AMERICAN ACADEMIC PUBLISHER

INTERNATIONAL JOURNAL OF DATA SCIENCE AND MACHINE LEARNING (ISSN: 2692-5141)

Volume 05, Issue 01, 2025, pages 300-322
Published Date: - 19-06-2025
Doi: -https://doi.org/10.55640/ijdsml-05-01-25

A Review of Large Language Models in Edge Computing: Applications, Challenges, Benefits, and Deployment Strategies

Venkata Srinivas Kompally
Northeastern University, Boston, MA,

ABSTRACT

Large Language Models (LLMs) have achieved very good success in natural language processing, but deployment of these powerful models on edge computing devices across all domains presents unique challenges. This paper reviews the state of LLMs in edge computing, focusing on four key aspects: their emerging applications across various sectors, the technical challenges of running LLMs on resource-constrained edge devices, the potential benefits of bringing LLM capabilities closer to data sources, and effective deployment strategies to enable LLMs at the edge. We also discuss on how LLM edge deployment could offer low-latency, privacy-preserving intelligent assistance throughout a range of domains, such as healthcare, IoT, industrial automation, and more. We also look at some techniques and architectures that can overcome the limitations of edge devices, such as cloud-edge collaboration, federated learning, model compression, and on-device inference. This review identifies practical ways to integrate LLMs into edge environments by examining current practices and their trade-offs. It also provides guidance for future research to address the remaining issues in this quickly expanding field.

KEYWORDS

Edge computing, large language models (LLMs), Edge AI, On-Device Inference, IoT, Federated learning, Model compression, Deployment Strategies

1. INTRODUCTION

Large Language Models (LLMs) like the GPT models and other transformer-based models have become a revolution in natural language processing because of their ability to interpret and generate text like humans. These models can achieve state-of-the-art performance on tasks such as language translation and summarization to answering a question and code generation. However, most LLM deployments now depend on powerful cloud servers or data center infrastructure due to the models' powerful computational and memory requirements. On the other hand, edge computing refers to computing that is performed near the data source or end-user, for example, on local devices like smartphones, IoT sensors, or edge servers, instead of centralized cloud data centers. This paper tackles the research issue of how to efficiently bring the capabilities of large language models into edge computing environments with limited resources and possibly uneven network connectivity.

Running LLMs on the network edge offers several advantages: it can enhance privacy by keeping sensitive data on-device, reduce latency by processing data locally in real-time, and allow continuous service even in areas with spotty internet. An advanced AI assistant on a smart device, for instance, could comprehend voice commands and produce immediate replies without having to reach a cloud server. Factory floor machines or autonomous robots could run language models on local controllers to understand complicated instructions on-site or make quick, split-second decisions. These examples are highly desirable for applications that require quick or reliable responses, such as healthcare diagnostics in remote clinics, autonomous vehicles navigating in real time, or military and disaster-response operations where connectivity may be limited. Enabling LLMs on edge devices is therefore important since it will help to open new intelligent possibilities in fields where timing, privacy, and dependability are top priorities.

Although it is promising, achieving a very effective LLM deployment on edge devices is a big challenge. Edge devices are inherently resource-constrained, and they have much less processing power, memory, and energy capacity than cloud servers. An LLM which can run smoothly on a cloud GPU cluster might be impossible to load into a small device's memory. Furthermore, edge environments are heterogeneous: one finds a wide variety of hardware (from smartphones and edge gateways to tiny microcontrollers) and software stacks, which makes it difficult to develop one-size-fits-all solutions. This paper aims to address these issues by reviewing and synthesizing the current research and developments on LLMs in edge computing. The goals of the paper are to first survey application domains where edge-based LLMs are being applied or could be beneficial, showing the practical use-cases driving this field; and then to discuss the technical challenges and limitations that arise when deploying LLMs on edge hardware; and finally to examine the benefits and strategies for overcoming these challenges, including methodologies like model compression, federated learning, and distributed inference. Meeting these goals helps the review to provide a thorough knowledge of how LLMs can be pushed beyond the cloud and into edge devices, why this is important, and what tools are available to do so.

In summary, this paper's contributions are a detailed review of the emerging area of LLM deployment at the edge, highlighting key applications, challenges, benefits, and deployment strategies. The discussion is intended to inform researchers and practitioners of the current state-of-the-art and to identify areas where further innovation is needed. The practical applicability of this topic is broad – success in this area could enable smarter personal devices, more autonomous industrial systems, and generally a new class of AI solutions that operate closer to where data is generated.

2. METHODOLOGY

This review has been done using a structured methodology to ensure an exhaustive and unbiased coverage of relevant literature. First, we identified the scope of the topic, this paper is an intersection of large language models and edge computing. We focused on works that particularly address running or training LLMs in edge or resource-constrained environments. A search was performed in major scholarly databases (such as IEEE Xplore, ACM Digital Library, arXiv, and Google Scholar) using keywords including "large language models on edge devices", "edge AI language models", "on-device NLP models", "LLM deployment strategies edge computing", and similar terms. We primarily considered publications from the last several years (approximately 2018–2025), as this is a fast-evolving field with most relevant advancements occurring recently.

Second, To choose a relevant set of literature, we used inclusion and exclusion criteria. We included research papers, survey articles, and significant industry whitepapers that discussed any of the following: reducing LLM model size or complexity for edge deployment, case studies of LLM applications on edge devices, challenges encountered when moving NLP models to the edge, and proposed solutions or frameworks for edge-based inference or training. We excluded works that dealt with edge computing or LLMs in isolation without addressing their intersection (for example, general edge computing articles with no focus on AI models, or general LLM papers that assume standard cloud-based deployment). This yielded a body of literature consisting of academic papers from conferences and journals in computer science (machine learning, distributed computing, IoT) as well as some relevant technical reports and blog articles from the industry that shed light on cutting-edge developments.

Third, we performed a thematic study of the collected literature. The themes were derived both from the research questions of this review and from common topics emerging in the papers. We found that the literature naturally clustered around a few key themes: (1) Applications of LLMs at the edge (i.e., what are LLMs used for in edge scenarios, and in which sectors), (2) Challenges - the limitations and issues faced when bringing LLMs to the edge, (3) Benefits and motivations for edge deployment of LLMs - essentially why one would want to deploy locally despite the difficulties, and (4) Deployment strategies and solutions - methods proposed to make edge LLM deployment feasible (spanning techniques in model optimization, distributed computing, etc.). We organized the structure of this paper around these themes. Each theme was carefully analyzed: we compared findings from different sources, identified consensus as well as conflicting approaches, and distilled example cases to illustrate key points. Where quantitative data (e.g., model sizes, latency measurements, accuracy impacts of compression) were available in the literature, we noted these to inform the discussion, though this paper does not present new experimental results of its own.

Finally, as part of the methodology, we tried to maintain an objective tone, presenting the strengths and limitations of various approaches as described by their authors. The result is a synthesized review that aims to capture the current state-of-the-art in deploying large language models on edge computing infrastructure, organized in a manner that addresses the key aspects of applications, challenges, benefits, and deployment strategies.

