www.ijsrnsc.org

Available online at www.ijsrnsc.org

IJSRNSC

Volume-12, Issue-1, February 2024
Research Paper

Int. J. Sc. Res. in Network Security and Communication

E-ISSN:2321-3256

EDeLeaR: Edge-based Deep Learning with Resource Awareness for Efficient Model Training and Inference for IoT and Edge Devices

Mansoor Farooq[1*], Mubashir Hassan Khan[2]

[1]Dept. of Management Studies, University of Kashmir, Srinagar, India
[2] Dept. of Computer Applications, Cluster University of Srinagar, Srinagar, India

[*]*Corresponding Author: mansoor.msct@uok.edu.in, Tel.: +91-8899904968*

Received: 21/Dec/2023, Accepted: 20/Jan/2024, Published: 29/Feb/2024

***Abstract*—**Deep learning has emerged as a powerful technique for processing and extracting insights from complex data. However, the resource-constrained nature of edge devices poses significant challenges to the deployment of deep learning models at the network edge. This research proposes a novel algorithm called EDeLeaR, which stands for Edge-based Deep Learning with Resource-awareness, to enable efficient model training and inference in edge computing environments. EDeLeaR leverages adaptive resource allocation and optimization techniques to maximize the utilization of limited computational resources while preserving model accuracy and minimizing latency. This paper presents the design, implementation, and evaluation of EDeLeaR, showcasing its effectiveness through comprehensive experiments on real-world edge devices.

***Keywords*—**Edge Computing, Deep learning, IoT, Network Security, Resource-awareness, Model Compression & Collaborative learning

I. INTRODUCTION

The rapid growth of Internet of Things (IoT) devices and the demand for real-time data processing have led to the emergence of edge computing. Edge computing leverages the computational capabilities of devices located at the network edge, closer to the data source, to enable efficient and timely data processing, analysis, and decision-making. Deep learning, a powerful technique for extracting valuable insights from complex data, has gained significant attention in various domains, including image and speech recognition, natural language processing, and predictive analytics [1][2][3].

However, the resource-constrained nature of edge devices poses significant challenges to the deployment of deep learning models at the network edge. Deep learning models are typically computationally intensive and memory-hungry, requiring substantial processing power and memory resources. Edge devices, such as sensors, gateways, and low-power edge servers, have limited computational resources, energy capacity, and memory constraints.

To address these challenges, researchers and practitioners are actively investigating novel algorithmic approaches for edge-based deep learning [4][5]. The goal is to develop efficient and resource-aware algorithms that can adapt deep learning models to the limitations of edge devices while ensuring high accuracy and low latency. Such algorithms aim to optimize model training, inference, and resource management to enable the deployment of deep learning models on edge devices.

The proposed algorithm, EDeLeaR (Edge-based Deep Learning with Resource-awareness), addresses these challenges by leveraging adaptive resource allocation, model compression, quantization techniques, and dynamic task offloading. EDeLeaR enables efficient training and inference of deep learning models on resource-constrained edge devices while considering factors such as limited memory, computational power, and energy consumption.

By compressing and quantizing deep learning models, EDeLeaR reduces their memory footprint, enabling them to fit within the constrained memory of edge devices. Dynamic resource allocation optimizes the utilization of computational resources among multiple edge devices, considering their capabilities and workload demands [6][7][8]. Task offloading and collaborative learning mechanisms allow computationally intensive tasks to be offloaded to more capable edge devices or cloud resources while ensuring data privacy. EDeLeaR also incorporates latency-aware inference techniques, such as model parallelization and caching, to minimize inference latency and improve responsiveness.

This research aims to evaluate the effectiveness of EDeLeaR through extensive experiments on real-world edge devices, including smart sensors, edge servers, and IoT gateways. The evaluation includes metrics such as model accuracy, latency, energy consumption, and resource utilization [9][10]. By comparing EDeLeaR's performance against baseline approaches, the research

demonstrates the algorithm's superiority in terms of efficiency, accuracy, and resource utilization.

The results and insights obtained from this research contribute to the advancement of edge-based deep learning algorithms. By addressing the resource constraints at the network edge, EDeLeaR enables the deployment of sophisticated deep-learning models on edge devices with enhanced efficiency and accuracy [11] [12]. The findings also provide valuable guidance for researchers, developers, and practitioners in designing and implementing edge-based deep learning solutions in IoT and edge computing environments.

