Journal of Electrical Engineering, Electronics, Control and Computer Science – JEEECCS, Volume 12, Issue 41, pages 21-30, 2026

Large Language Models: Architectures and Applications in Electrical and Computer Engineering - A Short Review

1st Andreea-Daniela Savu
Faculty of Electronics, Communications and Computers
National University of Science and Technology Politehnica Bucharest
Pitesti, Romania
savuandreeadaniela@gmail.com

2nd Nicu Bizon
Department of Electronics, Computers and Electrical Engineering
National University of Science and Technology Politehnica Bucharest
Pitesti, Romania
nicubizon@yahoo.com

3rd Sebastian-Alexandru Dragusin
Department of Electronics, Computers and Electrical Engineering
National University of Science and Technology Politehnica Bucharest
Pitesti, Romania
dragusin.sebi@yahoo.com

Abstract –Large Language Models (LLMs) have rapidly evolved from research curiosities to general-purpose AI components increasingly embedded into engineering workflows. Although several surveys address theoretical foundations and broad application domains, a concise, engineering-oriented perspective tailored to electrical and computer engineering remains limited. This short review outlines the evolution and taxonomy of current LLM families, including proprietary and open-source models, code-oriented variants and multimodal extensions. Core techniques for adapting LLMs to domain-specific tasks are summarized, with emphasis on instruction tuning, parameter-efficient fine-tuning (e.g., LoRA (Low-Rank Adaptation), QLoRA (Quantized LoRA)) and retrieval-augmented generation, together with recent advances in tool use and LLM-based agents. Evaluation methodologies are briefly reviewed, covering general benchmarks, trustworthiness and safety aspects, as well as domain-specific assessment for coding, control and signal processing tasks. Representative application patterns in electrical engineering, control and computer science are highlighted, and key challenges and future research directions related to hallucination mitigation, robustness, efficiency and secure deployment are outlined. The survey is intended as a practical reference for integrating LLMs into engineering systems and educational environments.

Keywords- large language models, transformer, parameter-efficient fine-tuning, retrieval-augmented generation, evaluation, electrical and computer engineering

I. INTRODUCTION

LLMs based on transformer architectures are rapidly becoming foundational components in electrical and computer engineering, reshaping how engineers design, analyze and deploy complex systems. Building on advances in deep learning and large-scale self-supervised training, modern LLM families now include proprietary and open-source models, specialized code-centric variants and multimodal extensions that can jointly reason over text, code and signals. These capabilities enable new forms of automation and assistance in tasks such as code generation, documentation, system modeling, test-case creation and interactive technical support [1].

For practitioners in electrical and computer engineering, however, the key challenge is no longer limited to understanding the underlying theory, but increasingly concerns model choice, adaptation and evaluation. Engineers must decide which model family is appropriate for a given task, how to adapt it efficiently under hardware and data constraints, and how to assess its behavior under realistic operating conditions. Questions related to latency, memory footprint, quantization, robustness to domain-specific inputs and compliance with safety requirements become central when LLMs are embedded into engineering workflows and tools [2].

In parallel, the broader research community has produced a growing number of surveys on LLMs, mainly focused on theoretical foundations, natural language processing benchmarks and high-level application overviews across diverse domains such as general AI (Artificial Intelligence) assistants,

healthcare or education [3]. While such surveys provide valuable context, they typically treat engineering use cases only tangentially and rarely address the specific constraints and requirements encountered in electrical and computer engineering [3]. Aspects such as integration with simulation environments, interaction with domain-specific tools and languages (e.g., HDL (Hardware Description Language), MATLAB (MATrix LABoratory), SPICE (Simulation Program with Integrated Circuit Emphasis)), or support for embedded and resource-constrained platforms are often underrepresented [4].

To address this gap, the present short review provides an engineering-oriented overview of LLM architectures and adaptation techniques that are most relevant to electrical and computer engineering [5]. Attention is given to the main transformer-based LLM families, covering both proprietary and open-source models, as well as code-centric and multimodal variants. Core techniques for adapting LLMs to domain-specific tasks are summarized, with emphasis on instruction tuning, parameter-efficient fine-tuning and retrieval-augmented generation, together with emerging patterns for tool integration and autonomous LLM-based agents. Emphasis is placed on evaluation methodologies that go beyond generic benchmarks to capture trustworthiness, safety and domain-specific performance in coding, control and signal-processing workflows [2].

Beyond architectural and methodological aspects, recent application patterns of LLMs in electrical engineering, control and computer systems are highlighted, including examples related to code synthesis and debugging, support for modeling and simulation tasks, documentation and report generation, as well as educational use cases. These examples illustrate how LLMs can be embedded into real-world engineering pipelines, from early design stages to testing, deployment and maintenance of complex systems [6].

By synthesizing recent research and practical case studies, this short review aims to offer a concise guide to integrating LLMs into engineering contexts and educational settings, while also outlining open problems in robustness, hallucination mitigation, efficiency and secure deployment. The analysis is intended to support both researchers and practitioners in making informed decisions about the adoption and configuration of LLM-based solutions in electrical and computer engineering [7].