3. Applications of LLMs in Edge Computing

LLMs have started to integrate with edge computing devices in a lot of industry sectors. By implementing sophisticated language understanding and generation capabilities nearer to the source of the data, organizations are now able to unlock new use cases that will benefit from closeness and context awareness. This section highlights some of the domains where LLMs at the edge can be applied. Specific examples in each domain show how on-device models are used and why these use cases prefer edge deployment over the traditional cloud approach

A. Healthcare

In the healthcare industry, deploying LLMs on edge computing devices(such as hospital workstations or emergency response devices) will enable sensitive medical information to be processed in local device for real-time decision support. For instance, a medical diagnostic assistant LLM can stay on the hospital's local server, which is secure, or even hand handheld device used by the medical personnel. This model can help summarize the patient’s record, or even suggest possible diagnoses from the symptoms, and all of this can be performed without sending confidential data to the external cloud servers. The result is quick, and it also preserves the privacy of the patient. Another upcoming application is in the medical imaging devices: an LLM-based system can understand radiology reports or guide an MRI machine’s operations based on the size. Processing such data at the edge will reduce the delay and the potential downtime that would occur if the large image data were uploaded to the cloud for analysis. In remote and rural healthcare places with limited connectivity, edge AI can be a life-saving option.

B. Consumer IoT and Smart Homes

A large number of consumer gadgets, like smartphones and smart home assistants, have started to incorporate on-device intelligence. LLMs on personal gadgets can enable smarter and more responsive user interaction without depending on continuous cloud access. A good example would be a voice-controlled virtual assistant like Alexa, Siri, or Google Assistant sends voice data to the cloud for interpretation by the models. If the LLMs are capable enough to run on the home hub or the smartphone, then the device could understand and respond to the complicated voice commands locally. This will reduce latency and will keep the personal audio data private within the home. These applications benefit from edge deployment because they require immediate response and will operate in contexts where privacy is generally required.

C. Industrial and Manufacturing

Industrial environments are referred to as Industrial IoT (IIoT) and can use LLMs at the edge for improving automation and monitoring. Factories are increasingly connected with a lot of sensors and are generating streams of data continuously. While several sensor data is common, LLMs can be used to understand the unstructured text-based data or interact through natural language with the operators who are humans. An LLM deployed on the edge, for instance can ingest maintenance logs or error reports generated by machines and translate them into actionable insights or language that can be understood by the technicians. It could also serve as voice voice-controlled system for complex machinery, which allows engineers on the floor to query machines in natural language with the LLM processing these requests on the cloud server in the factory. Because manufacturing often has to deal with real-time control and decisions regarding safety, keeping the AI inference locally avoids the cloud latency. Moreover, most of the industries have isolated or secure networks, so on-premise LLM is preferable to send data off-site. To summarize, edge LLMs in factories can improve human-machine interaction on the production floor while meeting the strict reliability and security requirements.

D. Autonomous Vehicles and Transportation

Vehicles like cars, autonomous robots, and drones are edge devices with a lot of sensors. LLMs can play a significant role in these systems by handling decision-making and communication tasks. For instance, in an autonomous car, a tight language model can be used to interpret the voice commands from the human driver or summarize and explain the alerts from the navigation system. In the future, these vehicles can have onboard systems that use LLMs to analyze open-ended queries or even negotiate traffic instructions with other vehicles or roadside units in natural language formats. Because these vehicles must often perform in environments that have limited or no internet connectivity and have to make instant life-saving decisions, hosting AI models like LLMs on the vehicle's onboard computer is important. Drones and delivery robots can also similarly use on-device LLMs to understand high-level mission instructions or communicate their information in human human-readable format to enable better autonomy. In transportation infrastructure, roadside edge servers can deploy LLMs to summarize traffic reports for emergency management and provide local law enforcement with quick insights without relying entirely on central cloud analytics.

From personal devices to vital infrastructure, these instances show that edge-deployed LLMs cover a broad spectrum of uses. Common to all these use cases is the need for immediacy, data locality, and dependability. Table 1 gives us a summary of several major sectors and some sample LLM application scenarios at the edge, which highlight the motivations for local deployment in each case.

Table 1: Edge computing applications of LLMs by sector

Sector	**Example Edge LLM Application**	**Why on-Edge(Key Benefits)**
Healthcare	*On-site clinical assistant* - It is an LLM on a hospital's server that can summarize all patient records and suggest diagnoses in real time.	Immediate critical feedback, Data privacy, and can work in low-connectivity environments.
Consumer IoT/Smart Home	*Voice assistant hub* - Smart home devices like Alexa can run an LLM to interpret and respond to complicated voice commands	Low latency, responsive user experience, and the personal data never leaves the device
Industrial/Manufacturing	*Edge analytics on IIoT* - An industrial gateway with an LLM can interpret logs generated from machine and sensor descriptions to alert technicians of the issues.	Real-time operations can be done with no cloud delay and will keep the data on-premises
Automotive/Transportation	*Autonomous drone controller* - A drone that carries a lightweight LLM can understand high-level commands and can report the status in natural language	Fast, split-second decisions can be made to improve user trust and provide explanations locally

4. Challenges of Deploying LLMs on Edge Devices

Moving LLMs from powerful cloud servers to small edge devices has a lot of challenges. In this section, we will discuss the key technical and practical challenges that should be addressed when deploying LLMs in edge computing environments. These range from limitations to software and system-level issues.

A. Resource Constraints(Compute and Memory):

The most important challenge is that edge devices generally have constrained computations and memory available compared to the cloud infrastructure. LLMs like GPT have hundreds of billions of parameters and require gigabytes of VRAM and powerful GPUs to run efficiently. An edge device has a small fraction of the resources compared to the cloud. Running inference for LLM is computationally intensive: the self-attention mechanism and matrix multiplications that these models perform can overwhelm a low-power processor, resulting in latency or failure to run. Similarly, the memory needed to load the complete model and its weights can far exceed what is present on a device. In a lot of cases, general deployment is not possible simply because the model won't fit in memory, or the inference can take many seconds or minutes per query on edge hardware. This resource gap is a big obstacle requiring innovative solutions.

B. Hardware Heterogeneity:

Edge computing encompasses a wide variety of hardware platforms - from handheld phones and tablets to single-board computers like Raspberry Pi, to IoT modules and custom embedded systems. Each of these might have different CPU architectures, different support for accelerators, and different performance characteristics. There is no single ”edge device” profile, which can mean an LLM that is optimized for on-device might not run efficiently or at all on another device. Software frameworks also differ, which complicates the deployment process. The heterogeneity makes it challenging to develop and optimize the LLM deployments, which are portable across devices. It often necessitates device-specific training, optimization, or even re-training of models that match the constraints of the hardware.

C. Network Connectivity and Bandwidth

One reason to deploy LLMs at the edge is to reduce dependency on networks, but in practice, many edge deployments will still involve some communication with cloud and edge servers. A few strategies that involve splitting the model's workload between the device and cloud, or periodically updating a model that is on-device with knowledge from the cloud. In most of the cases like this, limited network bandwidth and connectivity which is unreliable connectivity become challenging. If an edge device downloads a large model or even just sends intermediate data to a server, network latency can bottleneck the system. For example, moving a multi-gigabyte model file over a slow connection can take an impractical long time. Interference that relies on cloud cooperation might stall if the connection is either weak or lost. Edge scenarios often involve many distributed devices, so collectively, the communication overhead can be substantial. Being efficient in managing network usage with techniques like model caching, update compression, or scheduling communication during nonpeak times will become important to ensure that the benefits of edge deployment are not negated by network delays or congestion.