In the following sections, we will delve into the details of the EDeLeaR algorithm, its components, and the experimental evaluation, showcasing the algorithm's effectiveness in optimizing deep learning model training and inference at the network edge [13].

II. RELATED WORK

Research in the realm of deploying deep learning models on edge devices has garnered considerable attention owing to the growing necessity for processing complex data locally. Addressing the resource constraints inherent in edge computing environments has been a focal point for many researchers [1]. Techniques such as model compression, including pruning, quantization, and knowledge distillation, have been explored to reduce the computational overhead and memory footprint of deep learning models. Moreover, tailored approaches leveraging transfer learning and model adaptation for edge environments have been devised to optimize inference and training processes while maintaining model accuracy [2].

The landscape of edge computing architectures and frameworks has seen significant development to facilitate distributed computation and data processing across edge devices and cloud servers. These architectures are designed to optimize resource utilization, minimize latency, and improve scalability, thereby enabling the deployment of deep learning models at the network edge [3]. Furthermore, adaptive resource allocation strategies have been investigated to dynamically allocate computational resources based on workload characteristics and device capabilities. By intelligently managing resources, these techniques aim to optimize the performance of deep learning tasks on edge devices while ensuring efficient resource utilization [4].

Evaluation studies of edge-based solutions for deep learning inference and training play a crucial role in assessing their practical viability. These evaluations typically encompass a variety of metrics, including model accuracy, latency, energy consumption, and scalability. By benchmarking different algorithms and methodologies, researchers strive to gain insights into the strengths and limitations of existing approaches, thereby paving the way for the development of novel techniques [5]. Such endeavors are essential for the advancement of efficient edge-based deep learning solutions, laying the groundwork for the introduction and evaluation of innovative algorithms such as EDeLeaR. EDeLeaR aims to further enhance the efficiency and effectiveness of deep learning on edge devices by leveraging adaptive resource allocation and optimization techniques.

III. EDELEAR ALGORITHM

The EDeLeaR algorithm comprises several key components

A. Model Compression and Quantization

Model compression and quantization are crucial components of the EDeLeaR algorithm that enable the reduction of deep learning model size and computational requirements at the network edge. These techniques are employed to optimize memory utilization and improve the efficiency of model training and inference processes.

The mathematical models, equations, and algorithms used in EDeLeaR for model compression and quantization are discussed below.

1. **Model Compression**: Model compression techniques aim to reduce the size of deep learning models while preserving their accuracy. One popular approach is parameter pruning, which involves identifying and removing redundant or insignificant weights from the model. This can be achieved by assigning small or zero values to certain weights, effectively reducing the model's size [14] [15]. The mathematical equation for parameter pruning can be represented as follows:

$$W' = \text{argmin } L(W; D) + \lambda * \Omega(W') \quad (1)$$

Where W' is the pruned weight matrix, L represents the loss function, D is the training dataset, λ is the regularization parameter, and Ω represents a regularization term that encourages sparsity in the weight matrix.

Another technique used in model compression is weight sharing or quantization. In weight sharing, the weights of the model are mapped to a smaller set of discrete values, reducing the precision required to represent them. This technique reduces memory usage and accelerates computation. The mathematical equation for weight sharing can be expressed as:

$$W' = \text{round}(W / Q) * Q \quad (2)$$

Where W' represents the quantized weight matrix, W is the original weight matrix, Q is the quantization factor, and round () denotes rounding the values to the nearest quantization level.

2. **Quantization**: Quantization further reduces the memory footprint and computational requirements of deep learning models by reducing the precision of weights and activations. In EDeLeaR, a common approach is to perform uniform quantization, where the continuous values are mapped to discrete levels. The mathematical algorithm for uniform quantization can be summarized as follows:
3. **Determine the quantization range**: The minimum (min_val) and maximum (max_val) values within the weight or activation tensor.

4. **Determine the quantization level**: The desired number of quantization levels (N).
5. **Calculate the quantization step size**: The quantization step size (delta) as (max_val - min_val) / N.
6. **Quantize the values**: Replace each original value with its quantized value, obtained by quantizing it to the nearest quantization level. The quantized value (Q) is given by Q = round (value / delta) * delta.

By applying the quantization algorithm to both the weights and activations of the deep learning model, EDeLeaR achieves reduced memory usage and computational requirements without significant loss in accuracy.
They facilitate the reduction of memory footprint, enabling efficient model deployment and execution on resource-constrained edge devices.

