Linguistic Acceptability (CoLA)	[254]
	Text summarization	SAMSum	[162,175]
	Cloze	LAMBADA	[204]
Text classification	Sentiment analysis	Stanford Sentiment Treebank (SST-2)	[254,255,257]
		MPQA Opinion Corpus	[254]
		Subjectivity dataset (SUBJ)	[254]
		Movie Reviews (MR)	[254]
		Yelp Review Polarity	[169,225,257,258]
	Topic classification	AG News	[169,225,257]
		YAHOO Dataset	[169]
		TREC-10	[254]

in LLMs. These frameworks aim to enhance the efficiency of handling diverse parallel workloads within LLMs, providing solutions to the complexities posed by heterogeneous computing environments, e.g. network interface cards. FlexModel [164] facilitates the processing of models distributed across multi-GPU and multi-node configurations, thereby enhancing the interpretability of distributed LLMs. On the other hand, LinguaLinked [179] is a system designed for decentralized, distributed LLM inference on mobile devices. Extensively tested across a range of Android devices, it has demonstrated an overall inference speedup ranging from 1.29× to 1.32×. An additional technique, intra-layer model parallelism, addresses memory limitations on devices when handling LLMs. This method achieves this by partitioning individual layers or operators across multiple devices within a distributed cluster of accelerators [252].

5. Dataset and open-source codes

In this section, we provide an overview of the datasets and open-source codes in the surveyed papers. These datasets encompass a diverse range of types and purposes, offering LLM customization opportunities. Furthermore, the availability of open-source codes enhances the reproducibility of research endeavors and establishes benchmarks for future studies.

5.1. Dataset

The datasets used in the surveyed papers can be categorized into two main groups: text generation, and text classification. Table 2 reports the dataset and relevant studies within each type.

Text generation. Text generation encompasses datasets specifically tailored to facilitate text-generation tasks, serving as foundational resources for training LLM focused on generating coherent and contextually relevant language. These datasets span various applications, including but not limited to multiple-choice tasks, dialogue generation for conversational agents, sentence completion challenges, and predicting the next word in a sequence, providing diverse avenues for exploring the intricacies of natural language generation. Question-answer (Q&A) sets consist of questions paired with corresponding responses. They are commonly employed in language modeling projects for training models to perform question-answering tasks or to understand queries and generate appropriate responses. Semantic Textual Similarity (STS) datasets contain pairs of sentences or text fragments annotated with similarity scores, indicating their semantic similarity or relatedness. STS datasets are widely used for training and evaluating NLP models in tasks like paraphrase detection, duplicate detection, text similarity assessment, and information retrieval. Text Summarization datasets pair documents or articles with corresponding summaries, where the summaries provide concise representations of the main points or key information contained in the original text. These datasets enable the development and evaluation of models capable of automatically condensing large amounts of information into informative summaries. A cloze dataset typically consists of sentences or passages with one or more words removed, and the task is to predict the missing words based on the context provided. These datasets are commonly used for evaluating language understanding and completion tasks, as well as training language models.

Text classification. Text classification group encompasses a diverse array of datasets specifically curated to facilitate endeavors in text classification. These datasets serve as foundational resources extensively employed to bolster the training of LLM geared towards adeptly classifying text, discerning, and assigning predetermined attributes or classes based on the textual content's characteristics. Within this collection, one finds datasets tailored for various applications, ranging from topic categorization and sentiment analysis to a myriad of other text classification tasks, thereby underpinning the advancement and efficacy of NLP techniques and algorithms. Sentiment Analysis datasets contain text samples annotated with sentiment labels indicating the emotional polarity of the text, while Topic Classification datasets involve assigning topics or categories to text documents based on their content.

Table 3
Open source projects in surveyed papers.

Paper	Year	Framework	Problem	Link
[144]	2021	Edge	Model compression	https://github.com/MohammadrezaBanaei/orientation_based_embedding_compression
[142]	2022	Edge	Model compression	https://github.com/neuralmagic/sparseml/tree/main/research/optimal_BERT_surgeon_oBERT
[253]	2023	Edge	System design	https://github.com/SamsungLabs/Sparse-Multi-DNN-Scheduling
[150]	2023	Edge	Model compression	https://github.com/yuhuixu1993/qa-lora
[259]	2023	Edge	Model compression	https://github.com/microsoft/DeepSpeed
[163]	2023	Edge	Model compression	https://github.com/Vahe1994/SpQR
[147]	2023	Edge	Model compression	https://github.com/mit-han-lab/llm-awq
[156]	2023	Edge	Fine-tuning	https://github.com/zyxxmu/DSnoT
[159]	2023	Edge	Model compression	https://github.com/OpenGVLab/OmniQuant
[205]	2023	Edge	Model compression	https://github.com/lightmatter-ai/INT-FP-QSim
[130]	2024	Edge	Fine-tuning	https://github.com/ZO-Bench/ZO-LLM
[225]	2021	Federated	Fine-tuning	https://github.com/statDataAnalyzer/scaling_fl
[172]	2023	Federated	Fine-tuning	https://github.com/JayZhang42/FederatedGPT-Shepherd
[173]	2023	Federated	Model compression	https://github.com/Azure/MS-AMP
[244]	2023	Federated	Split computing	https://github.com/nishio-laboratory/lambda_split
[168]	2023	Federated	Attacks in FL	https://github.com/FedML-AI/FedML/tree/master/python/fedml/core/security
[161]	2023	Federated	Fine-tuning	https://github.com/yuelinan/FedJudge
[155]	2023	Federated	PEFT	https://github.com/alibaba/FederatedScope/tree/llm
[166]	2023	Federated	Fine-tuning	https://github.com/FederatedAI/FATE-LLM
[146]	2023	Federated	Fine-tuning	https://github.com/alibaba/FederatedScope/tree/fedsp/federatedscope/nlp/fedsp
[254]	2023	Federated	Fine-tuning	https://github.com/llm-eff/FedPepTAO
[127]	2024	Federated	Attacks in FL	https://github.com/FedML-AI/FedML/tree/master/python/fedml/core/security
[128]	2024	Federated	Instruction tuning	https://github.com/rui-ye/OpenFedLLM
[240]	2024	Federated	Fine-tuning	https://github.com/UbiquitousLearning/FwdLLM
[152]	2022	Distributed	Model parallel	https://github.com/parasailteam/coconet
[164]	2023	Distributed	Model parallel	https://github.com/VectorInstitute/flex_model
[260]	2023	Distributed	Model parallel	https://github.com/xuqifan897/Optimus
[246]	2024	Distributed	Model parallel	https://github.com/zjc664656505/LinguaLinked-Inference

Table 4
Open source projects without paper.

