ITUJournal
Future and evolving technologies
ITU Journal on Future and Evolving Technologies, Volume 6, Issue 3, September 2025
ARTICLE

Private LLM technology: Security-layer definitions and optimal silicon solutions

Fa-Long Luo[1], Paul Master[1], Darlene Kindler[1]

[1]Cornami, Inc, 1999 S. Bascom Ave, Suite 800, Campbell, California 95008, USA

Corresponding author: Fa-Long Luo, f.luo@ieee.org

The privacy and security concerns related to execution data, model parameters, and processing algorithms have assumed an increasing pivotal role in the rapid development and massive deployment of Large Language Model (LLM) technology. This paper provides an overview of private LLM by addressing its working principles, application scenarios, security requirements, training and inference algorithms, and more importantly, its silicon and chip implementation in order to bring this game-changing technology to real-world products. This paper is organized into five sections. The first section is devoted to the background and motivations for proposing private LLM technology. According to different requirements for privacy and security, in the second section we categorize the proposed private LLM technology into three application scenarios (security levels) and then present the corresponding algorithms related to training and inferences. In order to converge private LLM into optimum silicon implementation, we present the proposed Cornami solution and its comparisons with existing solutions (GPU and ASIC) in terms of power consumption, cost and processing latency in Section 3 and Section 4. In the last section, we make some conclusions and note further discussions.

Keywords – Encryption, LLM, privacy, security, signal processing

1. BACKGROUND AND MOTIVATIONS

Driven by Large Language Model (LLM) and Generative Pre-trained Transformers (GPT) technology, Generative Artificial Intelligence (GenAI) is now finding a use in almost every aspect of industry and society due to its powerful capabilities in extracting, processing and expanding data, information and knowledge. GenAI can help to meet the increasing demands of our digital life in terms of cost, power, capacity, coverage, latency, efficiency, flexibility, compatibility, quality of experience and service [1].

According to OpenAI's publications [2], an LLM could be considered as nonlinear mapping from the input X (query) to output Y (response) by performing the following major processing blocks, shown in Fig. 1, namely, Embedding and Encoding, Multiple Head Attentions, Feed Forward Perceptron, Layer Normalization and Softmax function. In an LLM layer of Fig. 1, Embedding and Encoding is a pre-processing unit which mainly performs the matrix multiplications between the input matrix and the corresponding weight matrices. Multiple Head Attentions perform multiple matrix multiplications according to three weight matrices called Query Matrix, Key Matrix and Value Matrix, respectively. Feed Forward Perceptron layer is a conventional feedforward neural network which has at least one hidden layer. Layer Normalization simply normalizes each of its inputs. Softmax is an activation function that scales numbers/logits into probabilities. As shown in reference [3], it can be seen that the output of an LLM can uniquely be generated by all the pre-determined weight matrices and its inputs. These weight matrices can be obtained during an off-line training stage. During the inference stage, LLM simply performs all the above matrix multiplications and corresponding nonlinear functions such as layer normalization and softmax operations. Different LLMs have different parameter sizes (the total element number of the weight matrices) which mainly depends on the number of attention heads and number of LLM layers. For example, GPT-3 has 96 heads and 96 layers and hence there are about 175 billion parameters (elements of weight matrices) in total.

In terms of training and inference, the application and implementation scenarios of LLM could be mainly categorized into the following three groups, namely; "General LLM", "Fine-Tune LLM", and "Private LLM."

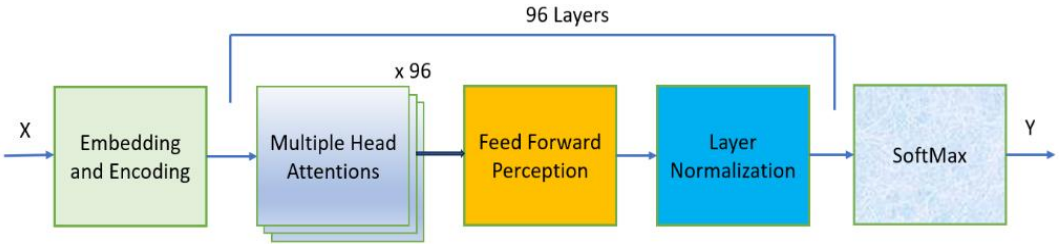

Figure 1 – Illustration of an LLM layer

Figure 2 – Illustration of General LLM

As shown in Fig. 2, General LLM is actually the one that is being most commonly used now where W denotes the entire weight matrix and F(W,X) represents the nonlinear mapping that LLM is to perform.