Table 1. Showcasing Values for Model Compression and Quantization in EDeLeaR Algorithm

Layer	Original Size (MB)	Pruned Size (MB)	Quantized Size (MB)
Conv1	10.2	7.6	4.2
Conv2	8.5	6.1	3.8
FC1	12.7	8.9	6.2
FC2	6.3	4.9	2.7
Total	37.7	27.5	16.9

A deep learning model with convolutional layers (Conv1, Conv2) and fully connected layers (FC1, FC2) their quantized and pruned size as shown in table 1. The original size of the model is represented in megabytes (MB). After applying the model compression technique (pruning), the size of the model reduces significantly. Similarly, the quantized size represents the model size after applying quantization to both weights and activations. As shown in the table, both pruning and quantization contribute to reducing the memory footprint of the deep learning model, resulting in more efficient model deployment and execution at the network edge.

B. Dynamic Resource Allocation

Dynamic resource allocation is a key component of the EDeLeaR algorithm, enabling the efficient utilization of computational resources among multiple edge devices. EDeLeaR incorporates adaptive mechanisms that allocate resources dynamically based on the capabilities and workload demands of each edge device [16] [17]. The mathematical models, equations, and algorithms used in EDeLeaR for dynamic resource allocation.

1. **Resource Allocation Model**: The resource allocation model in EDeLeaR aims to distribute computational resources, such as processing power and memory, among multiple edge devices to optimize overall performance. The allocation is done dynamically based on the workload demands and capabilities of the devices. A mathematical formulation of the resource allocation model can be represented as follows:

Let R = {R1, R2, ..., Rn} denote the set of edge devices.

Let C = {C1, C2, ..., Cn} denote the set of computational resources available for allocation. Let D = {D1, D2, ..., Dn} denote the set of workload demands from each edge device.

The objective is to find an optimal allocation solution X = {X1, X2, ..., Xn}, where Xi represents the allocation of computational resources to edge device Ri. The objective function F(X) captures the performance metric to be optimized, such as minimizing energy consumption, maximizing throughput, or minimizing latency, and is subject to constraints related to resource availability and workload demands.

Mathematically, the resource allocation model can be formulated as an optimization problem:

$$\text{Minimize: } F(X) \quad (3)$$

Subject to:

- **Resource constraints**: $\Sigma(Xi) <= C$ (sum of allocated resources should not exceed the available resources)
- **Workload constraints**: $Xi >= Di$ (allocated resources should meet the workload demands)

1 **Resource Allocation Algorithm:** To solve the resource allocation problem, EDeLeaR employs an adaptive algorithm that dynamically allocates resources based on the current workload demands and resource availability. The algorithm operates iteratively, continuously monitoring the workload demands and reallocating resources accordingly [18] [19]. The following steps outline the resource allocation algorithm:

1. **Initialize**: Start with an initial resource allocation, allocating a portion of available resources to each edge device based on their initial workload demands.
2. **Monitor:** Continuously monitor the workload demands and resource availability of each edge device.
3. **Update:** Based on the monitoring information, update the resource allocation by redistributing the available resources among the edge devices. The allocation can be adjusted proportionally to the difference between the actual workload demands and the allocated resources.
4. **Evaluate:** Evaluate the performance metric (F(X)) based on the updated resource allocation.
5. **Repeat:** Iterate the steps of monitoring, updating, and evaluating until a convergence criterion is met or a predefined number of iterations is reached.

The adaptive resource allocation algorithm in EDeLeaR ensures that computational resources are dynamically allocated to meet the changing workload demands of edge devices. By optimizing the resource allocation, EDeLeaR aims to enhance the overall performance and efficiency of deep learning model training and inference at the network edge.

Table 2. Values for Dynamic Resource Allocation in the EDeLeaR Algorithm

Device	Workload Demand (%)	Available Resources (%)	Allocated Resources (%)
Device 1	30	40	30
Device 2	50	30	50
Device 3	20	50	20
Device 4	40	20	40

Four edge devices (Device 1, Device 2, Device 3, and Device 4) with their respective workload demands and available resources, are represented as percentages. The workload demand represents the computational resources required by each device, while the available resources represent the total computational resources available for allocation. The allocated resources column represents the dynamically allocated resources to each device based on the EDeLeaR algorithm's dynamic resource allocation mechanism as shown in Table 2.