D. Privacy and Security Concerns

Even though one of the main motivations for edge deployments is to improve privacy, it also introduces its own challenges. Edge devices may be vulnerable to physical tampering or theft, which can potentially expose the model data. Ensuring data privacy on a device not only means avoiding sending data to the cloud but also securing the data at rest and while it is in use of the device. If an LLM can process sensitive information on the edge device, we have to consider how we can protect the device from leaks. For example, using encryption for stored model files or securing the enclaves for computation. Additionally, distributing a valuable LLM to potentially hundreds of edges will raise some concerns about intellectual property rights and the security of models. The model itself could be extracted or replicated if the device is compromised. Another security aspect is maintaining trust, the model's output edge devices might not have the same degree of oversight and logging as the cloud environments. When strong mechanisms for authentication, authorisation, secure and timely updates for the model, and even adversarial robustness are needed when LLMs are operated outside a secure data center.

E. Concurrent Execution and Real-Time Constraints:

Edge devices most of the time perform multiple tasks and are sometimes involved in real-time control loops. LLMs that are integrated in such an environment must be done carefully to avoid degrading the performance of important processes. For example, an autonomous robot might need some time to run LLM to make high-level decision-making, but it also has low-level control algorithms running with strict timing requirements. If the LLM decision takes too long, it can delay the critical tasks and cause system failures. Therefore, scheduling and management of resources is a major challenge. The system has to allocate enough resources to the LLM to function well, but not so much that other processes will starve. In industrial and automotive contexts, the can be an issue of safety. Techniques like priority-based scheduling, real-time operating system features, or running LLM on another process are required to mitigate this. Also, if a lot of users or processes are trying to use LLM on a single device, concurrency control and performance can be an issue in maintaining good quality of service.

To summarize, deploying LLMs on the edge devices exposes a lot of challenges that are less in traditional cloud deployments. To overcome these issues, making models smaller and faster, adapting to diverse hardware, managing network limitations, ensuring security, and balancing resources in real time is essential for the success of edge-based LLM applications.

5. Benefits of Edge Deployment of LLMs

Even after the challenges discussed above, there are a lot of benefits to deploying the LLMs in deep environments. These benefits are the driving factors behind the push for edge-based LLMs, and they are clearly aligned with the motivations we highlighted in the introduction. There, we discuss in detail the main advantages and positive outcomes expected from running .LLMs on edge devices with a few examples.

A. Low Latency and Real-Time Responsiveness

By processing the data locally on the device or near the source of the data, the deployment of edge can drastically reduce the latency for the LLM-based applications. Data does not need to travel over the internet to any remote server for processing, instead, the computation can happen on-site and will give us the immediate result. This is very important for real-time applications. For example, a smart home assistant like Alexa uses on-device LLM, which responds to the user's request in a fraction of a second, whereas a cloud-based solution might take several seconds to make a round trip, especially when the network is slow. In scenarios like self-driving vehicles or emergency response, even a slight latency reduction can make a difference in safety. Edge LLMs ensure that decision-making is very close to the source and will enable instant feedback and control. The real-time capability not only improves the performance but also enhances the user experience.

B. Offline Operation and Reliability

Running LLMs on edge devices will allow them to run without an internet connection. This offline capability is a very good benefit for use cases in remote locations, during Network outages, or even when the connectivity is very expensive or intermittent. Applications like translation or information retrieval can work on a smartphone even when the user is on a flight or in a remote Village where there is no signal. For critical systems like medical devices, there will be an ability to operate independently, and they will become more resilient to network failures. This will enhance reliability, the service will not simply go down because there is no internet. In addition, reducing the dependency on cloud can lower operational costs associated with the data transmission and also simplify compliance which scenarios where the devices operate in a highly secure network. Overall, Edge-deployed LLMs can improve the robustness of the AI services and not be limited because of intermittent Network availability

C. Privacy Preservation

One of the most important benefits of computing is data privacy and security. By keeping the data in the local network, there is no need for the sensitive information to be transmitted to the cloud, where it might be at risk of interception. For LLMs, which often process sensitive data, this is very important. On-device LLM can analyze users' emails, voice recordings documents and can provide assistance without ever leaving the device. This will reduce the risk of privacy breaches and can also help comply with the data protection regulations of that locality. Users are more likely to trust and adopt these LLM-based AI features when they know their data is processed locally. Also, because the model itself can be kept local, there is very little chance of external attacks that will try to steal the data via cloud APIs. In the edge deployed LLMs, the device can as as its own guardian of data, processing,g and then discarding the sensitive information internally. This will benefit Industries like healthcare and finance, where storing data locally is important because they are very sensitive.

D. Bandwidth Savings and Lower Cloud Dependence:

Running processes on the edge device can significantly reduce network bandwidth usage. Instead of streaming large amounts of data to the cloud for processing, the device can handle a lot of the data locally and only send minimal necessary information over the internet. This is very beneficial as applications scale up or incorporate high-volume data like video feeds. For example, consider a fleet of surveillance cameras that have built-in LLM-based Analytics If they can interpret and summarize the footage locally, then they don't need to constantly upload high bandwidth video streams unless an important event occurs. This will reduce network conditions and the costs that are associated with data transfer and Cloud computing. For organizations that deploy thousands of edge-enabled devices, the cloud computing bills can be significantly lowered by shifting more work today. Moreover, it reduces pressure on cloud data centers and contributes to a more scalable system overall. The addition of more devices doesn't linearly increase the cloud load if each of the devices processes the data by itself. In short, the edge LLMs will help optimize resource utilization by computing locally and also reserving Cloud usage only when it is required for centralized processing.

E. Personalization and Context Awareness

Edge deployment will allow LLM to learn and adapt from the local data and the user behavior in a way that would be harder for it to achieve in a cloud setting. Because the model resides on the device locally, it can continuously update and fine-tune itself depending on the data privately. This will lead to a more personalized experience. For example, a personal assistant on a phone call over time learn the user's writing style, terminology, and preferences, which provides more tailored responses and recommendations. Since these updates happen on the device, they don't expose personal and sensitive information and the response can be immediate. Similarly, an LLM in smart building can be conceptually aware of local conditions and thus provide more relevant output than a generic cloud model that lacks detailed context. This contextual adaptation is a very big Advantage for applications requiring local relevance. Additionally, personalization extends to language or dialect preferences, and LLM on the device can adapt to the specific dialect or slang of a primary user or a community. In summary, edge-based LLMs that can become specialized experts of the environment or user, something which is very difficult to scale in the Cloud, where a single model will serve a large number of users.

In combination, all the above benefits will make a strong case for pushing AI inference to the edge whenever possible. This desire for fast, reliable, private, and personalized AI Services is aligned with deploying LLMs on local devices. It is important to note that getting these benefits depends on overcoming the challenges discussed earlier. That is where strategic deployment techniques and optimizations come into play, as we discuss next.

6. Deployment Strategies for Edge-Based LLMs

There are a lot of challenges in edge deployments, and the incentives to do it, researchers and engineers have been developing a variety of strategies to make LLMs run on edge devices. These deployment strategies are essentially techniques that can bridge the gap between the large resource requirements of LLMs and the limited capabilities of the edge hardware. These Strategies can be categorized into those that make the model itself smaller or more efficient, those that can distribute the workload in clever ways between the edge and other resources, and those that can change how the model learns or is updated to fit the edge device. In this section, we discuss major strategies along with a few examples.