Project name	About	Link
LeapfrogAI	Production-ready Generative AI for local, cloud native, airgap, and edge.	https://github.com/defenseunicorns/leapfrogai
llama-utils	The easiest & fastest way to run customized and fine-tuned LLMs locally or on the edge.	https://github.com/second-state/llama-utils
LLM API	Fully typed & consistent chat APIs for OpenAI, Anthropic, Azure's chat models for browser, edge, and node environments.	https://github.com/dzhng/llm-api
balena-serge	Run an LLM on your edge device with balena.io.	https://github.com/klutchell/balena-serge
Edge Infer	EdgeInfer enables efficient edge intelligence by running small AI models, including embeddings and OnnxModels, on resource-constrained devices like Android, iOS, or MCUs for real-time decision-making.	https://github.com/unit-mesh/edge-infer
llama4j	An easy-to-use Java SDK for running LLaMA models on edge devices, powered by LLaMA.cpp.	https://github.com/JavaLLM/llama4j
llm-edge-web	Web app for LLMs on EDGE devices.	https://github.com/timothyoei/llm-edge-web
LLM InferenceNet	LLM InferenceNet is a C++ based project designed to achieve fast inference from LLMs by leveraging a client–server architecture.	https://github.com/adithya-s-k/LLM-InferenceNet
llama.cpp	Inference of LLMs in pure C/C++	https://github.com/ggerganov/llama.cpp
SplitLLM	LLM, Vertical Federated Learning, and some funny experiments.	https://github.com/zfscgy/SplitLLM
fedGPT	An implementation of training nanoGPT through Federated Learning and implementing Differential Privacy	https://github.com/aneesh-aparajit/fedGPT/tree/main

5.2. Open-source codes

Numerous authors have contributed to the proliferation of open-source implementations of their proposed models, thereby fostering accessibility and collaboration within the research community. In Tables 3 and 4, we provide the overview of the open-source projects dedicated to addressing relevant challenges with LLMs. Notably, these research works widely leverage frameworks such as TensorFlow, PyTorch, and FATE, and Python and C++ as the language of choice for building LLMs. This concerted effort toward openness and standardization promotes reproducibility and facilitates innovation and advancement in NLP research.

6. Open issues and conclusion

Navigating the landscape of LLMs in the context of edge and federated learning brings forth a set of challenges and opens avenues for intriguing future directions. These aspects are crucial to consider as they shape the trajectory of advancements in this intersection of technologies. Deploying LLMs on edge devices introduces resource constraints, such as limited computational power and memory. One of the primary challenges lies in adapting LLMs to the inherent resource constraints of edge devices. These devices, characterized by limited computational power and memory, demand specialized optimization techniques to ensure that LLMs operate efficiently without compromising performance.

In the realm of FL, communication overhead emerges as a critical bottleneck. The process of transmitting model updates between edge devices and a central server can be hampered by unreliable or constrained networks, necessitating the exploration of more efficient communication protocols. Privacy considerations loom large in FL scenarios, where models are trained locally on edge devices. Balancing the need for model updates with user privacy becomes a delicate challenge, urging researchers to develop robust privacy-preserving mechanisms. Interpreting the decisions made by LLMs at the edge presents a non-trivial task. Ensuring transparency and interpretability of these models is essential, especially when deployed in applications where the rationale behind decisions is crucial, such as in healthcare or finance.

The trajectory of research in this domain points towards the exploration of specialized optimization techniques for edge-deployed LLMs. Techniques encompassing model compression, quantization, and architectural modifications are envisioned to be at the forefront, catering to the unique resource constraints of edge devices. Efforts in FL should focus on refining communication protocols. Techniques such as model sparsity and differential privacy may prove instrumental in reducing the volume of information transmitted between edge devices and central servers, mitigating communication overhead. Enhancing privacy-preserving mechanisms in FL is crucial for the widespread adoption of this paradigm. Future research may delve into advanced cryptographic techniques or novel FL frameworks that prioritize user privacy without compromising the performance of the trained models. In the quest for interpretable models at the edge, research endeavors are anticipated to focus on providing insights into the decision-making process of LLMs. Real-time interpretability is paramount, especially in applications where understanding model decisions is critical for user trust and compliance with regulations.

In conclusion, the fusion of LLMs with edge and FL pushes researchers to address challenges collaboratively. The journey ahead involves not only overcoming hurdles but also shaping the landscape of distributed, privacy-aware, and interpretable LMs that cater to the evolving needs of diverse applications.

CRediT authorship contribution statement

Francesco Piccialli: Conceptualization, Formal analysis, Methodology, Supervision, Writing – original draft. **Diletta Chiaro:** Investigation, Methodology, Supervision. **Pian Qi:** Investigation, Resources, Visualization, Writing – original draft. **Valerio Bellandi:** Conceptualization, Writing – original draft, Writing – review & editing. **Ernesto Damiani:** Supervision, Validation, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

- PNRR project FAIR - Future AI Research (PE00000013), Spoke 3, under the NRRP MUR program funded by the NextGenerationEU.
- G.A.N.D.A.L.F. - Gan Approaches for Non-iiD Aiding Learning in Federations, CUP: E53D23008290006, PNRR - Missione 4 “Istruzione e Ricerca” - Componente C2 Investimento 1.1 “Fondo per il Programma Nazionale di Ricerca e Progetti di Rilevante Interesse Nazionale (PRIN)”.

Data availability

No data was used for the research described in the article.

References

[1] S. Pinker, The Language Instinct: how the Mind Creates Language, Penguin uK, 2003.

[2] A.M. Turing, Computing machinery and intelligence., Creat. Comput. 6 (1) (1980) 44–53.

[3] K. Chowdhary, K. Chowdhary, Natural language processing, Fundam. Artif. Intell. (2020) 603–649.

[4] B. Min, H. Ross, E. Sulem, A.P.B. Veyseh, T.H. Nguyen, O. Sainz, E. Agirre, I. Heintz, D. Roth, Recent advances in natural language processing via large pre-trained language models: A survey, ACM Comput. Surv. 56 (2) (2023) 1–40.

[5] N. Omar, Q. Al-Tashi, Arabic nested noun compound extraction based on linguistic features and statistical measures, GEMA Online® J. Lang. Stud. 18 (2) (2018).

[6] S. Diao, R. Xu, H. Su, Y. Jiang, Y. Song, T. Zhang, Taming pre-trained language models with n-gram representations for low-resource domain adaptation, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 3336–3349.

[7] V.A. Petrushin, Hidden markov models: Fundamentals and applications, in: Online Symposium for Electronics Engineer, 2000.

[8] S. Khudanpur, J. Wu, Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling, Comput. Speech Lang. 14 (4) (2000) 355–372.

[9] H. Wang, J. He, X. Zhang, S. Liu, A short text classification method based on N-gram and CNN, Chin. J. Electron. 29 (2) (2020) 248–254.

[10] R. Rosenfeld, Two decades of statistical language modeling: Where do we go from here? Proc. IEEE 88 (8) (2000) 1270–1278.

[11] E. Arisoy, T.N. Sainath, B. Kingsbury, B. Ramabhadran, Deep neural network language models, in: Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-Gram Model? on the Future of Language Modeling for HLT, 2012, pp. 20–28.

[12] J.R. Bellegarda, Exploiting latent semantic information in statistical language modeling, Proc. IEEE 88 (8) (2000) 1279–1296.

[13] F. Alva-Manchego, C. Scarton, L. Specia, Data-driven sentence simplification: Survey and benchmark, Comput. Linguist. 46 (1) (2020) 135–187.

[14] M. Malik, M.K. Malik, K. Mehmood, I. Makhdoom, Automatic speech recognition: a survey, Multimedia Tools Appl. 80 (2021) 9411–9457.

[15] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing 408 (2020) 189–215.

[16] M. Crawford, T.M. Khoshgoftaar, J.D. Prusa, A.N. Richter, H. Al Najada, Survey of review spam detection using machine learning techniques, J. Big Data 2 (1) (2015) 1–24.

[17] M. Neethu, R. Rajasree, Sentiment analysis in twitter using machine learning techniques, in: 2013 Fourth International Conference on Computing, Communications and Networking Technologies, ICCCNT, IEEE, 2013, pp. 1–5.

[18] A. Go, L. Huang, R. Bhayani, Twitter sentiment analysis, Entropy 17 (2009) 252.

[19] Q. Lhoest, A.V. del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, et al., Datasets: A community library for natural language processing, 2021, arXiv preprint arXiv:2109.02846.

[20] O. Sharir, B. Peleg, Y. Shoham, The cost of training nlp models: A concise overview, 2020, arXiv preprint arXiv:2004.08900.