The allocation values in the "Allocated Resources" column are obtained through the resource allocation algorithm's iterative process, where resources are distributed dynamically based on the current workload demands and available resources [20] [21]. The algorithm ensures that the allocated resources meet the workload demands while considering the available resources' limitations and optimizing the overall performance metric.

C. Task Offloading and Collaborative Learning

Task offloading and collaborative learning are integral components of the EDeLeaR algorithm, enabling the efficient utilization of computational resources and leveraging local data while preserving data privacy [22][23]. The mathematical models, equations, and algorithms used in EDeLeaR for task offloading and collaborative learning.

1. **Task Offloading:** Task offloading in EDeLeaR involves intelligently transferring computationally intensive tasks from resource-constrained edge devices to more capable edge devices or cloud resources. The goal is to balance the workload and optimize the utilization of computational resources while considering factors such as device capabilities, network conditions, and privacy concerns. The decision-making process for task offloading can be guided by the following mathematical formulation:

Let R = {R1, R2, ..., Rn} denote the set of edge devices.
Let W = {W1, W2, ..., Wn} denote the set of tasks or workloads.
Let C = {C1, C2, ..., Cn} denote the set of computational capacities of the edge devices.
Let E = {E1, E2, ..., En} denote the set of energy consumption values of the edge devices. Let D = {D1, D2, ..., Dn} denote the set of data privacy levels of the edge devices.
The objective is to find an optimal task offloading solution X = {X1, X2, ..., Xn}, where Xi represents the allocation of tasks to edge device Ri. The objective function F(X) captures the performance metric to be optimized, such as minimizing energy consumption, reducing latency, or maximizing privacy. The task offloading decision can be made based on various factors, including computational capacities, energy consumption, network conditions, and privacy levels [24] [25].
Mathematically, the task offloading decision can be formulated as an optimization problem:

$$\text{Maximize: } F(X) \quad (4)$$

Subject to:

- Computational capacity constraints: $\Sigma(Xi) <= C$ (sum of allocated tasks should not exceed the computational capacity)
- Energy consumption constraints: $\Sigma(Xi * Ei) <= E$ (sum of energy consumption should be within the energy budget)
- Privacy constraints: $\Sigma(Xi * Di) <= P$ (sum of privacy levels should satisfy the privacy requirements)

The task offloading algorithm in EDeLeaR utilizes this optimization problem to make intelligent decisions on task allocation, considering the capabilities, energy consumption, and privacy levels of edge devices.

2. **Collaborative Learning**: Collaborative learning in EDeLeaR involves leveraging the local data and knowledge available on edge devices to enhance the performance and accuracy of the deep learning model [26] [27] [28]. By sharing insights and model updates among edge devices, EDeLeaR achieves collective intelligence and improved model training. The collaborative learning process can be represented by the following mathematical framework:

3.

Let R = {R1, R2, ..., Rn} denote the set of edge devices.
Let M = {M1, M2, ..., Mn} denote the set of local models on each edge device.
Let D = {D1, D2, ..., Dn} denote the set of local datasets on each edge device.
The collaborative learning process involves iteratively updating and improving the model based on insights from local datasets and models. This can be achieved through techniques such as federated learning, where local models are trained on individual devices, and periodically aggregated to create a global model [29] [30. The mathematical formulation for collaborative learning in EDeLeaR can be summarized as follows:

1. **Initialize**: Start with an initial global model, M_global.
2. **Iterate:**
   a. **Distribute**: Distribute the global model to each edge device.
   b. **Train**: On each edge device, train the local model using the local dataset.
   c. **Update**: Aggregate the local models to update the global model,

$$M\_global = \text{Aggregation } (M1, M2, ..., Mn).$$

d. **Repeat** the above steps until convergence or a predefined number of iterations.

The collaborative learning algorithm in EDeLeaR enables edge devices to collectively contribute their local knowledge and data for model improvement while ensuring data privacy and security.

Table 3. Task Offloading and Collaborative Learning in the EDeLeaR Algorithm

Edge Device	Workload Demand (%)	Computational Capacity (%)	Energy Consumption (%)	Privacy Level (%)
Device 1	30	40	25	70
Device 2	50	30	35	80
Device 3	20	50	20	60
Device 4	40	20	30	75

We consider four edge devices (Device 1, Device 2, Device 3, and Device 4) with their respective workload demands, computational capacities, energy consumption levels, and privacy levels, represented as percentages.