For the General LLM case, the processing steps include:

- The Client sends their data X to the server which owns LLM parameters.
- The Server performs the LLM operation with X as inputs of LLM as shown in Fig. 1.
- The Server sends the Client the generated output Y which equals to $F(W,X)$.

Meanwhile, taking the existing pre-trained General LLM as a starting point, it is possible to perform processing using very specific knowledge to produce LLMs that are experts in the narrow knowledge domain, which is defined as a Fine-Tune LLM [4]. A Fine-Tune LLM involves taking a pre-existing general model that has been trained on a large dataset, such as a language model like Llama3 and refining it for a specific task or domain. During fine-tuning, the model is further trained on a smaller, domain-specific dataset. This process adapts the model's parameters to the nuances of the target task, improving its performance and making it more capable in handling specific tasks. A Fine-Tune LLM is a cost-effective and efficient way to leverage the knowledge learned by a pre-trained general model while tailoring it to specific applications, reducing the need for extensive training from scratch. Fine-tune LLM allows for rapid development of domain specific AI solutions with high accuracy and applicability. Fig. 3 shows how to perform a Fine-Tune LLM task where ΔW is the difference between the new modified weight matrix (from specific datasets) and the original weight matrix W (from larger datasets).

More specifically, the processing steps of Fig. 3 includes:

- The Client sends their data X to the server which owns LLM parameters and fine-tune parameters.
- The Server performs the LLM operation with X as inputs of Fine-Tune LLM.
- The Server sends the Client the generated output Y which equals to $F(W,X)+F(\Delta W,X)$ or $F(W+\Delta W,X)$.

In the above General LLM and Fine-Tune LLM, both client and server are fully transparent and there is neither privacy or protection nor security among their input data and output data as well as those customized fine-tuning parameters owned by the server. Hence, there are three problems regarding privacy and data protection [5].

(1) From the owner of the fine-tuned LLM models' perspective, the theft of one of these fine-tuned LLM parameters is catastrophic. The LLM can enable competitors to produce products or offer services that compete with the original LLM owners' business or if simply released on the Internet can drive the revenue of the original LLM owners' company to zero. There is a need to prevent the use by unauthorized third parties of a company's fine-tuned LLM.

(2) A client of these LLMs is providing queries to the LLM and receiving results from these queries. The client's queries may contain intellectual property and the answers to these queries may contain new and novel intellectual