The remainder of this short review is organized as follows. Section 2 presents a literature review on transformer architectures and LLMs, synthesizing key surveys, architectural analyses, evaluation frameworks and application studies relevant to engineering. Section 3 discusses the architectures and applications of large language models in electrical and computer engineering, beginning with the foundations of transformer-based LLMs, then outlining representative model families (GPT (Generative Pre-trained Transformer), LLaMA (Large Language Model Meta AI), Mistral, Qwen and code-oriented models), followed by architectural variants and adaptation techniques for engineering tasks and, finally, application domains and usage patterns in ECE. Section 4 concludes the paper and highlights open challenges and future research directions.

II. Literature Review

Paper [8] introduces the Transformer, the first sequence transduction architecture built entirely on self-attention without any recurrent or convolutional layers. The authors replace sequence aligned recurrence with stacked multi head self-attention and position wise feed forward networks in an encoder–decoder structure, enabling highly parallel computation and shortening the path length for modeling long range dependencies. Evaluated on WMT (Workshop on Machine Translation) 2014 English-German and English-French machine translation [5], the Transformer achieves new state of the art BLEU (Bilingual Evaluation Understudy) scores while requiring substantially less training time and compute than prior RNN (Recurrent Neural Network) and CNN (Convolutional Neural Network) based models, demonstrating that attention only architectures can outperform more complex recurrent and convolutional systems for large scale translation tasks.

Study [9] examines the GPT-3 architecture, a 175 billion parameter autoregressive transformer trained on large scale internet text and evaluated as a zero, one, and few shot learners without task specific fine tuning. The authors show that scaling model size leads to smooth performance gains across many NLP (Natural Language Processing) benchmarks, with GPT-3 achieving strong results on language modeling, cloze and completion tasks, question answering, and other standard datasets using only in context instructions and examples. At the same time, the paper reports notable weaknesses on several reasoning focused benchmarks and discusses broader issues such as data contamination, bias, and societal impacts, concluding that large scale pretraining substantially improves in context learning while leaving important open challenges.

Study [10] synthesizes the evolution of large language models from early statistical methods to modern transformer-based architectures and generative AI systems. The authors describe core LLM architectures and training paradigms, catalog major foundation models, and organize applications across domains such as medicine, education, finance, engineering, media, law, and agriculture, emphasizing both text-only and multimodal (vision-language) use cases. The survey also analyzes key challenges: ethical and societal risks, bias, hallucination, privacy, security, environmental and computational costs, and discusses techniques for robustness, controllability, and responsible deployment, positioning LLMs as powerful but nontrivial technologies whose reliable real-world use requires careful governance and best-practice guidelines.

The work [11] provides a structured overview of the evolution, architectures, training strategies, and applications of LLMs. The authors outline the progression from early GPT and BERT (Bidirectional Encoder Representations from Transformers) models to

modern multimodal systems, highlighting key techniques such as pre-training [12], instruction tuning, RLHF (Reinforcement Learning from Human Feedback), and dataset design. Their survey emphasizes the broad impact of LLMs across domains including engineering, healthcare, and finance, while also noting challenges related to safety, bias, interpretability, and computational cost. Overall, the study offers a comprehensive reference for understanding current developments and open issues in LLM research.

Study [3] provides an analysis of LLMs, offering a comprehensive overview of their evolution, architectures, training methods, and real-world applications. The objective is to consolidate recent advances in Transformer-based LLMs, including adaptation strategies such as PEFT (Parameter-Efficient Fine-Tuning) and RLHF, and to evaluate their capabilities across established benchmarks. The results show that modern LLMs are increasingly multimodal, agentic, and economically impactful, with significant effects on sectors such as engineering, healthcare, and education. The study confirms that despite rapid progress, key challenges remain, including high compute costs, alignment difficulties, evaluation gaps, and societal risks, highlighting the need for continued research in governance, efficiency, and responsible deployment.

Paper [13] traces the historical progression from statistical and early neural language models to modern transformer-based LLMs and positions LLMs as a fourth wave characterized by massive scale and emergent capabilities. The authors focus on three major model families (GPT, LLaMA, and PaLM (Pathways Language Model)), outlining their architectures, training data, and distinctive abilities such as in context learning, instruction following, and multi-step reasoning, and then summarize techniques for building and augmenting LLMs, including data preparation, pretraining, fine tuning, alignment, and agent-style tool use. The survey also catalogs key datasets and benchmarks, compares performance of prominent LLMs across representative tasks, and highlights open challenges around efficiency, hallucination, safety, alignment, and the shift toward smaller or post attention architectures and multimodal models as important directions for future research.

Study [14] focuses on how to assess and improve alignment rather than model capabilities. The authors propose a taxonomy of LLM trustworthiness with seven main categories: reliability, safety, fairness, resistance to misuse, explainability and reasoning, social norms, and robustness, split into 29 finer subcategories, and design measurement studies that show even well aligned models still fail on issues such as hallucination, bias, misuse, copyright leakage, and robustness to prompt attacks.