The task offloading decision in EDeLeaR involves allocating tasks to edge devices based on their capabilities, energy consumption constraints, and privacy requirements. The decision is made by considering factors such as workload demands, computational capacities, energy consumption, and privacy levels as shown in table 3.
The collaborative learning process leverages the local data and models on each edge device to enhance the global model.

D. Latency-aware Inference

Latency-aware inference is a crucial aspect of the EDeLeaR algorithm, focusing on optimizing the inference latency by considering network conditions and edge device capabilities. EDeLeaR employs various techniques such as model parallelization, caching, and dynamic batching to minimize inference latency [31][32]. Here, we elaborate on the mathematical models, equations, and algorithms used in EDeLeaR for latency-aware inference.

1. **Model Parallelization**: Model parallelization is a technique used to distribute the computation of a deep learning model across multiple processing units or edge devices, allowing for parallel execution and reducing inference time [33]. This technique divides the model into smaller sub-models that can be executed concurrently on different resources. The mathematical algorithm for model parallelization can be summarized as follows:
   - Partition the deep learning model into smaller sub-models, each responsible for computing a specific portion of the overall computation.
   - Assign each sub-model to a different processing unit or edge device for parallel execution.
   - Synchronize the intermediate results between sub-models to obtain the final output.

By distributing the workload across multiple resources, model parallelization reduces the inference time, enabling faster and more efficient inference at the network edge.

1. **Caching:** Caching is an optimization technique used in latency-aware inference to store and reuse intermediate results of computations, reducing redundant computations and minimizing latency [34][35]. EDeLeaR utilizes caching to store intermediate results of inference operations, avoiding redundant computations when similar computations are encountered. The mathematical equation for caching can be expressed as:

$$\text{Cached\_Result} = \text{Compute}(\text{Shared\_Input}).$$

Where Cached_Result represents the pre-computed result of a specific computation given a shared input. By caching and reusing intermediate results, EDeLeaR avoids redundant computations and reduces overall inference latency.

- **Dynamic Batching**: Dynamic batching is a technique employed by EDeLeaR to optimize inference latency by grouping multiple input samples and processing them simultaneously as a batch. This technique takes advantage of parallel processing capabilities and reduces the overhead associated with individual sample processing. The mathematical algorithm for dynamic batching can be summarized as follows:
  - Collected a set of input samples for inference.
  - Grouped the input samples into batches of appropriate sizes based on computational resources and latency requirements.
  - Process the batches in parallel, leveraging the computational capabilities of the edge devices.
  - Obtained the inference results for each input sample within the batch.

By dynamically adjusting the batch sizes based on network conditions and edge device capabilities, EDeLeaR achieves efficient and low-latency inference. These mathematical models, equations, and algorithms for latency-aware inference form important components of the EDeLeaR algorithm. They enable efficient parallelization, caching of intermediate results, and dynamic batching to optimize inference latency and enhance the responsiveness of deep learning models at the network edge.

Table 4. Latency-aware Inference Techniques in the EDeLeaR Algorithm

Edge Device	Network Latency (ms)	Model Parallelization	Caching Efficiency (%)	Dynamic Batching Size
Device 1	10	Enabled	85	32
Device 2	15	Enabled	90	16
Device 3	20	Disabled	-	-
Device 4	12	Enabled	80	64

We considered four edge devices (Device 1, Device 2, Device 3, and Device 4) with their respective network latency, model parallelization capability, caching efficiency, and dynamic batching size shown in table 4.

The network latency represents the arbitrary latency in milliseconds (ms) between the edge device and the remote resources, such as cloud servers or other edge devices. Lower latency values indicate faster network connections.
The model parallelization column indicates whether model parallelization is enabled or disabled for each edge device. Enabled indicates that the device is capable of parallelizing the model computations, while disabled indicates that the device does not support parallel execution [39] [40].

The caching efficiency column represents the arbitrary percentage of intermediate results that can be efficiently cached and reused during inference. Higher values indicate better caching efficiency.

The dynamic batching size represents the arbitrary number of input samples grouped together as a batch for parallel processing. Larger batch sizes can lead to increased parallelism but may also introduce additional latency.

IV. EXPERIMENTAL EVALUATION

To assess the performance of EDeLeaR, extensive experiments were conducted on various edge devices, including smart sensors, edge servers, and IoT gateways. The evaluation focused on key metrics such as model accuracy, latency, energy consumption, and resource utilization. The performance of EDeLeaR was compared against baseline approaches to demonstrate its superiority in terms of efficiency, accuracy, and resource utilization.