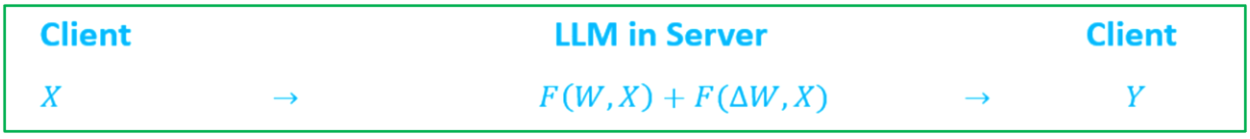

Figure 3 – Illustration of Fine-Tune LLM

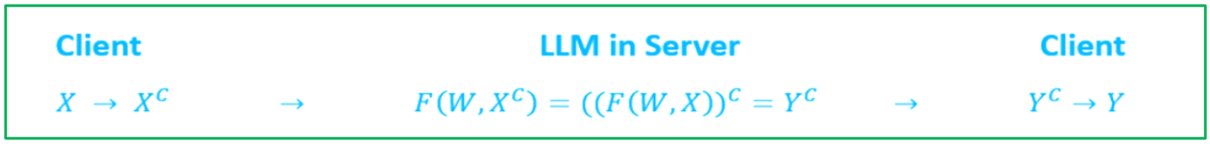

Figure 4 – Illustration of Level-1 Private LLM

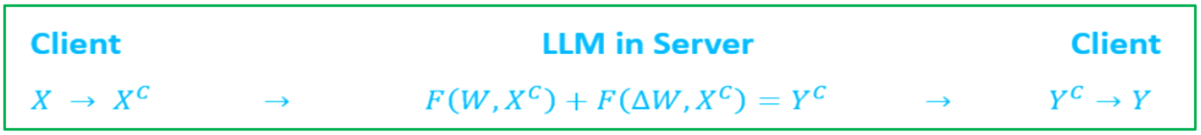

Figure 5 – Illustration of Level-2 Private LLM

property that the owner of the LLM now has access to. There is a need to prevent the leakage of a user's intellectual property into the LLM from the user's queries as well as to protect any new and novel intellectual property in the LLM's responses.

(3) The information that is used for the fine-tuning LLM may be highly sensitive, proprietary, classified or protected under privacy laws such as the European GDPR rules or the US HIPPA rules. There is a need to be able to perform training without exposing the training information during the fine-tuning process.

2. PRIVATE LLMS

It is the goal of the proposed Private LLM technology to attack the above three problems. In theory, this could be accomplished by using Fully Homomorphic Encryption (FHE) approaches [6]. According to different requirements for privacy and security, Private LLM can be grouped into three levels by taking different data format (plaintext or ciphertext) for input query, weights and output responses of an LLM as shown in Table 1 and figures 4-6. It should be noted that superscript "C" denotes the ciphertext format in the rest of this section. The processing steps of Level-1 Private LLM include:

- The Client first encrypts their data X (plaintext) to the ciphertexts X^C by using some schemes to support Fully Homomorphic Encryption (FHE) such as CKKS or tFHE algorithms.
- The Client sends their encrypted data X^C to the server which owns general LLM parameters. These parameters are in plaintext format.
- The Server performs LLM operation with X^C as inputs of the General LLM as shown in Fig. 4.
- The Server sends to Client the generated output Y^C which equals to $F(X^C)$ and also should equal to $(F(X))^C$; that is, the ciphertexts corresponding to the plaintexts $F(X)$.
- The Client decrypts the received ciphertext Y^C to finally get the desired result which equals to $F(X)$.

Likewise, the processing steps of Level-2 Private LLM in Fig. 5 include:

- The Client first encrypts their data X (plaintext) to the ciphertexts X^C with FHE algorithms.
- The Client sends their encrypted data X^C to the server which owns the general LLM parameters and fine-tune parameters which are both still in plaintext format.
- The Server performs the fine-tune LLM operation with X^C being the inputs of the Fine-Tune LLM as shown in Fig. 5.
- The Server sends the Client the generated output Y^C which equals to $F(W, X^C) + F(\Delta W, X^C)$, also should equal to $(F(W, X) + F(\Delta W, X))^C$; that is, the ciphertexts corresponding to the desired output.
- The Client decrypts the received ciphertext Y^C to finally get the desired result Y.

Table 1 – Three levels of Private LLMs

Private LLM	Input (Query) X	Output (Responses) Y	Fine-Tuning Weights ΔW	General Weights W
Level -1	Ciphertexts	Ciphertexts	N/A	Plaintexts
Level- 2	Ciphertexts	Ciphertexts	Plaintexts	Plaintexts
Level- 3	Ciphertexts	Ciphertexts	Ciphertexts	Plaintexts

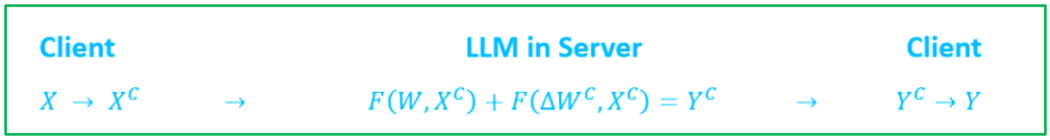

Figure 6 – Illustration of Level-3 Private LLM

As given in Table 1 and Fig. 6, not only input and output are encrypted, but also the fine-tune weights are encrypted by the Server side in Level-3 Private LLM. With this, its processing steps include:

- The Client first encrypts their data X (plaintext) to the ciphertexts X^C with FHE algorithms.
- The Client sends their encrypted data X^C to the server which owns the general LLM parameters in the format of plaintexts and the fine-tune parameters in the format of ciphertexts encrypted by the server itself.
- The Server performs the fine-tune LLM operation with X^C being the inputs as shown in Fig. 6.
- The Server sends the Client the generated output Y^C which equals to $F(W, X^C) + F(\Delta W^C, X^C)$, also should equal to $(F(W, X) + F(\Delta W, X))^C$, that is, the ciphertexts corresponding to the desired output.
- The Client decrypts the received ciphertext Y^C to finally get the desired result Y.