[21] L. Deng, Y. Liu, A joint introduction to natural language processing and to deep learning, Deep Learn. Natl. Lang. Process. (2018) 1–22.

[22] W. Yin, K. Kann, M. Yu, H. Schütze, Comparative study of CNN and RNN for natural language processing, 2017, arXiv preprint arXiv:1702.01923.

[23] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, S. Khudanpur, Recurrent neural network based language model., in: Interspeech, Vol. 2, Makuhari, 2010, pp. 1045–1048.

[24] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[25] S. Hochreiter, Recurrent neural net learning and vanishing gradient, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 6 (2) (1998) 107–116.

[26] S. Hihi, Y. Bengio, Hierarchical recurrent neural networks for long-term dependencies, Adv. Neural Inf. Process. Syst. 8 (1995).

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[28] P. Shaw, J. Uszkoreit, A. Vaswani, Self-attention with relative position representations, 2018, arXiv preprint arXiv:1803.02155.

[29] Q. Liu, M.J. Kusner, P. Blunsom, A survey on contextual embeddings, 2020, arXiv preprint arXiv:2003.07278.

[30] E. Adamopoulou, L. Moussiades, Chatbots: History, technology, and applications, Mach. Learn. Appl. 2 (2020) 100006.

[31] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E.D. Trippe, J.B. Gutierrez, K. Kochut, Text summarization techniques: a brief survey, 2017, arXiv preprint arXiv:1707.02268.

[32] Y. Ge, W. Hua, J. Ji, J. Tan, S. Xu, Y. Zhang, Openagi: When llm meets domain experts, 2023, arXiv preprint arXiv:2304.04370.

[33] K. Adnan, R. Akbar, An analytical study of information extraction from unstructured and multidimensional big data, J. Big Data 6 (1) (2019) 1–38.

[34] M. Awais, M. Naseer, S. Khan, R.M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, F.S. Khan, Foundational models defining a new era in vision: A survey and outlook, 2023, arXiv preprint arXiv:2307.13721.

[35] H. Zhang, X. Li, L. Bing, Video-llama: An instruction-tuned audio-visual language model for video understanding, 2023, arXiv preprint arXiv:2306.02858.

[36] A. Rouditchenko, A. Boggust, D. Harwath, B. Chen, D. Joshi, S. Thomas, K. Audhkhasi, H. Kuehne, R. Panda, R. Feris, et al., Avlnet: Learning audio-visual language representations from instructional videos, 2020, arXiv preprint arXiv:2006.09199.
[37] Y. Zhao, Z. Lin, D. Zhou, Z. Huang, J. Feng, B. Kang, Bubogpt: Enabling visual grounding in multi-modal llms, 2023, arXiv preprint arXiv:2307.08581.
[38] B. Ghojogh, A. Ghodsi, Attention mechanism, transformers, BERT, and GPT: tutorial and survey, 2020.
[39] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H.W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, J. Mach. Learn. Res. 24 (240) (2023) 1–113.
[40] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A.M. Dai, A. Hauth, et al., Gemini: a family of highly capable multimodal models, 2023, arXiv preprint arXiv:2312.11805.
[41] T.M. Gemma Team, C. Hardin, R. Dadashi, S. Bhupatiraju, L. Sifre, M. Rivière, M.S. Kale, J. Love, P. Tafti, L. Hussenot, et al., Gemma, 2024, http://dx.doi.org/10.34740/KAGGLE/M/3301, URL https://www.kaggle.com/m/3301.
[42] K. Tirumala, D. Simig, A. Aghajanyan, A.S. Morcos, D4: Improving llm pre-training via document de-duplication and diversification, 2023, arXiv preprint arXiv:2308.12284.
[43] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D.C. Schmidt, A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023, arXiv preprint arXiv:2302.11382.
[44] F.F. Xu, U. Alon, G. Neubig, V.J. Hellendoorn, A systematic evaluation of large language models of code, in: Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, 2022, pp. 1–10.
[45] 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 620 (7972) (2023) 47–60.
[46] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, Q. Wang, Software testing with large language model: Survey, landscape, and vision, 2023, arXiv preprint arXiv:2307.07221.
[47] A.J. Thirunavukarasu, D.S.J. Ting, K. Elangovan, L. Gutierrez, T.F. Tan, D.S.W. Ting, Large language models in medicine, Nat. Med. 29 (8) (2023) 1930–1940.
[48] J. Cabrera, M.S. Loyola, I. Magaña, R. Rojas, Ethical dilemmas, mental health, artificial intelligence, and llm-based chatbots, in: International Work-Conference on Bioinformatics and Biomedical Engineering, Springer, 2023, pp. 313–326.
[49] A. Creswell, M. Shanahan, I. Higgins, Selection-inference: Exploiting large language models for interpretable logical reasoning, 2022, arXiv preprint arXiv:2205.09712.
[50] E. Ferrara, Should chatgpt be biased? challenges and risks of bias in large language models, 2023, arXiv preprint arXiv:2304.03738.
[51] Z. Lin, G. Qu, Q. Chen, X. Chen, Z. Chen, K. Huang, Pushing large language models to the 6G edge: Vision, challenges, and opportunities, 2023, arXiv preprint arXiv:2309.16739.
[52] X.L. Dong, S. Moon, Y.E. Xu, K. Malik, Z. Yu, Towards next-generation intelligent assistants leveraging llm techniques, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 5792–5793.
[53] Z. Cai, J. Chen, W. Chen, W. Wang, X. Zhu, A. Ouyang, F-codellm: A federated learning framework for adapting large language models to practical software development, in: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, 2024, pp. 416–417.
[54] H. Woisetschläger, A. Erben, S. Wang, R. Mayer, H.-A. Jacobsen, Federated fine-tuning of llms on the very edge: The good, the bad, the ugly, in: Proceedings of the Eighth Workshop on Data Management for End-To-End Machine Learning, 2024, pp. 39–50.
[55] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer, 2020, pp. 213–229.
[56] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45.
[57] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training, 2018.
[58] A. Radford, J. Wu, R. Child, et al., Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019) 9.
[59] T. Brown, B. Mann, N. Ryder, et al., Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.
[60] OpenAI, ChatGPT, 2022, URL https://chat.openai.com/.
[61] A. Josh, A. Steven, A. Sandhini, et al., GPT-4 technical report, 2023, arXiv:2303.08774.
[62] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[63] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019, arXiv preprint arXiv:1910.13461.
[64] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019, arXiv preprint arXiv:1907.11692.
[65] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R.R. Salakhutdinov, Q.V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, Adv. Neural Inf. Process. Syst. 32 (2019).
[66] A. Roberts, C. Raffel, K. Lee, M. Matena, N. Shazeer, P.J. Liu, S. Narang, W. Li, Y. Zhou, Exploring the limits of transfer learning with a unified text-to-text transformer, Tech. Rep., 2019, Google.
[67] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, Z. Chen, Gshard: Scaling giant models with conditional computation and automatic sharding, 2020, arXiv preprint arXiv:2006.16668.
[68] S. Black, L. Gao, P. Wang, C. Leahy, S. Biderman, GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, 2021.
[69] B. Wang, A. Komatsuzaki, GPT-j-6B: A 6 billion parameter autoregressive language model, 2021.
[70] A. Andonian, Q. Anthony, S. Biderman, S. Black, P. Gali, L. Gao, E. Hallahan, J. Levy-Kramer, C. Leahy, L. Nestler, et al., GPT-NeoX: Large scale autoregressive language modeling in pytorch, 2021.
[71] M. Chen, J. Tworek, H. Jun, Q. Yuan, H.P.d. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al., Evaluating large language models trained on code, 2021, arXiv preprint arXiv:2107.03374.
[72] O. Lieber, O. Sharir, B. Lenz, Y. Shoham, Jurassic-1: Technical details and evaluation, White Paper, Vol. 1, AI21 Labs, 2021, p. 9.
[73] Anthropic, Anthropiclm, anthropiclm v4-s3, and claude., 2021, 2022, and 2023, URL https://www.anthropic.com/.
[74] N. Du, Y. Huang, A.M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A.W. Yu, O. Firat, et al., Glam: Efficient scaling of language models with mixture-of-experts, in: International Conference on Machine Learning, PMLR, 2022, pp. 5547–5569.
[75] J.W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al., Scaling language models: Methods, analysis & insights from training gopher, 2021, arXiv preprint arXiv:2112.11446.
[76] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, et al., Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model, 2022, arXiv preprint arXiv:2201.11990.
[77] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al., Training language models to follow instructions with human feedback, Adv. Neural Inf. Process. Syst. 35 (2022) 27730–27744.
[78] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al., Lamda: Language models for dialog applications, 2022, arXiv preprint arXiv:2201.08239.
[79] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D.d.L. Casas, L.A. Hendricks, J. Welbl, A. Clark, et al., Training compute-optimal large language models, 2022, arXiv preprint arXiv:2203.15556.
[80] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X.V. Lin, et al., Opt: Open pre-trained transformer language models, 2022, arXiv preprint arXiv:2205.01068.
[81] B. Workshop, T.L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A.S. Luccioni, F. Yvon, et al., Bloom: A 176b-parameter open-access multilingual language model, 2022, arXiv preprint arXiv:2211.05100.
[82] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T.L. Scao, M.S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf, et al., Crosslingual generalization through multitask finetuning, 2022, arXiv preprint arXiv:2211.01786.
[83] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, et al., Solving quantitative reasoning problems with language models, Adv. Neural Inf. Process. Syst. 35 (2022) 3843–3857.
[84] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, et al., Improving alignment of dialogue agents via targeted human judgements, 2022, arXiv preprint arXiv:2209.14375.
[85] H.W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al., Scaling instruction-finetuned language models, 2022, arXiv preprint arXiv:2210.11416.
[86] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, R. Stojnic, Galactica: A large language model for science, 2022, arXiv preprint arXiv:2211.09085.
[87] S. Iyer, X.V. Lin, R. Pasunuru, T. Mihaylov, D. Simig, P. Yu, K. Shuster, T. Wang, Q. Liu, P.S. Koura, et al., Opt-iml: Scaling language model instruction meta learning through the lens of generalization, 2022, arXiv preprint arXiv:2212.12017.
[88] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, 2023, arXiv preprint arXiv:2302.13971.