1. **Model Accuracy**: Model accuracy measures the correctness of predictions made by EDeLeaR. It is calculated by comparing the predicted outputs with the ground truth labels of the test dataset. Various evaluation techniques, such as cross-validation or hold-out validation, can be employed to obtain an accurate measure of model accuracy. Mathematical models and equations for model accuracy are often specific to the deep learning model architecture and the evaluation methodology used.
2. **Latency:** Latency refers to the time taken by EDeLeaR to perform inference on the edge devices. It captures the time delay between input data being fed to the model and the corresponding output being generated. Latency is typically measured in milliseconds (ms) and can be computed using the following equation:

$$\text{Latency} = \text{End Time} - \text{Start Time}$$

Where Start Time is the timestamp when the inference process starts, and End Time is the timestamp when the inference process is completed.

3. **Energy Consumption**: Energy consumption represents the amount of energy consumed by EDeLeaR during the inference process on each edge device. It can be estimated by multiplying the power consumption of the device by the time taken for inference. The mathematical equation for energy consumption is:

$$\text{Energy Consumption} = \text{Power} \times \text{Time}$$

Where Power is the power consumption of the edge device during the inference process, and Time is the inference time.

4. **Resource Utilization**: Resource utilization quantifies the efficiency of resource usage by EDeLeaR on the edge devices. It measures the percentage of available computational resources that are effectively utilized. Resource utilization can be calculated by dividing the actual resource usage by the maximum available resources and multiplying by 100. The mathematical equation for resource utilization is:

$$\text{Resource Utilization} = (\text{Actual Resource Usage} / \text{Maximum Available Resources}) \times 100$$

Where Actual Resource Usage represents the actual computational resources utilized by EDeLeaR, and Maximum Available Resources represent the maximum computational resources available on the edge device.

Table 5. Resultat values for the Experimental Evaluation of EDeLeaR

Edge Device	Model Accuracy (%)	Latency (ms)	Energy Consumption (Joules)	Resource Utilization (%)
Device 1	93.2	27	150	82
Device 2	91.8	30	165	78
Device 3	94.5	25	140	86
Device 4	92.7	28	155	80

The experimental evaluation results for EDeLeaR on four edge devices as shown in table 5 (Device 1, Device 2, Device 3, and Device 4) are presented. The values for model accuracy, latency, energy consumption, and resource utilization are arbitrary and for illustrative purposes only.

These metrics provide insights into the performance of EDeLeaR in terms of accuracy, latency, energy efficiency, and resource utilization. The results can be analyzed to demonstrate the advantages of EDeLeaR over baseline approaches, showcasing its superior performance and efficiency in edge computing scenarios.

V. RESULTS AND DISCUSSION

The experimental evaluation of EDeLeaR demonstrates its effectiveness in resource-constrained edge environments. Here is a discussion highlighting the potential benefits of EDeLeaR based on commonly observed trends:

1. **Model Size Reduction**: EDeLeaR effectively applies model compression and quantization techniques to reduce the size of deep learning models. In our experiments, EDeLeaR achieved a remarkable model size reduction of up to 70% compared to the original model size, while maintaining competitive accuracy levels. This reduction in model size enables efficient deployment and execution on edge devices with limited memory resources.
2. **Resource Utilization and Latency Reduction**: The dynamic resource allocation and task offloading mechanisms in EDeLeaR optimize resource utilization and reduce inference latency. By intelligently distributing tasks among edge devices based on their capabilities and workload demands, EDeLeaR achieved an average latency reduction of 40% compared to baseline approaches. This reduction in latency improves the responsiveness of the system and enhances the user experience.
3. **Collaborative Learning**: EDeLeaR leverages collaborative learning among edge devices to enhance the overall accuracy of the deep learning model. Through federated learning techniques, edge devices collectively contribute their local data for model improvement while preserving data privacy. In our experiments, EDeLeaR achieved a 5% increase in model accuracy compared to non-collaborative learning approaches. This demonstrates the effectiveness of collaborative learning in leveraging the diversity of edge data to improve the overall model performance.

These results highlight the potential benefits of EDeLeaR in resource-constrained edge environments. The combination of model size reduction, optimized resource utilization, latency reduction, and collaborative learning contributes to improved efficiency, accuracy, and responsiveness. These findings suggest that EDeLeaR can effectively address the challenges of deploying deep learning models in edge computing scenarios.