In comparison with General LLM and Fine-Tune LLM in plaintext format, the computational complexity and the corresponding hardware implementation cost and power consumption for Private LLM will be greatly increased mainly because most matrix multiplications are performed in ciphertexts format and also a large amount of extra bootstrapping operations are required in order to reduce the growth of noise incurred by FHE. Table 2 presents a quantitative comparison of two kinds of representative implementation platforms (ASIC and GPU) in terms of the total cost and power consumption in order to generate the desired responses for a complete sequence of tokens with using the parameters setting in GPT-3 [7].

Since the cost and power consumption even for exactly the same processing algorithm will greatly vary with many hardware factors such as the overall architecture, processing units, control and instruction units, data read and write (Cache, DRAM and SRAM), address-generation and process technologies, we propose a new unified metric called "hardware Multiplier-Equivalent (ME)", which could serve as a future standard metric based on taking all the above factors into account. For example, performing a complex-valued multiplication, an ASIC platform would use at least 6 MEs but a GPU platform would use at least 50 MEs because a data-read for multiplication from DRAM into GPU costs at least five times more than the operation of multiplication itself.

The first column of Table 2 is the listing of five LLMs. The second and third column of Table 2 indicate the number of trillion MEs (TME) to be needed by ASIC and GPU in order to generate the desired output for a complete sequence of tokens in GPT-3, respectively. For the case of General LLM inference as an example, 30.5 TME and 336 TME are required by ASIC solution and GPU solution, respectively. For the same inference task, if we use Level-1 Private LLM, 1530 TME and 59, 670 TME are required, which is more than 100 times for General LLM. If we use Level-3 Private LLM, 175,950 TME and 6,719,333 TME are required, which is another 100 times increase. For further illustrations, the fourth column gives the ratio of five LLMs with the General LLM in terms of the number of GPUs. More specifically, if we assume that the total GPU needed to perform General LLM is the unity, 178 GPUs, 190 GPUs, 19,998 GPUs are required for implementing Level-1, Level-2 and Level-3 Private LLM, respectively, which is not practical and even impossible for Level-3 Private LLM. To attack this challenge, we will present our proposed solution and show its effectiveness in the following sections..

3. CORNAMI'S RECONFIGURABLE COMPUTING ARCHITECTURE

As shown In Fig. 7, Cornami has developed and implemented an architecture that combines aspects of both dataflow and reconfigurable computing to stream data through a computational fabric architecture with highly functional computational elements that can dynamically scale over many chips. The computational fabric is represented by one or many custom ASIC chip(s) residing in one or multiple PCIe cards within one or multiple host servers. Each host server has x86 processor(s) running Linux as an interface to the computational fabric. The custom ASICs have several key functional components that are linked by following three types of core communication mechanisms. The first is the adjacent core-to-core which is one core communicating with a physically adjacent core as laid out on the silicon substrate. Adjacent core communication is the most efficient inter-core communication mechanism and takes place via the north, south, east, or west core interfaces. The second is Network-On-Chip (NOC) which generalizes cores to core communication interface where they are NOT side-by-side on the same chip or when cores reside on different chips. The third is PCIe for intra-system communications between the host and PCIe boards.

Table 2 – Cost and power comparisons for five LLMs

Application Scenarios	ASIC Solution (Unit: TME)	GPU Solution (Unit: TME)	Ratio with GPU Solution for General LLM
General LLM	30.5	336.	1.00
Fine-Tune LLM	33.6	369.	1.09
Private LLM Level-1	1,530.	59.670.	178
Private LLM Level-2	1,615.	63, 851.	190
Private LLM Level-3	175,950.	6,719,333.	19,998

There are two versions of computational cores which can dynamically switch from one type to the other.