6.1. Personalization and Context Awareness

Reducing the model size and the resource demand is one of the most direct ways to fit LLM onto an edge device. This can be done using compression and optimization techniques. Model compression to create a smaller version of an original model that will still achieve acceptable performance on tasks. A few approaches under this strategy are:

A. Quantization

This technique will reduce the numerical precision of the model's parameters and computations. For example, instead of using 32-bit floating-point numbers to represent model weights, a quantized model can use 8-bit integers. This will drastically reduce the size of the model and also speed up inference by using integer automatic, which is often faster on many processors. Quantization can introduce some loss in the accuracy of the model because of the rougher representation of numbers. Many LLMs can be quantized to 8-bit or 4-bit with minimal impact to the quality of outputs, especially if it is combined with calibration or fine-tuning.

B. Pruning

In pruning, the parts that are unnecessary in the network are removed to make it smaller and faster. For LLMs, pruning might involve removing redundant neurons, attention heads, or entire layers that contribute little to the model's output. Structured pruning can result in a smaller model that is still architecture-wise similar to the original one and can run easily on devices. Unstructured pruning can also reduce parameter count, even though it often requires specialized hardware to see runtime speed benefits. The challenge is to prune significantly without degrading the capability of the language model techniques exist to prune a model of gradually and then fine-tune it so that it can regain as much performance as possible in the cut-down form.

C. Knowledge Distillation

Distillation involves training "smaller" student models that imitate larger "teacher" models. Instead of directly using the original large LLM on the device, we can create a smaller model that has the capability of producing similar outputs by training it on examples. Primarily, the large model's knowledge is transferred into a small model, which is lightweight. The student model might not match up performance of the teacher, but if it is trained well, it can achieve a strong fraction of it while being far more feasible to deploy on the edge. For instance, a massive model with 100+ billion parameters might distill into a model with only a few hundred parameters, which is still large, but it will be manageable on an edge server or a high-end device with quantization. Distilled models often form the backbone of an on-device assistant because they capture much of the capability of the large LLM in a smaller package.

D. Efficient Architectures and Other Optimizations

Along with these techniques, researchers are trying to explore new model architectures that are more resource-efficient. For example, transformer-based models like GPT can be redesigned to use fewer layers or new attention mechanisms that are less computationally intensive. There are approaches like low rank adaptation(LoRA) and others that factorize the model's matrices, which reduce the number of parameters that need to be changed for a new task. Some optimizations target the runtime specifically, like operator fusion, optimizing scheduling of computation, or using memory swapping techniques that can handle models that are larger than the RAM by streaming parts in and out whenever needed. To summarize, this strategy is about squeezing the model using mathematical engineering means so that it becomes small efficient enough to meet edge constraints.

6.2. Federated Learning and On-Device Training

Another important strategy involves how the models are trained and updated in an edge environment. Federated learning is a paradigm that will allow multiple edge devices to collaboratively train or improve a model without sharing their raw data to a central cloud server. In the context of LLMS and edge computing, federated learning and related techniques can be very powerful.

A. Federated Learning(FL)

In a federated setup, each edge device keeps a local copy of the model, and then it trains on the local data it has. Instead of sending the data to the remote cloud server, each device will only send the model updates to a central coordinator. The coordinator will aggregate all the updates that it receives from the devices to improve the global model, which is then sent back down to the devices. This is one way using which an LLM can continuously be improved or personalized without exposing private data beyond the device. There are two benefits for edge LLM deployment here: it can be customized to edge use cases and preserve privacy. Apple's Siri is an example that uses versions of federated learning for language models to personalize predictions on-device.

B. On-Device Fine-tuning and Adaptation

It is related to federated learning, and it is based on the idea of performing on-device fine-tuning of language models. Instead of training a global model across devices, sometimes each device model needs to adapt for a specific environment. Lightweight fine-tuning techniques like training only certain layers, or using small learning rates on a subset of parameters can allow pre-trained LLMs to adjust to a specific domain or the user. For example, a company might deploy a base language model to all Edge notes in a network and then let each note fine-tune on local data so that it becomes an expert in that local area. Fine-tuning models that ran locally could even sometimes share their knowledge with each other or via a central mode. The important point here is making training feasible on edge hardware, which means optimizing the training process itself.

C. Federated Averaging and Privacy Techniques

A Challenge in the Federated setups for language models is privacy and communication. The model updates shared can be large. Techniques like Federated averaging reduce communication by sending only the aggregated updates, adding some trouble. Privacy can also be further enhanced using differential privacy or secure multiparty computation. While these are more methodological details, they can enable scalable and secure collaborative training of LLMs in edge scenarios.

Overall, Federated learning and on device training strategies are aimed to involve each devices actively in learning process, on the other hand the traditional approaches of training only in the cloud and then merely deploying a static model to devices but incorporating this strategies we can help overcome data privacy barrier and produce models that are better tuned for Edge use cases then generating Cloud trained language models

6.3. Distributed and Collaborative Inference

Even if you make models smaller and train them properly, an edge device could still have trouble handling an LLM's inference by itself, especially if the models are big or the reaction time needed has to be very fast. Distributed inference is one method that can divide the inference task into smaller pieces and send those pieces to different devices or the cloud:

A. Edge-Cloud Collaboration (Hybrid Deployment)

In a hybrid method, the edge device and a more powerful server share the effort of running the LLM. One easy way to do this is to break up the model's layers. For example, the first few layers of the neural network may operate on the edge device, processing the raw input. Then the intermediate representation could be transferred to the cloud, where the rest of the layers continue the processing. The device then gets the result back. This model partitioning can greatly reduce the strain on the device, but it will add some communication latency. If done correctly, the latency can still be low enough for interaction, especially with fast networks like 5G. The good thing is that the cloud (or edge server) does the hard work for the bits of the model that the device can't handle, and the device does enough to cut down on the amount of data sent. Dynamic offloading is a smart version of this. It lets the device determine on the fly whether to run locally or call out to the cloud based on things like network speed, how critical low latency is for the current query, and so on.

B. Distributed Across Multiple Edge Devices

When many edge devices work together to execute a model that none of them could run on their own, this is another type of distributed inference. This is a more experimental way to do things, and it's useful in situations like IoT sensor networks or fleets of vehicles. For example, a large LLM may be split up so that each device only has a part of the model's parameters. When a device gets an inference task, like a challenging question, it shares the query with other devices, and each one uses its own shard of the model to figure out part of the answer. Then, they all combine their answers. This needs very quick communication and synchronization, and it is a subject of active research since coordinating devices that may have different speeds and availability is hard. But if it works, it basically makes a distributed edge cluster that works like one large virtual model. A simpler option is to utilize an edge server (a very strong computer that is close to the devices, perhaps on the same local network) as a coordinator. Edge devices provide data to the local server, which executes an LLM and sends back results. This is like a small cloud in the edge area. It can give low latency because it is close by, but it can still offload devices. This is how a lot of industrial IoT systems work: sensors and actuators are light, but a gateway or server nearby runs the expensive AI models and feeds back choices or summaries.