Paper [15] examines how transformer-based LLMs are adapted to programming tasks such as code completion, synthesis from natural language, repair, and explanation. The authors categorize code LLMs by architecture, training data (source repositories, docstrings, and issue reports), and objectives (next token prediction, infilling, and dual NL (Natural Language)-code modeling) and summarize common evaluation benchmarks spanning multiple languages and real-world repositories. They also discuss practical challenges, including security vulnerabilities, code correctness and test coverage, data licensing, and the risk of over-reliance by developers, highlighting open problems in trustworthy, domain-specialized code generation.

Study [16] covers emerging uses of large language models as assistants for simulation, modeling, experiment design, and automation in STEM (Science, Technology, Engineering, and Mathematics) domains [17]. The paper outlines typical multi-step workflows where LLMs generate code, interface with computational tools, interpret numerical results, and help document findings, while emphasizing the need for human verification and domain-specific grounding. Its further points to open challenges in numerical reliability, reproducibility, data privacy, and integration with existing engineering software stacks, arguing for tightly coupled human-AI collaboration rather than fully autonomous pipelines.

Paper [18] presents a structured overview of prompt engineering and retrieval-augmented generation as key techniques for controlling and extending LLM behavior. The authors categorize prompt patterns (instruction, few-shot, chain of thought, and role prompts), discuss common retrieval pipelines that couple LLMs with vector databases, and provide case studies in question answering and document-centric applications. They also highlight practical issues such as prompt sensitivity, context length limits, and knowledge freshness, motivating systematic prompt design and evaluation in real-world deployments.

Study [19] surveys clinical and biomedical applications of LLMs across documentation, clinical decision support, patient communication, and biomedical literature mining. The paper compares general-purpose and domain-specialized models on medical benchmarks, outlines typical fine-tuning and instruction-tuning strategies for clinical text, and reports both promising results (e.g., note summarization, question answering) and failure modes such as hallucinated facts and unsafe recommendations. It stresses regulatory, ethical, and safety constraints in healthcare, calling for rigorous evaluation, alignment with clinical guidelines, and careful workflow integration with human clinicians.

Study [20] summarizes the broader societal and economic implications of deploying LLMs at scale. The authors discuss impacts on labor markets, creativity, education, and scientific research, alongside risks involving misinformation, bias amplification, environmental cost, and concentration of power in a few model providers. They argue that technical advances in efficiency, transparency, and alignment must be coupled with governance mechanisms, standards, and participatory policymaking to ensure that LLMs are developed and used in ways that are socially beneficial and equitable.

III. Architectures and Applications of Large Language Models in Electrical and Computer Engineering

This section introduces the architectural and practical foundations needed to understand how large language models can be deployed as engineering tools in electrical and computer engineering.

A. Foundations of Transformer-Based Large Language Models

Transformer-based architectures constitute the core computational paradigm underlying modern LLMs. Transformers have largely replaced recurrent and convolutional sequence models by relying on self-attention, which enables highly parallel computation and efficient modeling of long-range dependencies. This subsection outlines the main building blocks of the Transformer: self-attention, multi head attention, feed forward networks, and positional encodings [21], and summarizes how these components are organized in encoder–decoder and decoder only LLM architectures [22].

The original Transformer is composed of an encoder and a decoder, each implemented as a stack of N identical layers. Encoder-decoder configurations, as in T5-like sequence-to-sequence models [23], remain important for tasks such as translation, while most large-scale generative LLMs (e.g. GPT style models) use only the decoder block in an autoregressive setup [24].

The encoder layer is composed of several key components. It includes a multi-head self-attention mechanism, which allows the model to capture relationships between all tokens in the input sequence. This is followed by a position-wise FFN (Feed-Forward Network) that processes each token independently. Each of these sublayers is wrapped with residual connections, followed by layer normalization, which helps stabilize training and improve convergence [8].

The decoder layer has a similar structure but introduces additional mechanisms. It employs masked multi-head self-attention to prevent the model from accessing future tokens, ensuring proper autoregressive generation. In full encoder–decoder architectures, the decoder also includes cross-attention over the encoder outputs, enabling the integration of information from the input sequence. As in the encoder, a position-wise FFN follows, and residual connections with layer normalization are applied after each sublayer [25].

Modern LLMs typically adopt a decoder-only autoregressive architecture, removing the traditional encoder component altogether. Instead of the original sinusoidal positional encodings, these models use rotary positional embeddings or learned positional encodings, which provide greater flexibility. Additionally, they rely on high-dimensional multi-head attention with many layers and attention heads, allowing them to model complex dependencies over long sequences [26].

For each token, the model computes three vectors via learned linear projections: query Q, key K, and value V. The scaled dot product attention is defined as equation (1) [27]:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

where d_k is the dimensionality of the key vectors. The scaling factor $\frac{1}{\sqrt{d_k}}$ controls the magnitude of the dot products and stabilizes optimization [27].