- A full RISC-V processor with associated SRAM able to execute traditional Control Flow programs as a function representing the computation within a dataflow node.
- Sixteen independently reconfigurable and programmable ALUs, called FracTLcores™, each with associated SRAM supporting multiple simultaneous integers and floating-point computations of up to 128-bits.

This reconfigurable computing architecture allows different functions to also be defined by dynamically changing the topological linkages of processing cores within a computational fabric to achieve superior silicon utilization in terms of application performance, throughput, power consumption, and processing latency. The computational fabric significantly reduces the dependence on memory to store intermediate computational results and exceeds the flexibility and programmability of an FPGA or DSP or GPU while still providing near (ASIC)-level solution performance.

4. PROPOSED SOLUTION FOR PRIVATE LLM

As suggested in the above section, FracTLcore Fabric is a very powerful hardware computing platform to perform extensive matrix multiplications required in General LLM and Private LLM with near zero programming complexity. However, it is still not practical if we simply use FracTLcores to implement Level-3 Private LLM without further reducing its computational complexity because fine-tune weights need to be operated in ciphertext format. To attack this problem, a low-rank adaption (LoRA) algorithm can be used to first reduce the size of the fine-tune weights ΔW where ΔW (dxd) can be generated by two much lower-rank matrices A and B, that is, $\Delta W = A \times B$ as shown in Fig.8. The rank r could be as small as 1 which is 1000 times less than d.

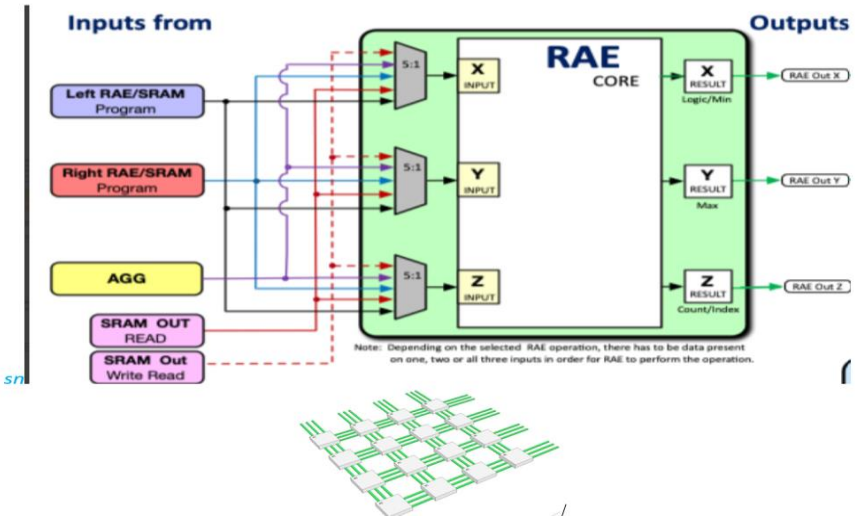

Figure 7 – The schematic diagram of the reconfigurable computing architecture

With this decomposition, instead of encrypting weights ΔW which have the same dimensions and sizes as W, we can first encrypt the small size matrices A and B and then use their encrypted version to replace ΔW^C and further perform the computations in the ciphertext domain as required in all the processing steps of Level-3 Private LLM of Fig. 6. Since the dimensions of matrices A and B are much smaller, the corresponding computational complexity for ciphertext would be as low as the ones for plaintexts, which means the reduction in 100 times can be achieved by the proposed LoRA based algorithm. Fig. 9 further illustrates the data-flow of the inference stage of LoRA based Leve-3 Private LLM.