C. Optimizing Communications and Pipelines:

Fast and efficient communication is necessary for distributed inference algorithms to work. Some methods utilized are compressing the intermediate data such that the part sent over the network is small, even if the model is split. There is also a study on pipeline parallelism at the edge. This means sending data across model partitions in a way that keeps all parts occupied, like an assembly line, to cut down on idle time. If one device is waiting for a response from the cloud, it could be able to start processing the next token or request at the same time, for example. The idea is to have any necessary communication have as little effect as possible. Some systems also include a backup plan: if the request is too complicated or needs greater accuracy, they will use the bigger model in the cloud. Otherwise, they will try to process the request on a device with a smaller model. In this manner, simple tasks can be done completely offline, while only the more difficult ones use up network resources. More and more people are using multi-tier architectures to find a compromise between performance and resource utilization.

In practice, the best deployment strategy usually uses a mix of the methods listed above. One way to do this is to utilize a compressed model on the device and let it offload to the cloud if it needs to. This is an example of integrating model compression with edge-cloud collaboration. You could also use federated learning to build a good edge model and then utilize quantization to deploy it for inference. There are many options available, and the best one for your application will rely on things like how many devices you have, how important latency is, how important accuracy is, and how well they link to each other. Table 2 compares some of the main tactics we've spoken about, showing their pros and cons next to each other.

Table 2:Comparison of Different Edge LLM Deployment Strategies

Strategy	**Description**	**Advantages**	**Limitations**
Model Compression	Make the model smaller and more efficient so it can run on the device. Some of the methods are lowering the accuracy of weights, getting rid of extra elements, and teaching smaller models to act like	- Greatly reduces memory and compute requirements, which will enable on-device inference.	- Some loss of accuracy or capability is common - Very large models may still be too big even after aggressive

	bigger ones.	- Often can maintain good enough accuracy for many tasks. - One-time optimization.	compression - Requires effort to compress and validate models
Federated Learning	Train or fine-tune models on a lot of devices without having to store all the data in one place. Every device sends local data to the model, and these changes are combined to make a better global model.	- Preserves data privacy (raw data never leaves devices). - Enables personalizatio n of models to local usage patterns. - Can continuously improve models in deployment.	- Communicatio n overhead for sharing model updates (can be heavy for large LLMs). - Devices need enough capability to do local training (which can be taxing). - Handling of stragglers and ensuring convergence can be complex.
Distributed Inference	Split the inference task among multiple entities. For example, run part of the model on the edge device and part on a more powerful server, or let multiple edge devices each handle a portion of the model.	- Allows use of larger models than a single device could handle, improving accuracy - Edge-cloud hybrid can balance latency vs load.	- Introduces dependency on network (latency and reliability of connection affect performance). - Requires careful partitioning to avoid too

		- Scales across device clusters for potential collective intelligence.	much communication overhead. - – More complex system architecture (coordination, synchronization issues).

7. RESULTS AND DISCUSSION

When we look at the techniques and use cases above, it becomes evident what the trade-offs and design choices are for putting LLMs on the edge. In this part, we talk about these trade-offs and what they mean for the big picture when it comes to using edge LLMs. This work doesn't give any new experimental results, but we use what we've learned from other research to look at what the best existing technology can do and where it still falls short.

Effectiveness of Compression vs. Accuracy Trade-off: One major point is how far model compression can be done before the performance of the LLM degrades significantly. Reported results from various studies indicate that techniques like 8-bit quantization will often preserve model accuracy within a few percentage points of the original model, while producing substantial speed-ups on edge hardware. Pruning and distillation outcomes vary depending upon how aggressively they are applied. A moderate pruning (removing perhaps 20-30% of parameters) might have little impact on accuracy, but extreme pruning (removing 80% or more) can hurt the model's understanding and generation capabilities significantly. Distilled models, on the other hand, can perform well – in some cases achieving ~90% of the teacher model's quality while being an order of magnitude smaller. These findings show that edge devices can indeed run useful versions of state-of-the-art LLMs, but careful tuning is required. For important applications, one might choose a slightly larger compressed model to ensure accuracy remains high, accepting a higher resource usage on a more capable edge device. Less important or more interactive applications might opt for a very small model to maximize speed and memory savings. The encouraging result here is that the community has demonstrated multiple ways to scale LLMs down, and ongoing research is likely to produce even more efficient methods.

Latency vs. Model Size (Edge-Cloud Balance): Another important trade-off is between latency and model complexity, especially in the context of edge-cloud hybrid approaches. Running a full large model in the cloud produces the best raw accuracy but introduces network latency which runs a tiny model fully on-device gives the fastest response but lower accuracy or capability. Many systems are now trying to achieve a balance by using multi-tiered deployments. For example, a device could first try to answer using its on-board compressed LLM for an immediate response. If that response's confidence is low or if the query is beyond its abilities, it can then query a cloud-hosted larger model. This way, the user often experiences low latency, and only occasionally (for very hard queries) experiences a delay but gets a more powerful answer. Empirical evidence suggests that a large portion of user queries or inputs in some domains can be handled by smaller local models, with only a small fraction needing the big models in the cloud. The discussion here highlights that the optimal solution is not always purely edge or purely cloud, a hybrid that dynamically offloads as needed can yield a superior combination of responsiveness and accuracy. The downside is increased system complexity (having to maintain two models and a decision mechanism), but frameworks are emerging to support this seamlessly.

Scalability and Resource Considerations: When deploying LLMs across potentially thousands of edge devices, questions of scalability arise beyond just the model performance. For example, if each edge device is running a computation-intensive model continuously, the aggregate energy consumption could be significant. One result from recent experimentation is that quantized models not only run faster but also consume less energy, which is vital for battery-powered devices. Additionally, some studies note that distributing tasks can lead to hotspots. For example, if all devices offload to a single edge server, that server becomes a bottleneck. Thus, architects are considering more distributed edge hierarchies (multiple edge servers, peer-to-peer sharing of load, etc.) to avoid single points of failure or a bottlenext. The implication is that edge LLM deployment is as much a systems engineering challenge as it is a model optimization challenge. Tools and middleware that can monitor device loads, make scheduling decisions, and balance workloads are becoming important parts of the solution (especially in industrial or city-scale IoT deployments).

Quality of Service and User Experience: From a user or application perspective, one must consider how edge-based LLMs impact the end-user experience. One clear positive result is the reduction in latency, users notice and appreciate faster responses and the ability to use features offline. On the other hand, if a highly compressed on-device model occasionally produces errors or lower-quality outputs (compared to a powerful cloud model), user trust could be affected. Managing this might involve transparently communicating to users (for instance, a device might say “I’m working offline with limited mode” vs “Connected to full server mode” in an interface). In critical applications like healthcare, rigorous validation is needed to ensure that any sacrifice in model performance due to edge optimization does not lead to incorrect decisions. In our review, we found that for many consumer applications (like keyboard suggestions, voice assistants for general queries, etc.), edge-optimized models have reached a point where their outputs are very good for everyday use. However, for highly specialized knowledge or extremely complex reasoning, the largest models (which likely still require cloud infrastructure) maintain an edge. This highlights a continuum of use cases not all AI tasks need the biggest model. Edge LLMs are proving sufficient for a wide middle ground of tasks, and using them can drastically improve responsiveness and privacy, which are part of the quality of service considerations.