[89] Google, Bard, 2023, URL https://bard.google.com/?hl=en-GB.
[90] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J.E. Gonzalez, et al., Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023, See https://vicuna.lmsys.org. (Accessed 14 April 2023).
[91] AI21 Labs, Jurassic-2, 2023, URL https://www.ai21.com/blog/introducing-j2.
[92] A.D.T.I. Institute, Falcon 40b/180b, 2023, URL https://falconllm.tii.ae/.
[93] S.G. Patil, T. Zhang, X. Wang, J.E. Gonzalez, Gorilla: Large language model connected with massive apis, 2023, arXiv preprint arXiv:2305.15334.
[94] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, A. Awadallah, Orca: Progressive learning from complex explanation traces of gpt-4, 2023, arXiv preprint arXiv:2306.02707.
[95] A. Mitra, L. Del Corro, S. Mahajan, A. Codas, C. Simoes, S. Agarwal, X. Chen, A. Razdaibiedina, E. Jones, K. Aggarwal, et al., Orca 2: Teaching small language models how to reason, 2023, arXiv preprint arXiv:2311.11045.
[96] S. A.I., Stable LM 2, 2024, URL https://stability.ai/.
[97] I. A.I., Inflection-2.5, 2024, URL https://inflection.ai/.
[98] S. Shahriar, B.D. Lund, N.R. Mannuru, M.A. Arshad, K. Hayawi, R.V.K. Bevara, A. Mannuru, L. Batool, Putting gpt-4o to the sword: A comprehensive evaluation of language, vision, speech, and multimodal proficiency, Appl. Sci. 14 (17) (2024) 7782.
[99] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, et al., Qwen2 technical report, 2024, arXiv preprint arXiv:2407.10671.
[100] xAI, Grok-2, 2024, URL https://x.ai/.
[101] Google DeepMind, Gemini 1.5 flash-8B, 2024, URL https://deepmind.google/technologies/gemini/flash/.
[102] A. Baladón, I. Sastre, L. Chiruzzo, A. Rosá, RETUYT-inco at BEA 2023 shared task: Tuning open-source LLMs for generating teacher responses, in: Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023), 2023, pp. 756–765.
[103] J.J. Nay, Large language models as fiduciaries: A case study toward robustly communicating with artificial intelligence through legal standards, 2023, arXiv preprint arXiv:2301.10095.
[104] T.Y. Zhuo, Z. Li, Y. Huang, Y.-F. Li, W. Wang, G. Haffari, F. Shiri, On robustness of prompt-based semantic parsing with large pre-trained language model: An empirical study on codex, 2023, arXiv preprint arXiv:2301.12868.
[105] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, Y. He, Zeroquant: Efficient and affordable post-training quantization for large-scale transformers, Adv. Neural Inf. Process. Syst. 35 (2022) 27168–27183.
[106] A. Zou, Z. Wang, J.Z. Kolter, M. Fredrikson, Universal and transferable adversarial attacks on aligned language models, 2023, arXiv preprint arXiv:2307.15043.
[107] D.M. Katz, M.J. Bommarito, S. Gao, P. Arredondo, Gpt-4 passes the bar exam, 2023, Available at SSRN 4389233.
[108] K.I. Roumeliotis, N.D. Tselikas, D.K. Nasiopoulos, Llama 2: Early adopters' utilization of meta's new open-source pretrained model, 2023.
[109] A. Byrd, Truth-telling: Critical inquiries on LLMs and the corpus texts that train them., Compos. Stud. 51 (1) (2023) 135–142.
[110] X. Zhang, B. Yu, H. Yu, Y. Lv, T. Liu, F. Huang, H. Xu, Y. Li, Wider and deeper llm networks are fairer llm evaluators, 2023, arXiv preprint arXiv:2308.01862.
[111] I. Yildirim, L. Paul, From task structures to world models: What do LLMs know? 2023, arXiv preprint arXiv:2310.04276.
[112] H. Jin, X. Han, J. Yang, Z. Jiang, C.-Y. Chang, X. Hu, GrowLength: Accelerating LLMs pretraining by progressively growing training length, 2023, arXiv preprint arXiv:2310.00576.
[113] R.V.P. Marcel, B.E.M. Fernando, Y.V.J. Roberto, A brief history of the artificial intelligence: chatgpt: The evolution of GPT, in: 2023 18th Iberian Conference on Information Systems and Technologies, CISTI, IEEE, 2023, pp. 1–5.
[114] E.Y. Chang, Examining GPT-4: Capabilities, implications and future directions, in: The 10th International Conference on Computational Science and Computational Intelligence, 2023.
[115] M. Zhang, J. Li, A commentary of GPT-3 in MIT technology review 2021, Fundam. Res. 1 (6) (2021) 831–833.
[116] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, 2019, arXiv preprint arXiv:1909.11942.
[117] D. Demszky, D. Yang, D.S. Yeager, C.J. Bryan, M. Clapper, S. Chandhok, J.C. Eichstaedt, C. Hecht, J. Jamieson, M. Johnson, et al., Using large language models in psychology, Nat. Rev. Psychol. 2 (11) (2023) 688–701.
[118] M.U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M.B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al., A survey on large language models: Applications, challenges, limitations, and practical usage, Authorea Prepr. (2023).
[119] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, et al., Summary of chatgpt-related research and perspective towards the future of large language models, Meta-Radiol. (2023) 100017.
[120] Y. Yuan, R. Kong, Y. Li, Y. Liu, Wip: An on-device LLM-based approach to query privacy protection, in: Proceedings of the Workshop on Edge and Mobile Foundation Models, 2024, pp. 7–9.
[121] S.M. Hasan, A.M. Alotaibi, S. Talukder, A.R. Shahid, Distributed threat intelligence at the edge devices: A large language model-driven approach, 2024, arXiv preprint arXiv:2405.08755.
[122] Y. He, J. Fang, F.R. Yu, V.C. Leung, Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: An active inference approach, IEEE Trans. Mob. Comput. (2024).
[123] P. Sundaravadivel, P.J. Roselyn, V. Narayanaswamy, V.I. Jeyaraj, A. Ramesh, A. Khanal, Integrating image-based LLMs on edge-devices for underwater robotics, in: Real-Time Image Processing and Deep Learning 2024, Vol. 13034, SPIE, 2024, pp. 119–125.
[124] K. Alizadeh, I. Mirzadeh, D. Belenko, K. Khatamifard, M. Cho, C.C. Del Mundo, M. Rastegari, M. Farajtabar, Llm in a flash: Efficient large language model inference with limited memory, 2023, arXiv preprint arXiv:2312.11514.
[125] X. Yuan, H. Li, K. Ota, M. Dong, Generative inference of large language models in edge computing: An energy efficient approach, in: 2024 International Wireless Communications and Mobile Computing, IWCMC, IEEE, 2024, pp. 244–249.
[126] C. Du, S.-B. Ko, H. Zhang, Energy efficient FPGA-based binary transformer accelerator for edge devices, in: 2024 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2024, pp. 1–5.
[127] S. Han, B. Buyukates, Z. Hu, H. Jin, W. Jin, L. Sun, X. Wang, W. Wu, C. Xie, Y. Yao, et al., Fedsecurity: A benchmark for attacks and defenses in federated learning and federated llms, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 5070–5081.
[128] R. Ye, W. Wang, J. Chai, D. Li, Z. Li, Y. Xu, Y. Du, Y. Wang, S. Chen, Openfedllm: Training large language models on decentralized private data via federated learning, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 6137–6147.
[129] J. Du, T. Lin, C. Jiang, Q. Yang, C.F. Bader, Z. Han, Distributed foundation models for multi-modal learning in 6G wireless networks, IEEE Wirel. Commun. 31 (3) (2024) 20–30.
[130] Y. Zhang, P. Li, J. Hong, J. Li, Y. Zhang, W. Zheng, P.-Y. Chen, J.D. Lee, W. Yin, M. Hong, et al., Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark, 2024, arXiv preprint arXiv:2402.11592.
[131] F. Urbina, C.T. Lowden, J.C. Culberson, S. Ekins, MegaSyn: integrating generative molecular design, automated analog designer, and synthetic viability prediction, ACS Omega 7 (22) (2022) 18699–18713.
[132] F. Wu, Z. Li, Y. Li, B. Ding, J. Gao, Fedbiot: Llm local fine-tuning in federated learning without full model, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 3345–3355.
[133] M.J. Page, J.E. McKenzie, P.M. Bossuyt, I. Boutron, T.C. Hoffmann, C.D. Mulrow, L. Shamseer, J.M. Tetzlaff, E.A. Akl, S.E. Brennan, et al., The PRISMA 2020 statement: an updated guideline for reporting systematic reviews, Int. J. Surg. 88 (2021) 105906.
[134] P. Andrews, O.E. Nordberg, S. Zubicueta Portales, N. Borch, F. Guribye, K. Fujita, M. Fjeld, Aicommentator: A multimodal conversational agent for embedded visualization in football viewing, in: Proceedings of the 29th International Conference on Intelligent User Interfaces, 2024, pp. 14–34.
[135] H. Cui, Y. Du, Q. Yang, Y. Shao, S.C. Liew, LLMind: Orchestrating AI and IoT with LLM for complex task execution, IEEE Commun. Mag. (2024).
[136] N. Zhong, Y. Wang, R. Xiong, Y. Zheng, Y. Li, M. Ouyang, D. Shen, X. Zhu, CASIT: Collective intelligent agent system for internet of things, IEEE Internet Things J. (2024).
[137] X. Li, Z. Lu, D. Cai, X. Ma, M. Xu, Large language models on mobile devices: Measurements, analysis, and insights, in: Proceedings of the Workshop on Edge and Mobile Foundation Models, 2024, pp. 1–6.
[138] M.A. Ferrag, M. Ndhlovu, N. Tihanyi, L.C. Cordeiro, M. Debbah, T. Lestable, N.S. Thandi, Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for iot/iiot devices, IEEE Access (2024).
[139] J. Ren, D. Zhang, S. He, Y. Zhang, T. Li, A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet, ACM Comput. Surv. 52 (6) (2019) 1–36.
[140] Q. Dong, X. Chen, M. Satyanarayanan, Creating edge ai from cloud-based llms, in: Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications, 2024, pp. 8–13.
[141] L. Qian, J. Zhao, User association and resource allocation in large language model based mobile edge computing system over wireless communications, 2023, arXiv preprint arXiv:2310.17872.
[142] E. Kurtic, D. Campos, T. Nguyen, E. Frantar, M. Kurtz, B. Fineran, M. Goin, D. Alistarh, The optimal bert surgeon: Scalable and accurate second-order pruning for large language models, 2022, arXiv preprint arXiv:2203.07259.
[143] Z. Tang, Y. Wang, X. He, L. Zhang, X. Pan, Q. Wang, R. Zeng, K. Zhao, S. Shi, B. He, et al., FusionAI: Decentralized training and deploying LLMs with massive consumer-level GPUs, 2023, arXiv preprint arXiv:2309.01172.
[144] K. Bałazy, M. Banaei, R. Lebret, J. Tabor, K. Aberer, Direction is what you need: Improving word embedding compression in large language models, 2021, arXiv preprint arXiv:2106.08181.
[145] F. Zheng, Input reconstruction attack against vertical federated large language models, 2023, arXiv preprint arXiv:2311.07585.
[146] C. Dong, Y. Xie, B. Ding, Y. Shen, Y. Li, Tunable soft prompts are messengers in federated learning, 2023, arXiv preprint arXiv:2311.06805.