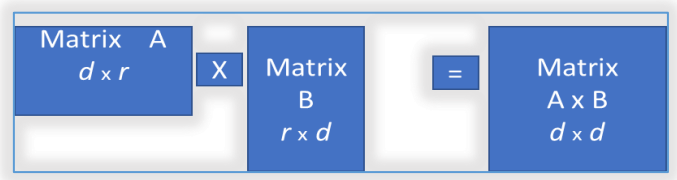

Figure 8 – Low-rank decomposition of fine-tune weights

Using the same metric unit (TME) and the same ratio definition as those in Table 2, Table 3 presents a quantitative comparison of the GPU platform to Cornami's platform in terms of the total cost and power consumption in order to generate the desired responses for a complete sequence of tokens using the parameters setting for GPT-3. It can be seen from the fifth column that Cornami's platform costs are only about one tenth of the GPU platform costs for performing the same General LLM inference task. For performing a Level-1 Private LLM inference task, the GPU platform costs 179 times but Cornami's platform costs only 3.03 times. As shown in the last row for performing a Level-3 Private LLM inference task by using a LoRA-based algorithm, the GPU platform costs 196 times but Cornami's platform costs only 3.51 times, which suggests that the GPU platform is still not practical. Instead, Cornami's platform serves as a feasible and practical solution for the deployment of all three levels Private LLM into real-world applications [8].

5. CONCLUSIONS

According to different requirements for privacy and security of both a client's data and a server's model, three levels (layers) of Private-LLM have been defined in this paper. For a Level-1 Private LLM scenario, all the client-related data (such as query and inference output) are fully encrypted, but the server holds public LLM models in a plaintext format. In a Level-2 Private LLM scenario, everything is the same as for Level-1 except that the server also owns, in a plaintext format, some fine-tune models to be used for specific-domain service tasks. In a Level-3 Private LLM scenario, not only the client data but also the fine-tune models that the server owns are encrypted, which could offer the highest protection between the client and server.

Through quantitative results in terms of computational complexity, memory accessing and processing latency, we have shown that trillions of matrix multiplications in the ciphertext-to-ciphertext format or cyphertext-to-plaintext format are required for all these Private LLM tasks, which means that it is not practical and, even not possible if the current GPU platforms are used to perform all operations in Private LLM.

In order to attack the above implementation challenge, the Cornami solution has been proposed and reported in this paper. Having the features with minimum data movement, near-zero programming complexity and reconfigurable capability, Cornami FracTLcore Fabric is a very powerful hardware computing platform to perform extensive matrix multiplications and nonlinear functions in both plaintext and ciphertext formats required in Private LLMs. These operations and computations include Plaintext-Ciphertext Matrix Multiplication (PCMM) and Ciphertext-Ciphertext Matrix Multiplication (CCMM), KV cache compression, mixture-of-experts routing, rotary positional embedding and load balancing as well as the related bootstrapping and key switching [9]. It is hoped that this paper could offer some useful insights and perspectives for privacy and security standardization, technical regulations and specifications as well as practical implementations and deployment of LLM technology.

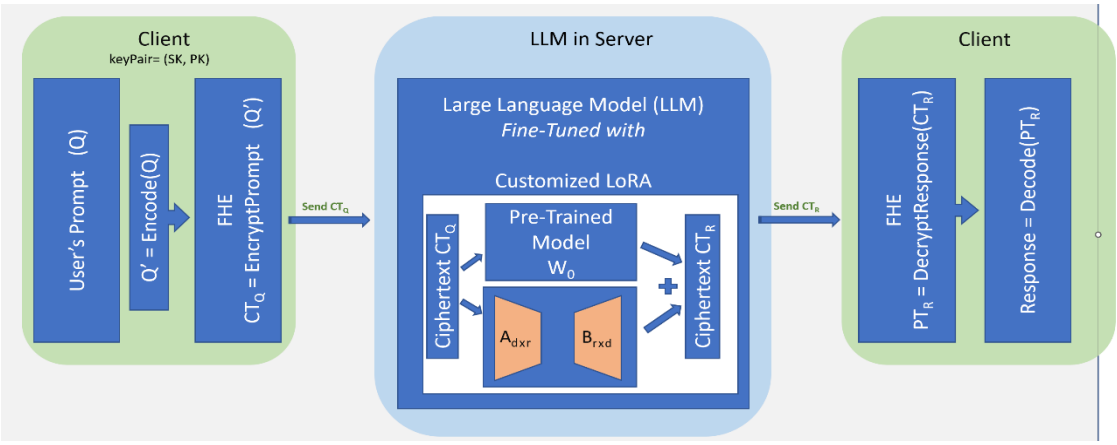

Figure 9 – Inference data-flow of LoRA Based Level-3 Private LLM

Table 3 – Comparison between GPU and FracTLcores

Application Scenarios	GPU Solution (Unit: TME)	Cornami's Solution (Unit: TME)	GPU Ratio with the One for General LLM	Cornami's Ratio with GPU Solution for General LLM
General LLM	336.	35.3	1.00	0.105
Fine-Tune LLM	369.	35.8	1.09	0.106
Private LLM Level-1	59.670.	1,009	179	3.03
Private LLM Level-2	63, 851.	1,073	190	3.19
Private LLM Level-3	6,719,333.	110,937	19,998	330
Private LLM Level-3 (with Cornami's Algorithm)	65,980.	1179.	196	3.51