Integration with Edge Hardware Advances: It’s worth discussing the interplay between LLM deployment strategies and advances in hardware. The results we see today are partly because modern edge devices are far more powerful than those of even a few years ago. Mobile processors now often include AI accelerator cores (NPUs or DSPs optimized for neural network operations). Our discussion noted that quantization aligns well with such hardware – indeed, many mobile NPUs are designed to excel at 8-bit integer operations. This means that a quantized LLM can run not just in theory but in practice fast on a phone using those neural chips. Similarly, as small-form-factor GPUs and specialized edge TPUs become more common on devices like drones or industrial controllers, they enable running medium-sized models that previously were infeasible. The trend suggests a co-evolution: as model optimization techniques improve, hardware is also improving**,** and together they broaden the horizon of what can be done on the edge. For example, an edge device in 2025 might run a 6-billion-parameter model in a few hundred milliseconds due to these combined advances, whereas in 2020 that would have been unimaginable. The discussion point here is optimistic: hardware and software innovations are synergistic in pushing LLMs to the edge, and early results indicate substantial progress.

Remaining Challenges: Despite the positive trajectory, the discussion would be incomplete without acknowledging what remains difficult. Very large LLMs (tens of billions of parameters) are still generally out of reach for true edge deployment except in highly fragmented ways. If an application truly needs the full power of something like GPT-4, currently one still has to rely on a cloud backend or a powerful server; edge strategies would revolve around doing pre-processing or caching on-device rather than running the whole model. Another challenge is maintaining and updating models on many distributed devices. Ensuring all edge nodes have the latest model (especially if they learn locally) can be complex. There's also the issue of monitoring and managing models in the wild – if an edge model starts behaving oddly (perhaps due to unexpected inputs or drift in local fine-tuning), detecting and correcting that is harder than in a centralized setup. These issues suggest that while edge deployment can augment and enhance AI services, in many cases a hybrid approach with cloud oversight is prudent to ensure reliability and consistency.

In conclusion of this discussion, the implication is that deploying LLMs at the edge is not an all-or-nothing proposition. It involves a nuanced approach where the specific constraints and goals of the application dictate which combination of strategies to use. The results surveyed in this paper demonstrate that edge LLMs can work and provide clear benefits in latency, privacy, and personalization. The trade-offs primarily in terms of accuracy vs. efficiency and complexity of system design are increasingly manageable with new techniques. As research continues, we expect the gap between edge and cloud capabilities to further narrow, making it feasible to push even more advanced AI models into everyday devices.

8. FUTURE DIRECTIONS

The field of large language models in edge computing is rapidly evolving, and there are several promising avenues and open challenges that will likely guide future research and development. Based on the current trends and the gaps identified in our review, we highlight several future directions and opportunities:

Design of Edge-Optimized LLM Architectures: One important goal is to build new model architectures that are specifically designed for use at the edge. Researchers can make small but powerful LLMs from scratch instead of always taking a big cloud-trained transformer and compressing it. This could mean using new network topologies that use fewer parameters or using design-time techniques like modular networks and sparsity. The goal would be to get good at interpreting and generating language with fewer parameters than normal. This would make models easier to use on limited hardware. This could also mean looking at bio-inspired or neuromorphic methods for processing language that might work better with technology in the long run.

Advances in Edge Hardware and Acceleration: We expect edge devices to have more specialized accelerators for AI workloads on the hardware side. Neural processing units with more memory and computation throughput for AI will probably come with future smartphones, AR/VR glasses, automobiles, and IoT gateways. One way to move forward is to build LLM algorithms with these new hardware features in mind. For instance, we could use the new hardware's ability to do 4-bit arithmetic or sparsity. Also, using less energy when computing (such using analog computing or low-power modes) could make it easier to operate LLMs on devices that run on batteries. As this technology becomes available, it will be vital to go back and look at methods to make the most of them (like how quantization became popular when 8-bit hardware was ubiquitous).

Dynamic and Adaptive Deployment Strategies: We expect to see smarter systems that can choose how and where to operate an LLM on their own. AI could be used to govern AI in future edge deployments. For example, a meta-controller could watch the edge device's condition (battery level, current network latency, user's urgency) and switch between a local model and a cloud model or change the model's complexity up or down. Another interesting area is adaptive splitting of models, where the line between edge and cloud can move based on what is happening in real time. This flexible method will make the system more robust and guarantee the best performance in all situations. It also raises problems about how to use optimization algorithms or reinforcement learning to make these deployment decisions in real time.

Federated and Continual Learning at Scale: Federated learning is a well-known idea, but making it work with really big models and millions of devices is still hard. In the future, efforts will probably focus on making federated learning of LLMs more trustworthy (by dealing with dropouts, different data distributions, and so on) and more efficient at communicating (by using update compression, asynchronous protocols, and so on). People are also interested in constant learning, which is when models continually learning from new data over time without losing what they already know. For long-term deployments, it will be very important to be able to learn new things about the device all the time (so that an edge LLM can adjust as its usage patterns change without losing everything). This could mean using methods to refresh bits of the model locally or syncing up with global models every so often in a way that keeps useful customisation.

Benchmarking and Standardization: As edge LLM implementation grows, the community would benefit from having uniform benchmarks and ways to evaluate them. In the future, there may be benchmark suites that contain activities that are similar to edge scenarios (for example, a suite of tasks for an on-device assistant that includes offline translation, Q&A with limited memory, and so on) to test models for both accuracy and efficiency (latency, memory, energy). In addition to benchmarking, standards might be set for how to share and show models that are optimized for edge. For example, there could be standardized model formats that include different precision weights or that let you deploy only part of a model.

Security and Trustworthiness Research: As edge-resident AI makes more important judgments, security, reliability, and moral issues become ever more important. Future research will look into ways to safely update edge models (so that attackers can't change them), find and stop model inversion or extraction attacks on stolen devices, and make models stronger against attacks in edge settings (where attackers might be able to touch devices). Another thing to think about is making sure that models are easy to understand and see on the edge. For instance, consumers should be able to see why their on-device model made a given recommendation. This might be done by adding explanation modules or logging systems that protect privacy. Taking care of these issues will be important for getting users to accept and safely use the technology in sensitive areas.

Expanding to Multimodal Edge AI: Language models (textual data) are getting a lot of attention right now, but LLMs are starting to work with other types of data, like vision and audio, to make multimodal models, like an AI that can see and communicate. It is still a frontier to discover how to bring these kinds of multimodal features to the edge. One possible future scenario is a set of smart glasses that can observe the world through a camera and describe it to a person who can't see it themselves. This would use an edge-deployed multimodal model, which means that both a vision model and a language model would function together on the device. Making sure that these complicated models can be optimized for edge deployment will make edge AI more useful for activities other than just text-based ones, like comprehending and interacting with the environment in a more generic way.

In short, LLMs have a very bright future in edge computing. We think that things will keep getting better, making it easier to run powerful AI models on regular devices. Not just one area will make these gains; they will come from a mix of machine learning research, hardware engineering, and system design. In the end, there would be more and more intelligent products, including smartphones, appliances, cars, and sensors, that can understand and reason in more advanced ways, all while being very efficient, private, and reliable. To reach this goal, we need to solve the last problems by coming up with new ideas and working together across fields.