[147] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, S. Han, AWQ: Activation-aware weight quantization for LLM compression and acceleration, 2023, arXiv preprint arXiv:2306.00978.
[148] B. Spector, C. Re, Accelerating llm inference with staged speculative decoding, 2023, arXiv preprint arXiv:2308.04623.
[149] T. Prompt, Compress, Then Prompt: Improving Accuracy-Efficiency Trade-off of LLM Inference with Transferable Prompt.
[150] Y. Xu, L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. Zhang, Q. Tian, Qa-lora: Quantization-aware low-rank adaptation of large language models, 2023, arXiv preprint arXiv:2309.14717.
[151] J. Shi, Z. Yang, H.J. Kang, B. Xu, J. He, D. Lo, Towards smaller, faster, and greener language models of code, 2023, arXiv e-prints, arXiv–2309.
[152] A. Jangda, J. Huang, G. Liu, A.H.N. Sabet, S. Maleki, Y. Miao, M. Musuvathi, T. Mytkowicz, O. Saarikivi, Breaking the computation and communication abstraction barrier in distributed machine learning workloads, in: Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2022, pp. 402–416.
[153] R. Qin, J. Xia, Z. Jia, M. Jiang, A. Abbasi, P. Zhou, J. Hu, Y. Shi, Enabling on-device large language model personalization with self-supervised data selection and synthesis, 2023, arXiv preprint arXiv:2311.12275.
[154] Y. Wang, Y. Lin, X. Zeng, G. Zhang, PrivateLoRA for efficient privacy preserving LLM, 2023, arXiv preprint arXiv:2311.14030.
[155] W. Kuang, B. Qian, Z. Li, D. Chen, D. Gao, X. Pan, Y. Xie, Y. Li, B. Ding, J. Zhou, Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning, 2023, arXiv preprint arXiv:2309.00363.
[156] Y. Zhang, L. Zhao, M. Lin, Y. Sun, Y. Yao, X. Han, J. Tanner, S. Liu, R. Ji, Dynamic sparse no training: Training-free fine-tuning for sparse LLMs, 2023, arXiv preprint arXiv:2310.08915.
[157] I. Mirzadeh, K. Alizadeh, S. Mehta, C.C. Del Mundo, O. Tuzel, G. Samei, M. Rastegari, M. Farajtabar, Relu strikes back: Exploiting activation sparsity in large language models, 2023, arXiv preprint arXiv:2310.04564.
[158] E. Yvinec, A. Dapogny, K. Bailly, Nupes: Non-uniform post-training quantization via power exponent search, 2023, arXiv preprint arXiv:2308.05600.
[159] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, P. Luo, Omniquant: Omnidirectionally calibrated quantization for large language models, 2023, arXiv preprint arXiv:2308.13137.
[160] S.A. Jacobs, M. Tanaka, C. Zhang, M. Zhang, L. Song, S. Rajbhandari, Y. He, Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models, 2023, arXiv preprint arXiv:2309.14509.
[161] L. Yue, Q. Liu, Y. Du, W. Gao, Y. Liu, F. Yao, Fedjudge: Federated legal large language model, 2023, arXiv preprint arXiv:2309.08173.
[162] H. Woisetschläger, A. Isenko, S. Wang, R. Mayer, H.-A. Jacobsen, Federated fine-tuning of llms on the very edge: The good, the bad, the ugly, 2023, arXiv preprint arXiv:2310.03150.
[163] T. Dettmers, R. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, D. Alistarh, SpQR: A sparse-quantized representation for near-lossless LLM weight compression, 2023, arXiv preprint arXiv:2306.03078.
[164] M. Choi, M.A. Asif, J. Willes, D. Emerson, FlexModel: A framework for interpretability of distributed large language models, 2023, arXiv preprint arXiv:2312.03140.
[165] F. Yang, S. Peng, N. Sun, F. Wang, K. Tan, F. Wu, J. Qiu, A. Pan, Holmes: Towards distributed training across clusters with heterogeneous NIC environment, 2023, arXiv preprint arXiv:2312.03549.
[166] T. Fan, Y. Kang, G. Ma, W. Chen, W. Wei, L. Fan, Q. Yang, Fate-llm: A industrial grade federated learning framework for large language models, 2023, arXiv preprint arXiv:2310.10049.
[167] M. Cho, K.A. Vahid, Q. Fu, S. Adya, C.C. Del Mundo, M. Rastegari, D. Naik, P. Zatloukal, eDKM: An efficient and accurate train-time weight clustering for large language models, IEEE Comput. Architect. Lett. (2024).
[168] S. Han, B. Buyukates, Z. Hu, H. Jin, W. Jin, L. Sun, X. Wang, C. Xie, K. Zhang, Q. Zhang, et al., FedMLSecurity: A benchmark for attacks and defenses in federated learning and LLMs, 2023, arXiv preprint arXiv:2306.04959.
[169] M. Xu, Y. Wu, D. Cai, X. Li, S. Wang, Federated fine-tuning of billion-sized language models across mobile devices, 2023, arXiv preprint arXiv:2308.13894.
[170] J. Yuan, C. Yang, D. Cai, S. Wang, X. Yuan, Z. Zhang, X. Li, D. Zhang, H. Mei, X. Jia, et al., Rethinking mobile AI ecosystem in the LLM era, 2023, arXiv preprint arXiv:2308.14363.
[171] A. Douillard, Q. Feng, A.A. Rusu, R. Chhaparia, Y. Donchev, A. Kuncoro, M. Ranzato, A. Szlam, J. Shen, DiLoCo: Distributed low-communication training of language models, 2023, arXiv preprint arXiv:2311.08105.
[172] J. Zhang, S. Vahidian, M. Kuo, C. Li, R. Zhang, G. Wang, Y. Chen, Towards building the federated GPT: Federated instruction tuning, 2023, arXiv preprint arXiv:2305.05644.
[173] H. Peng, K. Wu, Y. Wei, G. Zhao, Y. Yang, Z. Liu, Y. Xiong, Z. Yang, B. Ni, J. Hu, et al., Fp8-lm: Training fp8 large language models, 2023, arXiv preprint arXiv:2310.18313.