REFERENCES

[1] "Generative AI: A Game-Changer Society Needs to be Ready for". World Economic Forum, 9 January 2023.

[2] "Improving Language Understanding by Generative Pre-Training," OpenAI, 11 June 2018.

[3] Fa-Long Luo, "Machine Learning for Future Wireless Communications", IEEE-Wiley, USA, 2021

[4] Edward Hu, etc. "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685 [cs.CL], 2021.

[5] Pengfei Liu, etc. "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing", ACM Computing Surveys, Vol. 55, No.9, pp 1–35, 2023.

[6] Dongwoo Kim and Cyril Guyot, "Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption," IEEE Transactions on Information Forensics and Security, Vol. 18, 2023, pp. 2175-2183.

[7] Yi Sheng, etc, "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU," arXiv:2303.06865, 2023

[8] "DeepSeek-V3 Technical Report", https://github.com/deepseek-ai/DeepSeek-V3., 2025

AUTHORS

FA-LONG LUO is an IEEE Fellow. He is the former Chief Scientist of Micron Technology Inc., USA and has 39 years academic, industry and research experience in adaptive signal processing, wireless, neural networks and optimized silicon convergence of application algorithms, processing platforms and memory accesses. Dr. Luo has published eight books and more than 100 technical papers in related fields. Dr. Luo has also contributed 108 patents/inventions, which have successfully resulted in a number of new or improved commercial products in mass production. Dr. Luo has served as a technical Board Member of IEEE Signal Processing Society (SPS) and an IEEE Fellow Committee member. Currently, he is also the representative of Signal Processing Soceity in IEEE TAB Committee on Standards and a Senior Editor of IEEE SP Magazine.

PAUL MASTER is now serving as the Chief Technology Officer of Cornami, Inc: a Silicon Valley cutting-edge company delivering optimum real-time and intelligent computing on encrypted data. Graduating from the University of California at Berkeley, Paul has over 30 years' experience in high tech development and leadership in both large corporations and start-up company environments. His previous roles included positions as CEO, CTO, VP of Engineering, VP of Systems Engineering, VP of Architecture, and Director of Marketing among others. He has held positions in multiple startups, as well as larger companies such as The Boeing Company, ARGOSystems, and ACER America. As a well-recognized visionary and influential technical leader in the industry, Paul has over 100 patents issued or pending, over 35 professional publications and 13 first pass ASIC successes. He has been the invited keynote speaker and expert speaker at a variety of technical conferences.

DARLENE KINDLER has extensive experience in consumer electronics and semiconductor products. She has repeatedly focused on emerging technologies and products in multiple companies beginning with Nintendo, who was re-entering the then, nascent video game industry – 3dfx Interactive, a 3D graphics chip – CHIPs & Systems, an adaptive chip – AdscapeMedia, a dynamic in-game advertising company, and others. In her role at Nintendo, she was one of the key people credited with establishing an international effort in Europe by building out the distribution network. Ms. Kindler went on to establish a wholly owned subsidiary in the U.S. for a Japanese video game company rolling out a number of products successfully. In tune with her entrepreneurial spirit, Ms. Kindler took on the role of Vice President, Marketing/Third Party for a startup company called 3dfx Interactive. 3dfx was in the business of designing the first 3D graphics controller for the PC. She developed and supervised all aspects of the developer program as well as the additional responsibility for marketing, where she executed a branded retail program that resulted in 3dfx winning the Interactive Game Industry award for "Best HardwareMarketing". Ms. Kindler moved on to a new company in an emerging industry as VP Publishing for Adscape Media, a technology networking company focused on dynamic in-game advertising. Google acquired the company.