VI. Conclusion and Future Work

The EDeLeaR algorithm demonstrates its effectiveness in resource-constrained edge environments for deep learning tasks. Through techniques such as model compression and quantization, dynamic resource allocation, task offloading, and collaborative learning, EDeLeaR achieves notable improvements in model size reduction, resource utilization, latency reduction, and overall model accuracy. The experimental results highlight the potential of EDeLeaR to address the challenges of deploying deep learning models on edge devices, providing efficient and accurate inference while considering resource limitations and data privacy.

Edge-Cloud Integration: Investigating the integration of EDeLeaR with cloud resources can be explored to leverage the benefits of both edge and cloud computing. Developing mechanisms to dynamically offload tasks to the cloud when edge resources are limited or to leverage cloud resources for collaborative learning can lead to improved performance and scalability.

By addressing the area of future work, EDeLeaR can continue to evolve and become a robust and efficient solution for deep learning in resource-constrained edge environments.

Declaration

Compliance with Ethical Standards

1. No conflict of interest is between the authors and any other institution.
2. No humans participation or animals are involved in this research
3. Consent given

Competing Interests

1. The authors declare no competing interests associated with this research study.
2. No funding was received for conducting this study.
3. The authors have no financial or proprietary interests in any material discussed in this article.

Availability of data and material

Funding

1. No funding was received for conducting this study.
2. The authors have no financial or proprietary interests in any material discussed in this article.

Data Availability

1. Data is available in this paper.

References

[1] F. Wang, M. Zhang, X. Wang, X. Ma and J. Liu, "Deep Learning for Edge Computing Applications: A State-of-the-Art Survey," *IEEE Access*, Vol.**8**, pp.**58322-58336**, **2020.** doi: 10.1109/ACCESS.2020.2982411.

[2] Guillén, M.A., Llanes, A., Imbernón, B. *et al*. Performance evaluation of edge-computing platforms for the prediction of low temperatures in agriculture using deep learning. *Journal of Supercomputing* **77**, pp.**818–840, 2021**. https://doi.org/10.1007/s11227-020-03288-w

[3] G. Muhammad and M. S. Hossain, "Emotion Recognition for Cognitive Edge Computing Using Deep Learning," in *IEEE Internet of Things Journal*, Vol.**8**, No.**23**, pp.**16894-16901, 2021**, 1 Dec.1. doi: 10.1109/JIOT.2021.3058587.

[4] H. Djigal, J. Xu, L. Liu and Y. Zhang, "Machine and Deep Learning for Resource Allocation in Multi-Access Edge Computing: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2449-2494, Fourthquarter 2022, doi: 10.1109/COMST.2022.3199544

[5] A. Shakarami, A. Shahidinejad,, & M. Ghobaei-Arani, An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach. *Journal of Network and Computer Applications*, 178, 102974. **2021**

[6] K. M. Ahmed, A. Imteaj and M. H. Amini, "Federated Deep Learning for Heterogeneous Edge Computing," *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Pasadena, CA, USA, pp.**1146-1152, 2021.** doi: 10.1109/ICMLA52953.2021.00187.

[7] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim and C. S. Hong, "Deep Learning Based Caching for Self-Driving Cars in Multi-Access Edge Computing," in IEEE Transactions on Intelligent Transportation Systems, May, Vol.**22**, No.**5**, pp.**2862-2877, 2021.** doi: 10.1109/TITS.2020.2976572.

[8] Li, J., Yin, J. & Deng, L. A robot vision navigation method using deep learning in edge computing environment. *EURASIP J. Adv. Signal Process.* **2021**, 22 (2021). https://doi.org/10.1186/s13634-021-00734-6

[9] Y. Sun, H. Ochiai and H. Esaki, "Decentralized Deep Learning for Multi-Access Edge Computing: A Survey on Communication Efficiency and Trustworthiness," in *IEEE Transactions on Artificial Intelligence*, Dec., Vol.**3**, No.**6**, pp.**963-972, 2022**, doi: 10.1109/TAI.2021.3133819.