In multi head attention, this operation is performed in parallel across h heads (equation (2)) [27]:

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (2)$$

and the outputs are concatenated and projected (equation (3)) [27]:

$$\text{MHA}(Q,K,V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (3)$$

This multi head structure allows the model to capture diverse contextual relationships in different subspaces. Each Transformer layer contains a position wise FFN applied independently to each token representation (equation (4)) [27]:

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \quad (4)$$

GELU (Gaussian Error Linear Unit) activations are standard in contemporary LLMs due to their smooth nonlinear behavior and good empirical performance [27].

Because self-attention is permutation-invariant, positional information must be incorporated into token embeddings to preserve sequence order. The original Transformer architecture addressed this by introducing sinusoidal positional encodings, which use sine and cosine functions at different frequencies to represent token positions, with d denoting the model dimension (equation (5)). This design enables generalization to sequence lengths beyond those seen during training [27].

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (5)$$

Modern Transformer models often replace sinusoidal encodings with more flexible alternatives, such as RoPE (Rotary Positional Embeddings), which integrate positional information directly into the attention mechanism, ALiBi (Attention with Linear Biases), which applies linear positional biases to attention scores, and learned positional embeddings, where positional representations are optimized during training. These alternatives improve extrapolation to longer sequences and enhance stability at large scales [27].

Transformer-based encoder–decoder architecture provides a canonical reference design for many sequence-to-sequence large language models. In this configuration, an encoder stack transforms the input sequence into a set of contextual representations using layers of multi-head self-attention and position-wise FFN, each wrapped with residual connections and normalization. The decoder stack then generates the target sequence autoregressively by combining masked

self-attention over previously generated tokens with cross-attention to the encoder outputs [28], before mapping the final hidden states through a linear layer and SoftMax to produce token probabilities (Figure 1) [29].

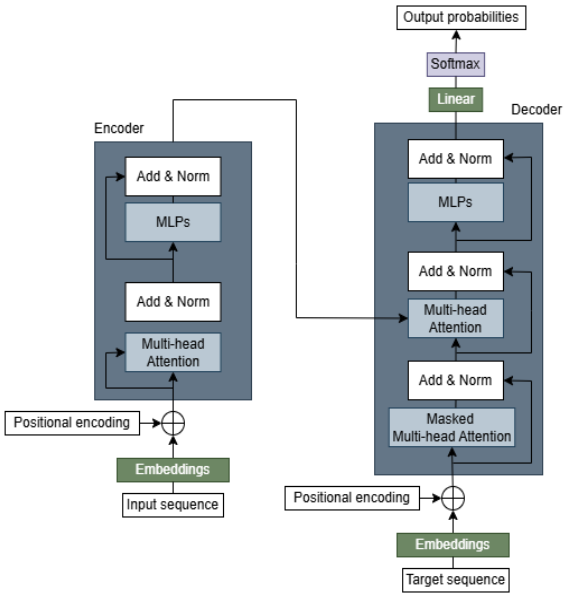

Figure 1. Transformer architecture with multi-head attention, feed-forward blocks and positional encodings for sequence-to-sequence modeling

B. Representative FamiQwen of Large Language Models

LLMs constitute a rapidly evolving landscape in which a variety of architectures and training philosophies coexist. Although most contemporary models share the fundamental Transformer-based decoder backbone, each family introduces its own priorities in terms of efficiency, multilingualism, specialization, or open-source accessibility. This section presents an integrated overview of several influential LLM families: GPT, LLaMA, Mistral, Qwen, and the main code-oriented models, culminating in a comparative characterization of what we call the broader FamiQwen ecosystem.

1. GPT Family (OpenAI)

The GPT family represents the canonical line of autoregressive large language models. Based on a decoder-only Transformer architecture, these models are trained through large-scale next-token prediction on heterogeneous corpora that include web text, books, code, and synthetic alignment data. Over successive generations, the GPT models have progressively increased both depth and hidden dimensionality while incorporating advanced alignment strategies that improve instruction following, reasoning, tool use, and multimodal processing. GPT remains a dominant benchmark for evaluating general-purpose LLM performance across a wide variety of tasks [30].

2. LLaMA Family (Meta)

Stop! The LLaMA series is designed as an efficient open-source alternative tailored for research and practical deployment. Its architectures maintain a decoder-only structure but emphasize optimized attention mechanisms, notably GQA (Grouped Query Attention), which significantly reduces memory consumption during inference. LLaMA models are generally more compact than proprietary counterparts, yet they provide competitive performance-per-parameter due to high-quality data curation and extended multilingual training. The release of LLaMA variants has stimulated a rich ecosystem of community-driven fine-tuned models used in academic and industrial applications [31].

3. Mistral Family (Mistral AI)

The Mistral models focus on maximizing efficiency and throughput without sacrificing accuracy. They rely on innovative mechanisms such as SWA (Sliding Window Attention), which scales linearly with sequence length and thus enables fast processing of extended contexts. Additionally, some variants, notably Mixtral, adopt Mixture-of-Experts designs that increase effective model capacity while activating only a subset of parameters during inference. Thanks to these optimizations, Mistral models achieve strong benchmarks despite relatively modest parameter counts, making them particularly appealing for large-scale, cost-sensitive deployments [32].