[174] D. Xu, W. Yin, X. Jin, Y. Zhang, S. Wei, M. Xu, X. Liu, LLMCad: Fast and scalable on-device large language model inference, 2023, arXiv preprint arXiv:2309.04255.
[175] R. Yi, L. Guo, S. Wei, A. Zhou, S. Wang, M. Xu, Edgemoe: Fast on-device inference of moe-based large language models, 2023, arXiv preprint arXiv:2308.14352.
[176] L. Zhu, L. Hu, J. Lin, W.-M. Chen, W.-C. Wang, C. Gan, S. Han, PockEngine: Sparse and efficient fine-tuning in a pocket, in: Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture, 2023, pp. 1381–1394.
[177] Y. Chen, Y. Yan, Q. Yang, Y. Shu, S. He, J. Chen, Confidant: Customizing transformer-based LLMs via collaborative edge training, 2023, arXiv preprint arXiv:2311.13381.
[178] Y. Zhu, Y. Liu, F. Stahlberg, S. Kumar, Y.-h. Chen, L. Luo, L. Shu, R. Liu, J. Chen, L. Meng, Towards an on-device agent for text rewriting, 2023, arXiv preprint arXiv:2308.11807.
[179] J. Zhao, Y. Song, S. Liu, I.G. Harris, S.A. Jyothi, LinguaLinked: A distributed large language model inference system for mobile devices, 2023, arXiv preprint arXiv:2312.00388.
[180] V. Jaganathan, D. Gouda, K. Arora, M. Aggarwal, C. Zhang, On-device video analysis with LLMs, in: Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications, 2024, pp. 153–153.
[181] S. Carreira, T. Marques, J. Ribeiro, C. Grilo, Revolutionizing mobile interaction: Enabling a 3 billion parameter gpt LLM on mobile, 2023, arXiv preprint arXiv:2310.01434.
[182] D. Peng, Z. Fu, J. Wang, Pocketllm: Enabling on-device fine-tuning for personalized llms, 2024, arXiv preprint arXiv:2407.01031.
[183] J. Bang, J. Lee, K. Shim, S. Yang, S. Chang, Crayon: Customized on-device LLM via instant adapter blending and edge-server hybrid inference, 2024, arXiv preprint arXiv:2406.07007.
[184] Y. Chen, R. Li, Z. Zhao, C. Peng, J. Wu, E. Hossain, H. Zhang, Netgpt: A native-ai network architecture beyond provisioning personalized generative services, 2023, arXiv preprint arXiv:2307.06148.
[185] N. Dhar, B. Deng, D. Lo, X. Wu, L. Zhao, K. Suo, An empirical analysis and resource footprint study of deploying large language models on edge devices, in: Proceedings of the 2024 ACM Southeast Conference, 2024, pp. 69–76.
[186] P. Choi, J. Kim, J. Kwak, Impact of joint heat and memory constraints of mobile device in edge-assisted on-device artificial intelligence, in: Proceedings of the 2nd International Workshop on Networked AI Systems, 2024, pp. 31–36.
[187] Y. Ding, C. Niu, F. Wu, S. Tang, C. Lyu, G. Chen, Enhancing on-device LLM inference with historical cloud-based llm interactions, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 597–608.
[188] Llama.cpp, 2023, URL https://github.com/ggerganov/llama.cpp.
[189] G. Gerganov, Llama. cpp: Port of facebook's llama model in c/c++, 2023, URL https://github.com/mlc-ai/mlc-llm.
[190] X. Shen, P. Dong, L. Lu, Z. Kong, Z. Li, M. Lin, C. Wu, Y. Wang, Agile-quant: Activation-guided quantization for faster inference of LLMs on the edge, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 18944–18951.
[191] Z. Li, Z. Hou, H. Liu, T. Li, C. Yang, Y. Wang, C. Shi, L. Xie, W. Zhang, L. Xu, et al., Federated learning in large model era: Vision-language model for smart city safety operation management, in: Companion Proceedings of the ACM on Web Conference 2024, 2024, pp. 1578–1585.
[192] Y. Rong, Y. Mao, H. Cui, X. He, M. Chen, Edge computing enabled large-scale traffic flow prediction with GPT in intelligent autonomous transport system for 6G network, IEEE Trans. Intell. Transp. Syst. (2024).
[193] N. Su, C. Hu, B. Li, B. Li, TITANIC: Towards production federated learning with large language models, in: IEEE INFOCOM, 2024.
[194] C. Liu, J. Zhao, Resource allocation in large language model integrated 6G vehicular networks, 2024, arXiv preprint arXiv:2403.19016.
[195] S. Paul, L. Zhang, Y. Shen, H. Jin, Enabling device control planning capabilities of small language model, in: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2024, pp. 12066–12070.
[196] U. Thakker, P.N. Whatmough, Z.-G. Liu, M. Mattina, J. Beu, Compressing language models using doped kronecker products, 2020, arXiv preprint arXiv:2001.08896.
[197] X. Wei, Y. Zhang, X. Zhang, R. Gong, S. Zhang, Q. Zhang, F. Yu, X. Liu, Outlier suppression: Pushing the limit of low-bit transformer language models, Adv. Neural Inf. Process. Syst. 35 (2022) 17402–17414.
[198] T. Choudhary, V. Mishra, A. Goswami, J. Sarangapani, A comprehensive survey on model compression and acceleration, Artif. Intell. Rev. 53 (2020) 5113–5155.
[199] J.H. Heo, J. Kim, B. Kwon, B. Kim, S.J. Kwon, D. Lee, Rethinking channel dimensions to isolate outliers for low-bit weight quantization of large language models, 2023, arXiv preprint arXiv:2309.15531.
[200] N.P. Pandey, M. Fournarakis, C. Patel, M. Nagel, Softmax bias correction for quantized generative models, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 1453–1458.
[201] W.X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., A survey of large language models, 2023, arXiv preprint arXiv:2303.18223.

