Resilient Artificial Intelligence Architecture

Dmitry Lande [1], Leonard Strashnoy [2]

[1] ORCID: 0000-0003-3945-1178
National Technical University of Ukraine – Igor Sikorsky Kyiv Polytechnic Institute
[2] ORCID: 0009-0008-5575-0286
University of California, Los Angeles (UCLA)

Abstract – Modern Large Language Models (LLMs) have revolutionized AI, yet they suffer from significant drawbacks: exorbitant energy consumption, centralized infrastructure vulnerabilities, and escalating computational costs with task complexity. This paper presents a resilient AI architecture based on a distributed swarm of Small Language Models (SLMs) as a compelling alternative. By decomposing complex tasks into subtasks handled by specialized SLMs, we achieve decentralized computation, enhanced energy efficiency, and superior survivability. Mathematical formalization demonstrates that SLMs exhibit linear cost growth, drastically lower than the exponential increase in LLM costs. Our analysis, supported by practical examples and comparisons to centralized systems like OpenAI, Microsoft Azure, reveals that SLM networks offer substantial advantages in reliability, scalability, and cost-effectiveness. Key findings include the inherent resilience of distributed SLMs to individual failures and their ability to dynamically adjust resources in response to changing demands. This study concludes that for the majority of practical applications, a distributed swarm of SLMs provides a more sustainable, robust, and economically viable solution, marking a significant shift from monolithic LLM architectures to a more adaptive and efficient paradigm for AI system design. This approach ensures resilience by decentralizing computational resources, enabling collective intelligence, and enhancing adaptability and survivability, ultimately concluding that a network of SLMs offers a more economical, scalable, and resilient solution than a single LLM.

Keywords: resilient AI, distributed intelligence, small language models, decentralized computing, energy efficiency, adaptability, cybersecurity, survivability.

Introduction

Large Language Models (LLMs) have undoubtedly achieved groundbreaking results in a variety of complex AI tasks, including natural language processing, code generation, and intricate reasoning. However, their impressive capabilities come at a steep price. These models are fundamentally dependent on massive computational resources, leading to several critical issues that threaten their sustainability and reliability. Firstly, the sheer scale of LLMs translates into exorbitant energy consumption, contributing significantly to carbon footprints and operational costs. Secondly, their centralized nature makes them vulnerable to single points of failure; if the central server or data center goes down, the entire system fails. Thirdly, the risk of catastrophic failure due to software glitches, cyberattacks, or hardware malfunctions is a constant concern. To address these significant challenges and build more robust AI systems, a shift towards a distributed paradigm is essential. Specifically, we propose a transition to a distributed system of Small Language Models (SLMs). This transition, however, is not straightforward and hinges on the successful implementation of effective strategies for task decomposition – breaking down complex problems into manageable subtasks – and result aggregation – combining the outputs of individual SLMs into a coherent whole. Only with these well-designed mechanisms in place can we ensure both computational efficiency and system-wide resilience, making the distributed SLM approach a viable and superior alternative to monolithic LLMs.

The objective of this work is to propose a resilient AI framework based on a distributed of SLMs, focusing on the mechanisms of task decomposition, distributed computation, and result aggregation. We also analyze the survivability of such systems under adverse conditions. Tasks:

1. Describe the concept of transitioning from LLMs to SLMs using task decomposition and result aggregation.
2. Develop a mathematical model for task decomposition and aggregation in a distributed system.
3. Compare the proposed approach with traditional LLM-based systems.
4. Highlight the role of resilience in maintaining system functionality during adverse events.
5. Provide practical examples with detailed mathematical analysis.

Recent studies have explored various aspects relevant to our research:

– Research on LLMs [highlights their capabilities but also discusses the challenges of resource centralization and energy consumption. In work [1], the article addresses the issue of reducing energy consumption in data centers running LLMs. A data center model is proposed that utilizes a cost-based scheduling framework for dynamically allocating LLM tasks across hardware accelerators with different levels of energy efficiency and computational power.
– Studies on SLMs focus on their efficiency and specialization but lack comprehensive frameworks for integrating them into distributed systems. The article [2] examines the transition from classical transformers to alternative architectures, such as state space models (SSM), which provide better scalability and efficiency, the growing role of smaller models (SLM), their democratization, and adaptation to specific industries.
– Works on decentralized AI [3] emphasize the benefits of distributing computational resources but often overlook the specifics of task decomposition and result aggregation. This study proposes an energy-efficient task distribution mechanism for blockchain Proof of Authority (POA) consensus, utilizing the Dynamic Voltage and Frequency Scaling (DVFS) technique to optimize energy consumption and performance in the Cloud Industrial Internet of Things (CIIoT). It ensures the allocation of computing resources through artificial intelligence and neural networks, with simulation results demonstrating reduced energy consumption and improved efficiency compared to traditional methods.
– Applications of distributed intelligence in robotics [4] provide insights into collective problem-solving, which can be adapted to AI systems. Distributed intelligence is a powerful approach to solving optimization and decision-making problems. This review examines its principles, algorithms, applications in various fields, performance evaluation, scalability, robustness, and interpretability.
– Techniques for decomposing complex tasks into simpler subtasks are well-documented in operations research, offering a foundation for applying similar principles to AI systems. The article [5] examines the use of large language models for automating cyber investigations and digital forensics, emphasizing the importance of transparency and data security, proposing local processing with smaller models that can be enhanced through cognitive augmentation, demonstrating significant improvements and opening prospects for further research.

Despite these advancements, there is a need for a unified framework that integrates all these elements into a resilient AI system.

Concept of Resilient AI Using a Geographically Distributed Swarm of Small Language Models

A resilient AI system consists of numerous specialized SLMs working collaboratively to solve complex problems. Each SLM operates locally, consuming minimal resources while focusing on specific tasks. By leveraging distributed intelligence, the system achieves:

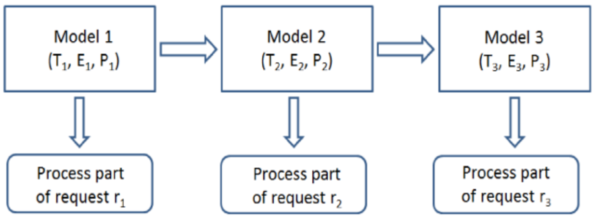
– Decentralization: Reduces reliance on centralized resources, mitigating risks associated with single points of failure.
– Energy Efficiency: Distributes computational loads, minimizing energy consumption per task.
– Survivability: Ensures continued operation even if some components fail, thanks to redundancy and collective problem-solving.

To transition from LLMs to SLMs, complex tasks must be decomposed into smaller subtasks, each of which can be handled by a specialized SLM. This process involves:

1. Identifying the key components of the task.
2. Assigning each component to an appropriate SLM based on its expertise.

3. Ensuring efficient communication between SLMs for result aggregation.

Distributed Swarm Architecture:


Model 1
(T_1, E_1, P_1)
Model 2
(T_2, E_2, P_2)
Model 3
(T_3, E_3, P_3)
Process part of request r_1
Process part of request r_2
Process part of request r_3

Each node in the swarm specializes in a subset of tasks, communicating with others to form a cohesive whole. This design mirrors biological swarms, where simple agents collectively exhibit sophisticated behaviors.

Once the subtasks are processed by individual SLMs, their results must be aggregated to form a coherent solution. This can be done using techniques such as weighted averaging, voting, or hierarchical combination, depending on the nature of the task.

Task Decomposition as a Bi-Criteria Optimization Problem

Let T be a complex task decomposed into mmm subtasks $T_1, T_2, \ldots, T_m$. Each subtask T_k is assigned to a small language model M_i based on its parameters:

- C_{ik} — compatibility score between SLM_i and subtask k, measuring how well the given SLM can handle the subtask.
- T_i — processing time of SLM_i, representing the computational delay before producing a result.
- E_i — energy consumption of SLM_i, indicating the power required to execute the subtask.

The optimization problem aims to minimize both total energy consumption and total processing time:

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik} E_i \to \min$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik} T_i \to \min$$

Subject to constraints:

$$\begin{cases} \sum_{i=1}^{n} x_{ik} = 1, \quad \forall k \in \{1, \ldots, m\} \\ \sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik} C_{ik} \ge P_{\min} \\ x_{ik} \in \{0,1\} \end{cases}$$

Explanation of constraints:

1. The first constraint ensures that each subtask is assigned to exactly one *SLM*.
2. The second constraint guarantees that the overall accuracy of task execution meets or exceeds the required threshold $P_{\min}$.
   - $P_{\min}$ represents the minimum required accuracy for completing the full task.
   - It is computed as a function of subtask completion quality, where C_{ik} values indicate how well each *SLM* performs a given subtask.
   - For example, in a classification task, $P_{\min}$ may refer to an accuracy percentage (e.g., 90% classification accuracy). In a text generation task, it may represent the semantic similarity between generated and expected outputs.
3. The binary constraint ensures that a subtask is either assigned to an *SLM* $(x_{ik} = 1)$ or not assigned ($x_{ik} = 0$).