4. Qwen Family (Alibaba / ModelScope)

The Qwen family occupies a central position in the FamiQwen conceptual framework due to its versatility and broad domain coverage. Qwen models integrate robust multilingual capabilities particularly for English and Chinese, while supporting long-context reasoning, tool use, function calling, and diverse downstream integrations. The family encompasses base LLMs, instruct-tuned versions, code-specialized derivatives, and multimodal vision-language models. The emphasis on practical usability, extensibility, and high-quality pretraining data has established Qwen as one of the most flexible open-source LLM lineages [33].

5. Code-Oriented Models

Code-centered LLMs form a specialized subcategory distinct from general-purpose language models. These systems are trained on large corpora of source code drawn from multiple programming languages and repositories, enabling them to perform tasks such as code synthesis, debugging, refactoring, and documentation generation with high structural fidelity [34]. Representative models include CodeLLaMA, Qwen-Coder, DeepSeek-Coder, and StarCoder. Their tokenization strategies and learned representations are adapted to the syntax and hierarchical nature of programming languages, resulting in improved precision and robustness for software-engineering tasks [35].

When examined side by side, representative LLM families share a common architectural backbone but diverge along strategic dimensions presented in TABLE I. Across these families, the Transformer decoder remains the unifying foundation, while improvements arise from innovations in attention mechanisms, positional encoding, data curation, and training efficiency.

TABLE I. COMPARATIVE OVERVIEW OF REPRESENTATIVE LARGE LANGUAGE MODEL FAMILIES

Model Family	Architecture	Core Innovations	Strengths	Typical Use Cases
GPT	Decoder-only	Advanced alignment, multimodality	reasoning, tool use	general AI agents
LLaMA	Decoder-only	GQA, efficient scaling	open research, finetuning	academic, enterprise
Mistral	Optimized decoder	SWA, MoE (Mixture of Experts) variants	efficiency, throughput	large-scale inference
Qwen	Decoder-only	multilingual, function-calling	versatility, multilingualism	assistants, enterprise apps
Code Models	Decoder-only	code-heavy corpora, syntax modeling	precise code generation	dev tools, automation

C. *Architectural Variants and Adaptation Techniques for Engineering Tasks*

LLMs can be instantiated in several architectural configurations that determine the types of engineering tasks they can support and the efficiency with which they can be adapted to domain-specific requirements. In ECE (Electrical and Computer Engineering), selecting an appropriate architecture: encoder-only, decoder-only or encoder–decoder is critical for integrating LLMs into workflows involving code generation, signal analysis, system modelling, or documentation. Alongside architectural choices, modern adaptation methods such as instruction tuning, full fine-tuning and PEFT play an essential role in aligning LLM behavior with engineering constraints, data availability and computational resources [36].

Encoder-only architectures, exemplified by BERT and its derivatives, produce contextualized representations of the input sequence through bidirectional attention. These models are not generative but offer strong discriminative capabilities, making them well suited for classification, anomaly detection, requirements tagging, log interpretation and other analysis-oriented tasks in ECE. Their capacity to extract semantic and structural information from technical text allows engineers to build systems for fault detection, requirements validation or signal annotation with high precision and relatively low computational cost [37].

Decoder-only architectures, such as GPT, LLaMA, Mistral and Qwen, operate autoregressively and are optimized for generating coherent sequences of tokens. These models dominate applications requiring synthesis, such as code generation, HDL module creation, refactoring, debugging, testbench generation and technical documentation. In engineering workflows, decoder-only LLMs are particularly valuable due to their ability to integrate reasoning steps, derive algorithms from natural-language descriptions, and generate scripts for simulation or analysis tools. Their flexibility and contextual depth make them the primary choice for building AI-assisted programming environments and design-automation systems [38].

Encoder-decoder architectures, represented by models such as T5, FLAN (Finetuned Language Net)-T5 or BART, specialize in structured input–output transformations, including translation, summarization, specification extraction and document-to-code transformations. These models are effective in ECE scenarios that require converting between formats or abstraction levels, such as turning hardware specifications into structured constraints, summarizing engineering reports, transforming logs into interpretable narratives or mapping input schemas to simulation configurations. Their separation of encoding and decoding processes enables more controlled and stable transformations compared to purely autoregressive models [39].

Beyond architectural structure, adaptation techniques are crucial for enabling LLMs to operate effectively in engineering environments, where data may be scarce, proprietary or domain specific. Instruction tuning represents the first layer of adaptation, enabling models to follow engineering-specific prompts by training on curated input-output pairs that reflect tasks such as generating firmware code, explaining circuit behavior or producing SPICE configurations. This procedure improves controllability but typically requires only a modest dataset [40].

Full fine-tuning adjusts all model parameters to a specialized engineering domain, offering maximal performance at the cost of high computational demand. It is appropriate for niche industrial applications, proprietary toolchains, domain-restricted languages or environments that require strict model alignment. However, due to resource cost and risks such as catastrophic forgetting, full fine-tuning is less common in academic and enterprise settings unless large domain datasets are available [41].