[202] T. Dettmers, M. Lewis, Y. Belkada, L. Zettlemoyer, Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale, Adv. Neural Inf. Process. Syst. 35 (2022) 30318–30332.
[203] X. Wu, Z. Yao, Y. He, Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats, 2023, arXiv preprint arXiv:2307.09782.
[204] Y. Zhang, L. Zhao, S. Cao, W. Wang, T. Cao, F. Yang, M. Yang, S. Zhang, N. Xu, Integer or floating point? New outlooks for low-bit quantization on large language models, 2023, arXiv preprint arXiv:2305.12356.
[205] L. Nair, M. Bernadskiy, A. Madhavan, C. Chan, A. Basumallik, D. Bunandar, INT-FP-QSim: Mixed precision and formats for large language models and vision transformers, 2023, arXiv preprint arXiv:2307.03712.
[206] L. Deng, G. Li, S. Han, L. Shi, Y. Xie, Model compression and hardware acceleration for neural networks: A comprehensive survey, Proc. IEEE 108 (4) (2020) 485–532.
[207] S.N. Sridhar, A. Sarah, Undivided attention: Are intermediate layers necessary for bert? 2020, arXiv preprint arXiv:2012.11881.
[208] T. Gale, E. Elsen, S. Hooker, The state of sparsity in deep neural networks, 2019, arXiv preprint arXiv:1902.09574.
[209] L. Yin, S. Liu, M. Fang, T. Huang, V. Menkovski, M. Pechenizkiy, Lottery pools: Winning more by interpolating tickets without increasing training or inference cost, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 10945–10953.
[210] E. Frantar, D. Alistarh, Optimal brain compression: A framework for accurate post-training quantization and pruning, Adv. Neural Inf. Process. Syst. 35 (2022) 4475–4488.
[211] I. Hubara, B. Chmiel, M. Island, R. Banner, J. Naor, D. Soudry, Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks, Adv. Neural Inf. Process. Syst. 34 (2021) 21099–21111.
[212] H. Shao, B. Liu, Y. Qian, One-shot sensitivity-aware mixed sparsity pruning for large language models, 2023, arXiv preprint arXiv:2310.09499.
[213] E. Frantar, D. Alistarh, Massive language models can be accurately pruned in one-shot, 2023, arXiv preprint arXiv:2301.00774.
[214] X. Ma, G. Fang, X. Wang, LLM-pruner: On the structural pruning of large language models, 2023, arXiv preprint arXiv:2305.11627.
[215] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015, arXiv preprint arXiv:1503.02531.
[216] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, Q. Liu, Tinybert: Distilling bert for natural language understanding, 2019, arXiv preprint arXiv:1909.10351.
[217] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, D. Zhou, Mobilebert: a compact task-agnostic bert for resource-limited devices, 2020, arXiv preprint arXiv:2004.02984.
[218] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, Adv. Neural Inf. Process. Syst. 33 (2020) 5776–5788.
[219] H. Tsai, J. Riesa, M. Johnson, N. Arivazhagan, X. Li, A. Archer, Small and practical BERT models for sequence labeling, 2019, arXiv preprint arXiv:1909.00100.
[220] S. Sun, Y. Cheng, Z. Gan, J. Liu, Patient knowledge distillation for bert model compression, 2019, arXiv preprint arXiv:1908.09355.
[221] D. Chatterjee, Making neural machine reading comprehension faster, 2019, arXiv preprint arXiv:1904.00796.
[222] P. Kaliamoorthi, A. Siddhant, E. Li, M. Johnson, Distilling large language models into tiny and effective students using pQRNN, 2021, arXiv preprint arXiv:2101.08890.
[223] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, J. Lin, Distilling task-specific knowledge from bert into simple neural networks, 2019, arXiv preprint arXiv:1903.12136.
[224] C. Hu, X. Li, D. Liu, H. Wu, X. Chen, J. Wang, X. Liu, Teacher-student architecture for knowledge distillation: A survey, 2023, arXiv preprint arXiv:2308.04268.
[225] A. Hilmkil, S. Callh, M. Barbieri, L.R. Sütfeld, E.L. Zec, O. Mogren, Scaling federated learning for fine-tuning of large language models, in: International Conference on Applications of Natural Language To Information Systems, Springer, 2021, pp. 15–23.
[226] S. Wang, S. Zhuang, B. Koopman, G. Zuccon, ReSLLM: Large language models are strong resource selectors for federated search, 2024, arXiv preprint arXiv:2401.17645.
[227] T.J. Chua, W. Yu, J. Zhao, K.-Y. Lam, FedPEAT: Convergence of federated learning, parameter-efficient fine tuning, and emulator assisted tuning for artificial intelligence foundation models with mobile edge computing, 2023, arXiv preprint arXiv:2310.17491.
[228] B. Wang, Y.J. Zhang, Y. Cao, B. Li, H.B. McMahan, S. Oh, Z. Xu, M. Zaheer, Can public large language models help private cross-device federated learning? 2023, arXiv preprint arXiv:2305.12132.
[229] C. Hou, H. Zhan, A. Shrivastava, S. Wang, S. Livshits, G. Fanti, D. Lazar, Privately customizing prefinetuning to better match user data in federated learning, 2023, arXiv preprint arXiv:2302.09042.
[230] X.-Y. Liu, R. Zhu, D. Zha, J. Gao, S. Zhong, M. Qiu, Differentially private low-rank adaptation of large language model using federated learning, 2023, arXiv preprint arXiv:2312.17493.
[231] L. Peng, G. Luo, S. Zhou, J. Chen, Z. Xu, R. Zhang, J. Sun, An in-depth evaluation of federated learning on biomedical natural language processing, medRxiv (2023) 2023-2011.
[232] L. Yunxiang, L. Zihan, Z. Kai, D. Ruilong, Z. You, Chatdoctor: A medical chat model fine-tuned on llama model using medical domain knowledge, 2023, arXiv preprint arXiv:2303.14070.
[233] W. Dong, X. Wu, J. Li, S. Wu, C. Bian, D. Xiong, FewFedWeight: Few-shot federated learning framework across multiple nlp tasks, 2022, arXiv preprint arXiv:2212.08354.
[234] Z. Qin, D. Chen, B. Qian, B. Ding, Y. Li, S. Deng, Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes, 2023, arXiv preprint arXiv:2312.06353.
[235] B. Ouyang, S. Ye, L. Zeng, T. Qian, J. Li, X. Chen, Pluto and charon: A time and memory efficient collaborative edge ai framework for personal LLMs fine-tuning, in: Proceedings of the 53rd International Conference on Parallel Processing, 2024, pp. 762–771.
[236] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, et al., Parameter-efficient fine-tuning of large-scale pre-trained language models, Nat. Mach. Intell. 5 (3) (2023) 220–235.
[237] G. Pu, A. Jain, J. Yin, R. Kaplan, Empirical analysis of the strengths and weaknesses of PEFT techniques for LLMs, 2023, arXiv preprint arXiv:2304.14999.
[238] L. Qian, J. Zhao, User association and resource allocation in large language model based mobile edge computing system over 6G wireless communications, in: 2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring), IEEE, 2024, pp. 1–7.
[239] G. Qu, Z. Lin, F. Liu, X. Chen, K. Huang, TrimCaching: Parameter-sharing AI model caching in wireless edge networks, 2024, arXiv preprint arXiv:2405.03990.
[240] M. Xu, D. Cai, Y. Wu, X. Li, S. Wang, Fwdllm: Efficient federated finetuning of large language models with perturbed inferences, in: USENIX ATC, 2024.
[241] G. Kim, J. Yoo, S. Kang, Efficient federated learning with pre-trained large language model using several adapter mechanisms, Mathematics 11 (21) (2023) 4479.
[242] M. Benington, L. Phan, C.P. Paul, E. Shoemaker, P. Ranade, T. Collett, G.H. Perez, C. Krieger, Scaling studies for efficient parameter search and parallelism for large language model pre-training, 2023, arXiv preprint arXiv:2310.05350.
[243] Y. Ghannane, M.S. Abdelfattah, Diviml: A module-based heuristic for mapping neural networks onto heterogeneous platforms, in: 2023 IEEE/ACM International Conference on Computer Aided Design, ICCAD, IEEE, 2023, pp. 01–09.
[244] S. Ohta, T. Nishio, Λ-Split: A privacy-preserving split computing framework for cloud-powered generative AI, 2023, arXiv preprint arXiv:2310.14651.
[245] W. Huang, Y. Wang, A. Cheng, A. Zhou, C. Yu, L. Wang, A fast, performant, secure distributed training framework for large language model, 2024, arXiv preprint arXiv:2401.09796.
[246] J. Zhao, Y. Song, I. Harris, S.A. Jyothi, et al., LinguaLinked: Distributed large language model inference on mobile devices, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), 2024, pp. 160–171.
[247] X. Zhou, Q. Jia, Y. Hu, R. Xie, T. Huang, F.R. Yu, Geng: An LLM-based generic time series data generation approach for edge intelligence via cross-domain collaboration, in: IEEE INFOCOM 2024-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2024, pp. 1–6.
[248] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, et al., Efficient large-scale language model training on gpu clusters using megatron-lm, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2021, pp. 1–15.
[249] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, B. Catanzaro, Megatron-lm: Training multi-billion parameter language models using model parallelism, 2019, arXiv preprint arXiv:1909.08053.
[250] J. Rasley, S. Rajbhandari, O. Ruwase, Y. He, Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3505–3506.
[251] D.P. Pau, F.M. Aymone, Forward learning of large language models by consumer devices, Electronics 13 (2) (2024) 402.
[252] S. Wang, J. Wei, A. Sabne, A. Davis, B. Ilbeyi, B. Hechtman, D. Chen, K.S. Murthy, M. Maggioni, Q. Zhang, et al., Overlap communication with dependent computation via decomposition in large deep learning models, in: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Vol. 1, 2022, pp. 93–106.
[253] H. Fan, S.I. Venieris, A. Kouris, N. Lane, Sparse-dysta: Sparsity-aware dynamic and static scheduling for sparse multi-DNN workloads, in: Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture, 2023, pp. 353–366.