This bi-criteria optimization formulation ensures that subtasks are distributed among models in a way that minimizes both energy consumption and processing time while maintaining the required accuracy level.

Result Aggregation

After processing, the outputs from each *SLM* are combined using a weighted aggregation function:

$$R = \sum_{i=1}^{n} w_i r_i,$$

where:

- r_i — result produced by *SLM* M_i.
- w_i — weight assigned to *SLM* M_i, which is determined based on multiple factors such as accuracy P_i and relevance R_i.

The weight assignment follows a normalization criterion:

$$w_i = \frac{f\left(P_i, R_i\right)}{\sum_{j=1}^{n} f\left(P_j, R_j\right)},$$

where $f\left(P_i, R_i\right)$ is a weighting function that balances the impact of accuracy P_i and relevance R_i in determining the final aggregated result.

Interpretation:

1. The function $f\left(P_i, R_i\right)$ can take different forms depending on the task.
   - For classification problems, it may be a function of confidence scores or precision-recall measures.
   - For text generation tasks, it could incorporate measures such as semantic similarity or fluency scores.
2. The normalization ensures that the weights sum to 1, preventing any single SLM from dominating the final result.
3. This approach guarantees that models with higher accuracy and relevance contribute more to the final outcome, ensuring robustness in aggregated predictions.

Centralized systems, such as those hosted on Microsoft Azure, rely heavily on a single point of failure. In contrast, a swarm of SLMs distributes computation geographically, reducing the risk of systemic failure. For example, consider a disaster scenario where a regional data center hosting an LLM is compromised. A swarm of SLMs deployed globally can continue functioning, ensuring uninterrupted service.

Comparison of the Efficiency of LLM and SLM Networks

With the advancement of artificial intelligence and machine learning technologies, the question arises: is it more effective to use one large universal model to solve all tasks, or is it better to build a network of small specialized models, each optimized for a specific task? It is known that large models, such as universal transformers, possess significant computational power; however, their energy efficiency and economic feasibility may be considerably worse compared to a network of small models. In this section, we will analyze why a network of SLM is more energy-efficient, reliable, and cost-effective compared to a LLM.

Formulation of Costs and Energy Efficiency

To gain a clearer understanding of the computational costs and energy consumption in large models and networks of small models, let us introduce several mathematical notations:

- Computation costs for a LLM: denoted as C_{LLM} . These are the costs associated with executing a task using a large, general-purpose model that must handle a wide range of tasks, including processing input data, solving specific subproblems, and generating the final result.
- Computation costs for a network of SLM: denoted as C_{SLM} , which represents the sum of costs for each small, specialized model SLM_k . Each of these models is optimized to solve a specific subproblem within the overall task, allowing computational resources to be distributed across different models.

Mathematically, this can be written as follows:

For the large model:

$$C_{LLM} = f_{LLM}\left(n\right),$$

where n is the number of subtasks or the complexity of the task, and $f_{LLM}\left(n\right)$ is a function describing the computational costs, which increase nonlinearly with the growth of complexity.

For the network of small models:

$$C_{SLM} = \sum_{k=1}^{n} C_{SLM_k},$$

where m is the number of small models, and C_{SLM_k} is the computational cost for each individual model.

Proof of Cost Inequality

Let's compare the computational costs for a large model and a network of small models. The total computational cost for a large model can be expressed as:

$$C_{SLM} \geq \sum_{k=1}^{m} C_{SLM_k} = C_{SLM}.$$

The proof of this comparison is based on the following factors:

- Specialization of small models: Each small model specializes in performing a single specific task. This allows for optimization of computational costs for each model, as it operates only on a limited set of data and operations, reducing computational complexity.
- Scalability and distributed computing: When using a network of small models, tasks can be distributed among the models, which lowers overall computational costs. Additionally, each small model can operate independently of others, reducing the overall system load.
- Increasing costs for large models: Since a large model must handle all types of tasks, computational costs increase non-linearly with the complexity of the task. This is because the large model must utilize more parameters and more complex algorithms for each new task.