PEFT methods, most notably LoRA and QLoRA, have emerged as practical alternatives for adapting large models in computationally constrained environments. LoRA introduces low-rank trainable matrices into the attention or feed-forward layers, enabling high-quality specialization while updating only a small fraction of the parameters. QLoRA extends this approach by quantizing the base model to 4-bit precision and training LoRA adapters on top of it, allowing fine-tuning of billion-parameter models using a single GPU (Graphics Processing Unit). These techniques are particularly valuable in ECE settings where models must be adapted to proprietary data (e.g., logs, embedded firmware code, domain-specific HDL corpora) without exposing that data to external systems or incurring large energy and hardware costs [42].

Collectively, the combination of architectural variants and adaptation techniques offers engineers multiple pathways for integrating LLMs into their workflows. Encoder-only models enable high-accuracy analysis, decoder-only models provide powerful generative capabilities for code and design tasks, and encoder-decoder models support structured transformations essential for documentation and translation. Instruction tuning, fine-tuning and PEFT methods ensure that these architectures can be tailored to specialized engineering requirements, balancing performance, cost and domain alignment [43].

IV. Application Domains and Usage Patterns in Electrical and Computer Engineering

LLMs are increasingly embedded as active components within ECE workflows, moving beyond generic conversational use toward domain-specific assistance, automation and decision support. Integrated with code repositories, simulation tools, documentation systems and educational platforms, LLMs support tasks such as code and HDL synthesis, testbench generation, configuration of SPICE and MATLAB-based simulations, summarization of technical reports and interactive tutoring. In these settings, they operate under strict correctness, reliability and traceability constraints, since errors can propagate into hardware designs, control algorithms or safety-critical software. Consequently, application domains and usage patterns in ECE are best understood in terms of concrete roles within existing engineering pipelines, ranging from "co-pilots" that propose candidate solutions under human supervision to orchestrators that coordinate external tools: across tasks related to coding, modeling and simulation, documentation and knowledge management, and education and training.

The adoption ECE is concentrated in a set of recurring domains that directly impact engineering productivity. As shown in TABLE II, LLMs are primarily used for software and firmware development, data analysis and signal processing, system design and modelling, technical documentation and compliance support, as well as education and knowledge transfer. Across these areas, LLMs automate code creation, streamline engineering data processing, assist with architectural decision-making, ensure documentation consistency, and enhance the accessibility of technical knowledge, making them valuable tools throughout the modern ECE workflow [44].

TABLE II. Application Domains of LLMs in Electrical and Computer Engineering

Domain	Typical Tasks in ECE	Relevant LLM Types	Engineering Observations
Software & Firmware Development	Code generation, refactoring, driver development, HDL (Verilog/VHDL (VHSIC (Very High-Speed Integrated Circuit) HDL) modules, testbench creation, protocol implementation	GPT, LLaMA, Qwen, CodeLLaMA, Qwen-Coder, DeepSeek-Coder	High accuracy in structured generation; supports embedded workflows and RTL (Register-Transfer Level) design automation
Data Analysis & Signal Processing	Log interpretation, waveform analysis, anomaly detection, data summarization, MATLAB/Python script generation	GPT, LLaMA, Mistral, Qwen	Strong reasoning and summarization capabilities; effective for telemetry and mixed-signal datasets
System Design & Modelling	Architecture planning, FSM (Finite State Machine) design, bus/interface specification, SPICE scripting, configuration generation, optimization tasks	GPT, Qwen, Mistral Mixtral	Helps formalize design constraints; accelerates high-level modelling and design-space exploration
Technical Documentation & Compliance	Specification writing, test reports, safety analyses, regulatory documentation, requirement traceability	GPT, LLaMA, T5/FLAN-T5, BERT-family	Excellent at text structuring and consistency; assists in ISO (International Organization for Standardization) / IEC (International Electrotechnical Commission) / DO / FDA (Food and Drug Administration) compliance workflows
Education & Knowledge Transfer	Concept explanation, problem solving, lab instruction, onboarding, internal tool documentation	GPT, LLaMA, Qwen	Supports learning and training; improves accessibility of domain knowledge

CONCLUSION

LLMs have demonstrated substantial potential to enhance workflows in ECE by supporting code generation, system modeling, data interpretation and technical documentation. Their effectiveness stems from advances in Transformer-based architectures and

scalable adaptation techniques that enable alignment with engineering-specific requirements. As these models become increasingly embedded in design and analysis processes, they offer new modalities for interacting with complex hardware–software systems.

Future research should prioritize improving the reliability and factual consistency of LLM outputs, particularly for safety-critical and resource-constrained engineering applications. Robustness to ambiguous prompts, distributional variance and adversarial manipulation remains an important challenge. Continued progress in efficiency: via quantization, sparsity, model compression and hardware co-optimization, will be essential for enabling real-time and embedded deployment. Moreover, deeper integration of LLMs with retrieval mechanisms, simulation environments, formal verification tools and domain-specific data repositories represents a promising direction for enhancing their technical accuracy and trustworthiness. Attention to privacy, security and controlled model behavior will likewise be necessary for industrial adoption.

A further research direction is the design and evaluation of a system capable of automatically determining whether a given technical text has been generated by a large language model or authored by a human, with direct relevance to engineering documentation, code-related narratives and educational content in electrical and computer engineering.