[254] T. Che, J. Liu, Y. Zhou, J. Ren, J. Zhou, V.S. Sheng, H. Dai, D. Dou, Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization, 2023, arXiv preprint arXiv:2310.15080.
[255] T. Tambe, J. Zhang, C. Hooper, T. Jia, P.N. Whatmough, J. Zuckerman, M.C. Dos Santos, E.J. Loscalzo, D. Giri, K. Shepard, et al., 22.9 a 12nm 18.1 TFLOPs/w sparse transformer processor with entropy-based early exit, mixed-precision predication and fine-grained power management, in: 2023 IEEE International Solid-State Circuits Conference, ISSCC, IEEE, 2023, pp. 342–344.
[256] L. Collins, S. Wu, S. Oh, K.C. Sim, Profit: Benchmarking personalization and robustness trade-off in federated prompt tuning, 2023, arXiv preprint arXiv:2310.04627.
[257] J. Sun, Z. Xu, H. Yin, D. Yang, D. Xu, Y. Chen, H.R. Roth, FedBPT: Efficient federated black-box prompt tuning for large language models, 2023, arXiv preprint arXiv:2310.01467.
[258] J. Jiang, X. Liu, C. Fan, Low-parameter federated learning with large language models, 2023, arXiv preprint arXiv:2307.13896.
[259] X. Wu, Z. Yao, Y.H. Zeroquant-fp, A leap forward in llms post-training w4a8 quantization using floating-point formats, 2023, arXiv preprint arXiv:2307.09782.
[260] Q. Xu, Y. You, An efficient 2d method for training super-large deep learning models, in: 2023 IEEE International Parallel and Distributed Processing Symposium, IPDPS, IEEE, 2023, pp. 222–232.