For these reasons, the computational costs for a large model will always be greater than or equal to the costs for a network of small models.

Examples of Result Aggregation in SLMs

1. Sentiment Classification

Task: Determine the overall sentiment of a text based on analysis from multiple SLMs.

Aggregation Function f : Weighted Average

$$R = \sum_{i=1}^{n} w_i r_i,$$

where:

- r_i is the sentiment score from model SLM_i (-1 for negative, 0 for neutral, +1 for positive);
- w_i is the weight of each SLM, proportional to its accuracy P_i.

Example Calculation:

Suppose three SLMs provide the following sentiment scores:

- $SLM_1 : r_1 = 1$, weight $w_1 = 0.5$;
- $SLM_2 : r_2 = -1$, weight $w_2 = 0.3$;
- $SLM_3 : r_3 = 1$, weight $w_3 = 0.2$.

The final sentiment score is calculated as follows:

$$R = (0.5 \times 1) + (0.3 \times (-1)) + (0.2 \times 1) = 0.5 - 0.3 + 0.2 = 0.4.$$

Since $R > 0$, the overall sentiment is classified as positive.

2. Machine Translation

Task: Select the best translation of a sentence among results from multiple SLMs.

Aggregation Function f : Maximum Weight Selection

$$R = r_{\text{argmax}}(w_i),$$

where:

- r_i is the translation provided by SLM_i;
- w_i is the quality score of the translation (e.g., BLEU score or semantic similarity).

Example Calculation:

Suppose three SLMs provide the following translations:

- SLM_1: "The weather is great today." $(w_1 = 0.85)$;
- SLM_2 : "Today, the weather is wonderful." ($w_2 = 0.92$);
- SLM_3: "It's a nice day outside." $(w_3 = 0.78)$.

Since SLM_2 has the highest weight $\left(w_2 = 0.92\right)$, its translation is selected:

$R = r_2 =$ "Today, the weather is wonderful."

3. Anomaly Detection

Task: Determine the overall anomaly score for a dataset.

Aggregation Function f : Weighted Average

$$R = \sum_{i=1}^{n} w_i r_i,$$

where:

- r_i is the anomaly score detected by SLM_i (ranging from 0 to 1, where 1 represents a severe anomaly);
- w_i is the weight of each model (based on its accuracy P_i).

Example Calculation:

Suppose three SLMs produce the following anomaly scores:

- $SLM_1 : r_1 = 0.9$, weight $w_1 = 0.4$;
- $SLM_2 : r_2 = 0.2$, weight $w_2 = 0.3$;
- $SLM_3 : r_3 = 0.7$, weight $w_3 = 0.3$.

The final anomaly score is computed as follows:

$R = \left(0.4 \times 0.9\right) + \left(0.3 \times 0.2\right) + \left(0.3 \times 0.7\right) = 0.63.$
If the anomaly threshold is set at 0.6, the system triggers an anomaly alert.

For classification tasks (e.g., sentiment analysis, anomaly detection), weighted averaging is an effective aggregation method. For selection-based tasks (e.g., machine translation, image recognition), choosing the best option based on weight is more suitable. Hybrid approaches can be used in cases requiring both ranking and averaging.

Technical Challenges and Solutions

1. Synchronization Between Models

Challenge:
In distributed systems, ensuring that the results from different SLMs are synchronized and consistent is critical. Without proper synchronization, the aggregated results may be inconsistent or erroneous, especially when tasks require combining outputs from multiple models.

Solutions:

Weighted Aggregation: Use weighted averaging or voting mechanisms to combine results from different SLMs. Each model's output is assigned a weight based on its accuracy or relevance.

$$R = \sum_{i=1}^{n} w_i r_i,$$

where w_i is the weight of the *i*-th SLM, and r_i is its result.

Timestamping: Implement timestamping to order the results from different SLMs, ensuring that the most recent or relevant data is used in aggregation.

Consensus Algorithms: Use consensus algorithms (e.g., Paxos or Raft) to ensure that all SLMs agree on the final result, especially in cases where multiple models contribute to the same task.

2. Managing a Large Number of SLMs

Challenge:
Managing hundreds or thousands of SLMs in a distributed network can be complex. Efficiently distributing tasks, avoiding overloading individual nodes, and ensuring balanced resource utilization are key challenges.