In summary, while LLMs already provide meaningful benefits across ECE domains, their broader and safer deployment depends on advances in reliability, efficiency and rigorous domain adaptation.

REFERENCES

[1] P. Peykani, F. Ramezanlou, C. Tanasescu, and S. Ghanidel, "Large Language Models: A Structured Taxonomy and Review of Challenges, Limitations, Solutions, and Future Directions," *Applied Sciences 2025, Vol. 15, Page 8103*, vol. 15, no. 14, p. 8103, Jul. 2025, doi: 10.3390/APP15148103.

[2] D. Kampelopoulos, A. Tsanousa, S. Vrochidis, and I. Kompatsiaris, "A review of LLMs and their applications in the architecture, engineering and construction industry," *Artificial Intelligence Review 2025 58:8*, vol. 58, no. 8, pp. 250-, May 2025, doi: 10.1007/S10462-025-11241-7.

[3] S. M. Sajjadi Mohammadabadi, B. C. Kara, C. Eyupoglu, C. Uzay, M. S. Tosun, and O. Karakuş, "A Survey of Large Language Models: Evolution, Architectures, Adaptation, Benchmarking, Applications, Challenges, and Societal Implications," *Electronics 2025, Vol. 14, Page 3580*, vol. 14, no. 18, p. 3580, Sep. 2025, doi: 10.3390/ELECTRONICS14183580.

[4] Y. Chang *et al.*, "A Survey on Evaluation of Large Language Models," *ACM Trans Intell Syst Technol*, vol. 15, no. 3, p. 39, Mar. 2024, doi: 10.1145/3641289.

[5] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pp. 3159–3166, 2019, doi: 10.1609/AAAI.V33I01.33013159.

[6] Y. Ye *et al.*, "LLMs4All: A Review of Large Language Models Across Academic Disciplines," Sep. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2509.19580

[7] M. Khan, M. A. Akbar, and J. Kasurinen, "Integrating LLMs in Software Engineering Education: Motivators, Demotivators, and a Roadmap Towards a Framework for Finnish Higher Education Institutes," vol. 17, Mar. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2503.22238

[8] A. Vaswani *et al.*, "Attention Is All You Need," p. 1, Jun. 2017, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/1706.03762

[9] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *Adv Neural Inf Process Syst*, vol. 2020-December, May 2020, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2005.14165

[10] W. X. Zhao *et al.*, "A Survey of Large Language Models," *arXiv preprint arXiv:2303.18223*, Mar. 2023, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2303.18223

[11] M. Usman Hadi *et al.*, "Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects," *Authorea Preprints*, Feb. 2025, doi: 10.36227/TECHRXIV.23589741.V8.

[12] T. L. Khoo, T. S. Lee, S. T. Bee, C. Ma, and Y. Y. Zhang, "A Comparative Review of Large Language Models in Engineering with Emphasis on Chemical Engineering Applications," *Processes 2025, Vol. 13, Page 2680*, vol. 13, no. 9, p. 2680, Aug. 2025, doi: 10.3390/PR13092680.

[13] S. Minaee *et al.*, "Large Language Models: A Survey," Feb. 2024, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2402.06196

[14] Y. Liu *et al.*, "Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment," Aug. 2023, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2308.05374

[15] Q. Zhang *et al.*, "A Survey on Large Language Models for Software Engineering," Dec. 2023, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2312.15223

[16] X. Hou *et al.*, "Large Language Models for Software Engineering: A Systematic Literature Review," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, p. 79, Dec. 2024, doi: 10.1145/3695988.

[17] S. Cruzes, "Revolutionizing optical networks: The integration and impact of large language models," *Optical Switching and Networking*, vol. 57, p. 100812, Oct. 2025, doi: 10.1016/J.OSN.2025.100812.

[18] X. Du, J. Yang, S. Filippi, and B. Motyl, "Large Language Models (LLMs) in Engineering Education: A Systematic Review and Suggestions for Practical Adoption," *Information 2024, Vol. 15, Page 345*, vol. 15, no. 6, p. 345, Jun. 2024, doi: 10.3390/INFO15060345.

[19] Y. Huang *et al.*, "TrustLLM: Trustworthiness in Large Language Models," *arXiv Preprint*, p. 40, Jan. 2024, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2401.05561

[20] M. Bernabei, S. Colabianchi, A. Falegnami, and F. Costantino, "Students' use of large language models in engineering education: A case study on technology acceptance, perceptions, efficacy, and detection chances," *Computers and Education: Artificial Intelligence*, vol. 5, p. 100172, Jan. 2023, doi: 10.1016/J.CAEAI.2023.100172.

[21] Y. Zeng and J. Wu, "A Review of End-to-End Precipitation Prediction Using Remote Sensing Data: from Divination to Machine Learning," Oct. 2025, Accessed: Dec. 31, 2025. [Online]. Available: https://arxiv.org/pdf/2510.22855

[22] M. Hammoud, A. Ai, and D. Acharya, "Don't Pay Attention," Jun. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2506.11305

[23] "Everyday AI: Real-World Applications of Transformer Based Language Models," *International Journal of Computer Trends and Technology*, vol. 73, no. 9, Sep. 2025, doi: 10.14445/22312803/IJCTT-V73I9P103.