9. CONCLUSION

Using massive language models in edge computing settings is an interesting mix of cutting-edge AI with real-world engineering. In this article, we talked about the reasons and new attempts to move LLM capabilities from the cloud to edge devices. We started by looking at the applications that might benefit from local AI processing. These include healthcare, personal devices, industrial systems, and cars. This shows that many areas are clearly interested in local AI processing for reasons of latency, privacy, and autonomy. Next, we talked about the problems that make edge deployment problematic, like not having enough hardware, having different types of hardware, network limits, and the requirement to keep security and real-time performance up. The community is working hard to solve these problems, even though they are big. There are several good reasons to push the limits of what can be done on-device, such as faster reaction times, the ability to work offline, better data privacy, less bandwidth utilization, and more customisation.

We also looked at a number of deployment options that let LLMs work within the limits of edge devices. Model compression (quantization, pruning, and distillation) can make models smaller so they can fit on smaller hardware. Federated learning and on-device training let edge devices help improve models while keeping data local. Distributed inference strategies use collaboration between devices and the cloud to handle big workloads. Practitioners have shown that surprisingly powerful language models may operate on edge platforms using these methods. This is especially true when both algorithms and hardware get better.

Our discussion made it clear that putting LLMs on the edge is a balancing act that often requires making choices between model accuracy and system efficiency. There is no one-size-fits-all answer; instead, the best way to deploy might be to use a mix of tactics and change them to meet the situation. It is apparent that edge and cloud solutions work together rather than against each other. Hybrid systems can get the best of both worlds by employing edge processing for speed and privacy and cloud processing for heavy lifting when needed. We are starting to see smart systems that can easily move AI tasks from one device to another and from the cloud to the device.

As we think about where we are now and where we're going (as described in Future Directions), it's clear that we're still in the early stages of edge LLM deployment. In the next several years, we will probably see even better models, better edge hardware, and more developed methods that make the problems we face today easier to deal with. As these improvements happen, we expect large language models to be used more and more in everyday devices and settings. This will make interactions more interesting and allow for wiser autonomous behavior in a wide range of applications.

In conclusion, getting huge language models to the edge is both a technological difficulty and a necessary step in the growth of AI use. The initiatives discussed in this paper show a lot of progress, and they also show a bigger trend: AI is getting closer to users and data sources. Researchers and engineers will be able to fully use LLMs in edge computing if they keep working on the problems that are still open and come up with new ways to deploy them. This will result in AI systems that are more responsive, secure, and aware of their surroundings. This combination of advanced language comprehension with edge computing will lead to the next generation of smart apps that make our lives better while still respecting the real-world values and limits that are so crucial, like privacy and reliability.

REFERENCES

1. Bhardwaj, Sarthak, Pardeep Singh, and Mohammad Khalid Pandit. "A survey on the integration and optimization of large language models in edge computing environments." 2024 16th International Conference on Computer and Automation Engineering (ICCAE). IEEE, 2024.

2. Zheng, Yue, et al. "A review on edge large language models: Design, execution, and applications." ACM Computing Surveys (2024).

3. Yuan, Xingyu, et al. "Generative inference of large language models in edge computing: An energy efficient approach." 2024 International Wireless Communications and Mobile Computing (IWCMC). IEEE, 2024.

4. Chen, Catherine Yu-Chi, et al. "Conformal Tail Risk Control for Large Language Model Alignment." arXiv preprint arXiv:2502.20285 (2025).

5. Eliganti Ramalakshmi, Venkata Srinivas Kompally, Baddam Deepika Reddy. (2020). Solar Powered Smart Irrigation and Monitoring System for Greenhouse Farming using IoT. International Journal of Advanced Science and Technology, 29(04), 8239 -. Retrieved from http://sersc.org/journals/index.php/IJAST/article/view/30559

6. S. K. Gunda, "Comparative Analysis of Machine Learning Models for Software Defect Prediction," 2024 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ICPECTS62210.2024.10780167. keywords: {Training;Logistic regression;Analytical models;Accuracy;Nearest neighbor methods;Predictive models;Software;Software reliability;Regression tree analysis;Testing;Software Defect Detection;Machine Learning;Logistic Regression;KNN;Decision Tree},

7. V. S. Kompally, "A microservices-based hybrid cloud-edge architecture for real-time IIoT analytics," Journal of Information Systems Engineering and Management, vol. 10, no. 16s, 2025. doi: 10.52783/jisem.v10i16s.2567

8. Tang, Yehui, et al. "A survey on transformer compression." arXiv preprint arXiv:2402.05964 (2024).

9. Ye, Rui, et al. "Openfedllm: Training large language models on decentralized private data via federated learning." Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining. 2024.

10. Hou, Zejiang, and Sun-Yuan Kung. "Multi-dimensional model compression of vision transformer." 2022 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2022.

11. Jiang, Feibo, et al. "Personalized wireless federated learning for large language models." arXiv preprint arXiv:2404.13238 (2024).

12. Du, Jiangsu, et al. "Co-designing Transformer Architectures for Distributed Inference with Low Communication." IEEE Transactions on Parallel and Distributed Systems (2024).

13. Rancea, Alexandru, Ionut Anghel, and Tudor Cioara. "Edge computing in healthcare: Innovations, opportunities, and challenges." Future internet 16.9 (2024): 329.

14. Gupta, Piyush, et al. "Prediction of health monitoring with deep learning using edge computing." Measurement: Sensors 25 (2023): 100604.

15. Gong, Taiyuan, et al. "Edge intelligence in intelligent transportation systems: A survey." IEEE Transactions on Intelligent Transportation Systems 24.9 (2023): 8919-8944.

16. Purohit, Rutuja, and Sanjay Bang. "SecurAI: Leveraging Edge Computing and Large Language Models for Intelligent Surveillance." 2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG). IEEE, 2024.

17. Li, Xiaoxia, et al. "Edge-computing-enabled unmanned module defect detection and diagnosis system for large-scale photovoltaic plants." IEEE Internet of Things Journal 7.10 (2020): 9651-9663.

18. Kubiak, Kacper, Grzegorz Dec, and Dorota Stadnicka. "Possible applications of edge computing in the manufacturing industry—systematic literature review." Sensors 22.7 (2022): 2445.

19. Bonam, Janakiramaiah, et al. "Lightweight cnn models for product defect detection with edge computing in manufacturing industries." Journal of Scientific & Industrial Research 82.04 (2023): 418-425.

20. Chen, Jiao, et al. "Towards General Industrial Intelligence: A Survey of Continual Large Models in Industrial IoT." arXiv preprint arXiv:2409.01207 (2024).

    Awaisi, Kamran Sattar, Qiang Ye, and Srinivas Sampalli. "A Survey of Industrial AIoT: Opportunities, Challenges, and Directions." IEEE Access (2024).

21. Linghe Kong, Jinlin Tan, Junqin Huang, Guihai Chen, Shuaitian Wang, Xi Jin, Peng Zeng, Muhammad Khan, and Sajal K. Das. 2022. Edge-computing-driven Internet of Things: A Survey. ACM Comput. Surv. 55, 8, Article 174 (August 2023), 41 pages. https://doi.org/10.1145/3555308

22. Kyle Hoffpauir, Jacob Simmons, Nikolas Schmidt, Rachitha Pittala, Isaac Briggs, Shanmukha Makani, and Yaser Jararweh. 2023. A Survey on Edge Intelligence and Lightweight Machine Learning Support for Future Applications and Services. J. Data and Information Quality 15, 2, Article 20 (June 2023), 30 pages. https://doi.org/10.1145/3581759

23. He, Ying, et al. "Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: An active inference approach." IEEE Transactions on Mobile Computing (2024).