Solutions:

Load Balancing: Implement load-balancing algorithms (e.g., round-robin, least connections, or priority-based scheduling) to distribute tasks evenly across SLMs.

$$\sum_{i=1}^{n} C_{ik} \cdot x_{ik} \le B_i,$$

where C_{ik} is the computational cost of task T_k on SLM_i and B_i is the capacity of SLM_i .

Dynamic Task Scheduling: Use dynamic scheduling algorithms that can adapt to the current load and availability of SLMs in real-time.

Task Queues: Implement task queues to manage incoming tasks and assign them to available SLMs based on their current workload and specialization.

3. Communication Overhead

Challenge:
In distributed systems, communication between

SLMs can introduce significant overhead, especially if the network is large or geographically dispersed. High latency or bandwidth limitations can slow down the system.

Solutions:

Edge Computing: Perform computations locally on edge devices to minimize the need for data transmission over long distances.

Data Compression: Compress data before transmission to reduce bandwidth usage and improve communication efficiency.

Routing Optimization: Use optimized routing algorithms to minimize communication delays and ensure efficient data transfer between SLMs.

$$C_{comm} = \sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} \cdot v_{ij},$$

where d_{ij} is the distance between SLM_i and SLM_j, and v_{ij} is the volume of data transmitted.

4. Fault Tolerance

Challenge:
In a distributed network, the failure of one or more SLMs can disrupt the system's functionality. Ensuring that the system remains operational despite individual node failures is crucial.

Solutions:

Redundancy: Duplicate critical tasks across multiple SLMs to ensure that if one fails, others can take over.

Recovery Mechanisms: Implement automatic recovery mechanisms that can redistribute tasks from failed SLMs to healthy ones.

Failure Detection: Use heartbeat mechanisms or monitoring tools to detect failures in real-time and trigger recovery processes.

$$P_{survive} = 1 - p^{m},$$

where p is the probability of a single SLM failing, and m is the number of SLMs in the network.

5. Scalability

Challenge:
As the number of SLMs grows, the system must scale efficiently without compromising performance or increasing complexity.

Solutions:

Modular Architecture: Design the system with a modular architecture, allowing new SLMs to be added or removed without disrupting the entire network.

Decentralized Control: Use decentralized control mechanisms to avoid bottlenecks and ensure that the system can scale horizontally.

Resource Allocation Algorithms: Implement algorithms that dynamically allocate resources based on the current demand and availability of SLMs.

6. Security and Privacy

Challenge:
Distributed systems are more vulnerable to security threats, such as data breaches or unauthorized access. Ensuring data privacy and security is a major concern.

Solutions:

Encryption: Encrypt data both in transit and at rest to protect it from unauthorized access.

Access Control: Implement strict access control mechanisms to ensure that only authorized SLMs can access sensitive data.

Blockchain for Data Integrity: Use blockchain technology to ensure data integrity and traceability in distributed AI systems.

While the use of Small Language Models (SLMs) in distributed AI systems offers significant advantages in terms of resilience, energy efficiency, and scalability, it also introduces several technical challenges. These challenges include synchronization between models, managing a large number of SLMs, communication overhead, fault tolerance, scalability, and security. However, by employing advanced algorithms, optimization techniques, and robust system design principles, these challenges can be effectively addressed. This ensures that SLM-based systems remain reliable, efficient, and adaptable for a wide range of practical applications.

Future Perspectives:

1. Scalability, expanding the swarm to global scales could enable real-time robotics and other autonomous systems.
2. Integrating SLMs into autonomous systems, such as smart cities or IoT

networks, would enhance their independence and reliability.
3. Combining distributed SLMs with traditional LLMs could balance flexibility and performance, creating hybrid AI systems suited for diverse applications.

Conclusions

The proposed framework for a resilient AI system, structured around a distributed swarm of Small Language Models, represents a transformative shift in how we conceive and deploy artificial intelligence. Moving beyond the conventional reliance on monolithic Large Language Models, this distributed approach unlocks unprecedented levels of performance, reliability, and adaptability. These enhancements are particularly crucial in scenarios that demand robustness, efficiency, and scalability, such as in real-time autonomous systems, large-scale data processing, and critical infrastructure management.