[24] S. Nair, Y. S. Rao, R. Shankarmani, and A. Professor, "Assessment of Transformer-Based Encoder-Decoder Model for Human-Like Summarization," Oct. 2024, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2410.16842

[25] S. Nagaraja and K. Y. Chandappa, "Self-attention encoder-decoder with model adaptation for transliteration and translation tasks in regional language," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 14, no. 1, pp. 243–253, Mar. 2025, doi: 10.11591/IJRES.V14.I1.PP243-253.

[26] A. Matarazzo and R. Torlone, "A Survey on Large Language Models with some Insights on their Capabilities and Limitations," Jan. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2501.04040

[27] Y. Zhang *et al.*, "Tensor Product Attention Is All You Need," Jan. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2501.06425

[28] M. Osama, A. Dey, K. Ahmed, and M. A. Kabir, "BeliN: A novel corpus for Bengali religious news headline generation using contextual feature fusion," *Natural Language Processing Journal*, vol. 11, p. 100138, Jun. 2025, doi: 10.1016/J.NLP.2025.100138.

[29] S. Vats *et al.*, "How Do LLMs Work?: A Deep Dive Into Transformer Models," *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/979-8-3373-3785-2.ch004*, pp. 85–105, Jan. 1AD, doi: 10.4018/979-8-3373-3785-2.CH004.

[30] K. Gao *et al.*, "Examining User-Friendly and Open-Sourced Large GPT Models: A Survey on Language, Multimodal, and Scientific GPT Models," Aug. 2023, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2308.14149

[31] P. Zhang, G. Zeng, T. Wang, and W. Lu, "TinyLlama: An Open-Source Small Language Model," Jan. 2024, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2401.02385

[32] A. Q. Jiang *et al.*, "Mixtral of Experts," Jan. 2024, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2401.04088

[33] Y. Chu *et al.*, "Qwen-Audio: Advancing Universal Audio Understanding via Unified Large-Scale Audio-Language Models," Nov. 2023, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2311.07919

[34] A. I. Cristea, E. Walker, Y. Lu, O. C. Santos, and S. Isotani, Eds., "Artificial Intelligence in Education," in *26th International Conference, AIED 2025, Palermo, Italy, July 22–26, 2025, Proceedings, Part II*, in Lecture Notes in Computer Science, vol. 15878. Cham: Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-98417-4.

[35] N. Huynh and B. Lin, "Large Language Models for Code Generation: A Comprehensive Survey of Challenges, Techniques, Evaluation, and Applications," Mar. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2503.01245

[36] M. A. Zadenoori, J. Dąbrowski, W. Alhoshan, L. Zhao, and A. Ferrari, "Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review," Sep. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2509.11446

[37] K. V. Arnautov and D. A. Akimov, "Application of Large Language Models for Optimization of Electric Power System States," *Proceedings of the 2024 Conference of Young Researchers in Electrical and Electronic Engineering, ElCon 2024*, pp. 314–317, 2024, doi: 10.1109/ELCON61730.2024.10468377.

[38] R. E. O. Roxas and R. N. C. Recario, "Scientific landscape on opportunities and challenges of large language models and natural language processing," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 36, no. 1, pp. 252–263, Oct. 2024, doi: 10.11591/IJEECS.V36.I1.PP252-263.

[39] V. Shukla and G. G. Parker, "Building Custom Large Language Models for Industries: A Comparative Analysis of Fine-Tuning and Retrieval-Augmented Generation Techniques," *ICAAEEI 2024 - 1st International Conference of Adisutjipto on Aerospace Electrical Engineering and Informatics: Shaping the Future Work for the Aerospace Technology in Science, Engineering, and Industry in the Disruptive Era*, 2024, doi: 10.1109/ICAAEEI63658.2024.10899129.

[40] S. Chimata, A. R. Bollimuntha, D. K. Devagiri, S. Puligadda, V. P. Kumar S, and V. K. Kishore K, "An Investigative Analysis on Generation of AI Text Using Deep Learning Models for Large Language Models," *International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering, ICSSEEC 2024 - Proceedings*, pp. 7–12, 2024, doi: 10.1109/ICSSEECC61126.2024.10649514.

[41] V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid, "The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities," Aug. 2024, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2408.13296

[42] I. Seregina, P. Lalanda, and G. Vega, "Parameter-Efficient Fine-Tuning for HAR: Integrating LoRA and QLoRA into Transformer Models," Dec. 2025, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2512.17983

[43] J. Jiang *et al.*, "A Survey on Large Language Models for Code Generation," *ACM Transactions on Software Engineering and Methodology*, vol. 37, no. 1, Jun. 2024, doi: 10.1145/3747588.

[44] J. Liu *et al.*, "Large Language Model-Based Agents for Software Engineering: A Survey," Sep. 2024, Accessed: Dec. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2409.02977