24. Zhang, Mingjin, et al. "Edgeshard: Efficient llm inference via collaborative edge computing." IEEE Internet of Things Journal (2024)

25. Zhao, Wentao, et al. "Edge and terminal cooperation enabled llm deployment optimization in wireless network." 2024 IEEE/CIC International Conference on Communications in China (ICCC Workshops). IEEE, 2024.

26. Bhardwaj, Sarthak, Pardeep Singh, and Mohammad Khalid Pandit. "A survey on the integration and optimization of large language models in edge computing environments." *2024 16th International Conference on Computer and Automation Engineering (ICCAE)*. IEEE, 2024.

27. Xu, Daliang, et al. "Llmcad: Fast and scalable on-device large language model inference." *arXiv preprint arXiv:2309.04255* (2023).

28. Yang, Zheming, et al. "Perllm: Personalized inference scheduling with edge-cloud collaboration for diverse llm services." arXiv preprint arXiv:2405.14636 (2024)

29. Zhu, Jing, et al. "Edge intelligence-assisted animation design with large models: a survey." Journal of Cloud Computing 13.1 (2024): 48.

30. Rancea, Alexandru, Ionut Anghel, and Tudor Cioara. "Edge computing in healthcare: Innovations, opportunities, and challenges." Future internet 16.9 (2024): 329

31. Rivkin, Dmitriy, et al. "AIoT Smart Home via Autonomous LLM Agents." IEEE Internet of Things Journal (2024).

32. ‘Kök, İbrahim, Orhan Demirci, and Suat Özdemir. "When IoT Meet LLMs: Applications and Challenges." 2024 IEEE International Conference on Big Data (BigData). IEEE, 2024.

33. Chen, Jiao, et al. "Edge-cloud collaborative motion planning for autonomous driving with large language models." 2024 IEEE 24th International Conference on Communication Technology (ICCT). IEEE, 2024.

34. Xu, Ronghua, Deeraj Nagothu, and Yu Chen. "AR-Edge: Autonomous and Resilient Edge Computing Architecture for Smart Cities." Edge Computing Architecture-Architecture and Applications for Smart Cities. IntechOpen, 2024.

35. Yu, Zhongzhi, et al. "Edge-llm: Enabling efficient large language model adaptation on edge devices via unified compression and adaptive layer voting." Proceedings of the 61st ACM/IEEE Design Automation Conference. 2024.

36. Jayakodi, Nitthilan Kannappan, Janardhan Rao Doppa, and Partha Pratim Pande. "A general hardware and software Co-design framework for energy-efficient edge AI." *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021.

37. Singh, Raghubir, and Sukhpal Singh Gill. "Edge AI: a survey." Internet of Things and Cyber-Physical Systems 3 (2023): 71-92.

38. Friha, Othmane, et al. "Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness." IEEE Open Journal of the Communications Society (2024).

39. Zhang, Zhaoyun, and Jingpeng Li. "A review of artificial intelligence in embedded systems." Micromachines 14.5 (2023): 897.

40. Natarajan, Sundaram, et al. "Diagnostic accuracy of community-based diabetic retinopathy screening with an offline artificial intelligence system on a smartphone." JAMA ophthalmology 137.10 (2019): 1182-1188.

41. Wu, Shanshan, et al. "Prompt public large language models to synthesize data for private on-device applications." *arXiv preprint arXiv:2404.04360* (2024).

42. Li, En, et al. "Edge AI: On-demand accelerating deep neural network inference via edge computing." IEEE transactions on wireless communications 19.1 (2019): 447-457.

43. Moon, Ji Joong, et al. "A new frontier of ai: On-device ai training and personalization." arXiv preprint

arXiv:2206.04688 (2022).

44. Zhu, Xunyu, et al. "A survey on model compression for large language models." Transactions of the Association for Computational Linguistics 12 (2024): 1556-1577.

45. Hilmkil, Agrin, et al. "Scaling federated learning for fine-tuning of large language models." International Conference on Applications of Natural Language to Information Systems. Cham: Springer International Publishing, 2021.

46. Xu, Jiajun, et al. "On-device language models: A comprehensive review." arXiv preprint arXiv:2409.00088 (2024).

47. Yang, Zheming, et al. "Perllm: Personalized inference scheduling with edge-cloud collaboration for diverse llm services." arXiv preprint arXiv:2405.14636 (2024).

48. Yao, Jiangchao, et al. "Edge-cloud polarization and collaboration: A comprehensive survey for ai." IEEE Transactions on Knowledge and Data Engineering 35.7 (2022): 6866-6886.

49. Li, Jinrong, et al. "CoLLM: A Collaborative LLM Inference Framework for Resource-Constrained Devices." 2024 IEEE/CIC International Conference on Communications in China (ICCC). IEEE, 2024.

50. Hosseinzadeh, Minoo, et al. "Optimal accuracy-time trade-off for deep learning services in edge computing systems." ICC 2021-IEEE International Conference on Communications. IEEE, 2021.

51. Stojkovic, Jovan, et al. "Towards greener llms: Bringing energy-efficiency to the forefront of llm inference." arXiv preprint arXiv:2403.20306 (2024).

52. Shen, Tao, et al. "Will LLMs Scaling Hit the Wall? Breaking Barriers via Distributed Resources on Massive Edge Devices." arXiv preprint arXiv:2503.08223 (2025).

53. Hayyolalam, Vahideh, Safa Otoum, and Öznur Özkasap. "Dynamic QoS/QoE-aware reliable service composition framework for edge intelligence." Cluster Computing 25.3 (2022): 1695-1713.

54. Zhou, P. J., et al. "A Neuromorphic Transformer Architecture Enabling Hardware-Friendly Edge Computing." IEEE Transactions on Circuits and Systems I: Regular Papers (2025).

55. Mazumder, Arnab Neelim, et al. "A survey on the optimization of neural network accelerators for micro-ai on-device inference." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 11.4 (2021): 532-547.

56. Rui, Lanlan, et al. "Smart network maintenance in an edge cloud computing environment: An adaptive model compression algorithm based on model pruning and model clustering." IEEE Transactions on Network and Service Management 19.4 (2022): 4165-4175.

57. Wang, Zifeng, et al. "Sparcl: Sparse continual learning on the edge." Advances in Neural Information Processing Systems 35 (2022): 20366-20380.

58. Zhang, Xuechen, et al. "Fedyolo: Augmenting federated learning with pretrained transformers." arXiv preprint arXiv:2307.04905 (2023).

59. Hu, Yaqi, et al. "A Cloud-Edge Collaborative Architecture for Multimodal LLMs-Based Advanced Driver Assistance Systems in IoT Networks." *IEEE Internet of Things Journal* (2024).

60. HaddadPajouh, Hamed, et al. "AI4SAFE-IoT: An AI-powered secure architecture for edge layer of Internet of things." *Neural Computing and Applications* 32.20 (2020): 16119-16133.

61. Qin, Ruiyang, et al. "Empirical guidelines for deploying llms onto resource-constrained edge devices." *ACM Transactions on Design Automation of Electronic System s* (2024).