A fundamental advantage of employing SLMs lies in the strategic decentralization of computational resources. Unlike LLMs, which necessitate substantial and often cost-prohibitive centralized infrastructure, and whose computational demands escalate exponentially with the complexity of the task, SLMs operate in a distributed manner, markedly improving processing efficiency and enabling seamless scalability. This distributed architecture not only leads to a significant reduction in energy consumption per computational task but also fosters an innate resilience against systemic failures. In a scenario where individual models experience failure, the system as a whole remains operational, with other nodes automatically compensating and ensuring uninterrupted functionality. This resilience is a critical differentiator, particularly in environments where continuous operation is paramount.

Our mathematical analysis unequivocally demonstrates the economic and operational benefits of SLMs. The computational and energy costs associated with distributing tasks across a swarm of specialized SLMs are consistently and substantially lower than those required by a single, large LLM. The linear scaling of energy demands in SLM systems, contrasted with the exponential scaling observed in LLMs, positions the distributed SLM framework as a far more sustainable and economically viable solution for long-term AI deployments. Moreover, the specialization of individual SLMs for particular tasks further optimizes resource utilization, translating into tangible reductions in operational costs when compared to the generalized approach of relying on a single, all-encompassing LLM. For example, deploying a specialized SLM for real-time language translation in a customer service setting would require significantly less processing power and energy than routing all translation requests through a large, general-purpose LLM, illustrating the practical cost savings.

Task decomposition and result aggregation are integral to harnessing the full potential of SLM-based systems. The strategy of breaking down intricate tasks into smaller, more manageable components, which can be processed concurrently by specialized models, dramatically reduces processing time and enhances overall system flexibility. The carefully managed aggregation of results from these specialized models yields highly accurate and nuanced outputs, a critical feature when dealing with complex or highly variable tasks where precision is paramount. In areas such as medical diagnosis or financial forecasting, the accuracy and reliability provided by the SLM approach can be life-altering.

In comparison with traditional centralized systems, exemplified by platforms such as Microsoft Azure's cloud infrastructure, the SLM network architecture exhibits enhanced robustness and adaptability. By mitigating the inherent risks linked to single points of failure and providing dynamic resource adjustments to match fluctuating demands, SLMs offer a more reliable and efficient solution. The inherent scalability and flexibility of the swarm-based architecture guarantee that AI systems can adapt seamlessly to changing operational conditions. This adaptability ensures that AI deployments are not only efficient but also prepared for future growth and changing requirements.

In conclusion, the strategic shift from LLMs to SLMs is not merely an incremental improvement but a fundamental paradigm change that addresses the core challenges of modern AI systems. By prioritizing resilience, scalability, and energy efficiency, the distributed SLM model stands as a robust and sustainable alternative. For the vast majority of practical applications, especially those where reliability, adaptability, and cost-effectiveness are critical factors, a distributed swarm of SLMs offers a compelling and superior alternative. Through the thoughtful orchestration of specialized models, decentralized computational resources, and optimized task management, the SLM framework effectively resolves many of the constraints associated with LLMs. This approach fosters the creation of more sustainable, robust, and ultimately more effective AI systems, driving innovation and ensuring that AI technologies are accessible and beneficial across a wider range of applications.

References

[1] Wilkins, G., Keshav, S., & Mortier, R. (2024). Hybrid Heterogeneous Clusters Can Lower the Energy Consumption of LLM Inference Workloads. The 15th ACM International Conference on Future and Sustainable Energy Systems, 506–513. https://doi.org/10.1145/3632775.3662830

[2] Kamath, U., Keenan, K., Somers, G. and Sorenson, S., 2024. LLMs: Evolution and New Frontiers. In Large Language Models: A Deep Dive: Bridging Theory and Practice (pp. 423-438). Cham: Springer Nature Switzerland.

[3] Javadpour, A., Sangaiah, A.K., Zhang, W., Vidyarthi, A. and Ahmadi, H., 2024. Decentralized AI-based task distribution on blockchain for cloud industrial internet of things. Journal of Grid Computing, 22(1), p.33.

[4] Zangana, Hewa Majeed, Zina Bibo Sallow, Mohammed Hazim Alkawaz, and Marwan Omar. "Unveiling the collective wisdom: A review of swarm intelligence in problem solving and optimization." Inform: Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi 9, no. 2 (2024): 101-110.

[5] Bergner, B., Skliar, A., Royer, A., Blankevoort, T., Asano, Y., & Bejnordi, B. E. (2024). Think big, generate quick: Llm-to-slm for fast autoregressive decoding. arXiv preprint arXiv:2402.16844.