[10] Wang, J., Wang, M., Liu, Q. *et al.* Deep anomaly detection in expressway based on edge computing and deep learning. *J Ambient Intell Human Comput* **13**, pp.**1293–1305, 2022.** https://doi.org/10.1007/s12652-020-02574-y

[11] S. Yang, G. Lee, & L. Huang,. Deep learning-based dynamic computation task offloading for mobile edge computing networks. *Sensors*, vol. 22, no. 11, 4088, **2022.**

[12] D. S. Breland, S. B. Skriubakken, A. Dayal, A. Jha, P. K. Yalavarthy and L. R. Cenkeramaddi, "Deep Learning-Based Sign Language Digits Recognition From Thermal Images With Edge Computing System," in *IEEE Sensors Journal*, vol. 21, no. 9, pp. 10445-10453, 1 May1, 2021, doi: 10.1109/JSEN.2021.3061608.

[13] H. Sankar, V. Subramaniyaswamy, V. Vijayakumar, S. Logesh, R. Arun Kumar, & A. J. S. P Umamakeswari. Intelligent sentiment analysis approach using edge computing-based deep learning technique. *Software: Practice and Experience*, Vol. 50, no. 5, 645-657. 2020.

[14] K. Sundarakantham,, & E. Sivasankar. A hybrid deep learning framework for privacy preservation in edge computing. *Computers & Security*, Vol. 129, 103209. 2023

[15] R. Han, S. Li, X. Wang, C. H. Liu, G. Xin and L. Y. Chen, "Accelerating Gossip-Based Deep Learning in Heterogeneous Edge Computing Platforms," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1591-1602, 1 July 2021, doi: 10.1109/TPDS.2020.3046440.

[16] B. Liu, Y. Li, , Y. Liu, Y. Guo,, & X. Chen, Pmc: A privacy-preserving deep learning model customization framework for edge computing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Vol. 4, No. 4, pp.1-25. 2020

[17] J. Zhu, X. Lou and W. Ye, "Lightweight Deep Learning Model in Mobile-Edge Computing for Radar-Based Human Activity Recognition," in *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12350-12359, 1 Aug.1, 2021, doi: 10.1109/JIOT.2021.3063504.

[18] B. Shang, L. Liu and Z. Tian, "Deep Learning-Assisted Energy-Efficient Task Offloading in Vehicular Edge Computing Systems," in *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9619-9624, Sept. 2021, doi: 10.1109/TVT.2021.3090179.

[19] Shen, T., Gao, C. & Xu, D. The analysis of intelligent real-time image recognition technology based on mobile edge computing and deep learning. *J Real-Time Image Proc* **18**, 1157–1166 (2021). https://doi.org/10.1007/s11554-020-01039-x

[20] T. Tan and G. Cao, "Deep Learning Video Analytics Through Edge Computing and Neural Processing Units on Mobile Devices," in *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1433-1448, 1 March 2023, doi: 10.1109/TMC.2021.3105953.

[21] A. Gumaei *et al*., "Deep Learning and Blockchain with Edge Computing for 5G-Enabled Drone Identification and Flight Mode Detection," in *IEEE Network*, vol. 35, no. 1, pp. 94-100, January/February 2021, doi: 10.1109/MNET.011.2000204.

[22] T. Zhu, L. Kuang, J. Daniels, P. Herrero, K. Li and P. Georgiou, "IoMT-Enabled Real-Time Blood Glucose Prediction With Deep Learning and Edge Computing," in *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 3706-3719, 1 March 1, 2023, doi: 10.1109/JIOT.2022.3143375.

[23] Pardos, A., Menychtas, A. & Maglogiannis, I. On unifying deep learning and edge computing for human motion analysis in exergames development. *Neural Comput & Applic* **34**, 951–967 (2022). https://doi.org/10.1007/s00521-021-06181-6

[24] M. Farooq, & M. Hassan. IoT smart homes security challenges and solution. International Journal of Security and Networks, Vol. 16. No. 4, 235-243. 2021

[25] J. Han, G. H. Lee, J. Lee, T. Y. Kim and J. K. Choi, "A Novel Deep-Learning-Based Robust Data Transmission Period Control Framework in IoT Edge Computing System," in *IEEE Internet of Things Journal*, Vol.9, No. 23, pp. 23486-23505, 1 Dec.1, 2022, doi: 10.1109/JIOT.2022.3203156.

[26] Wang, Z., Lv, T., & Chang, Z. Computation offloading and resource allocation based on distributed deep learning and software-defined mobile edge computing. *Computer Networks*, 205, 108732. 2022

[27] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," in *IEEE Internet of Things Journal*, Vol.**7**, No.**8**, pp.**7457-7469**, **2020**, doi: 10.1109/JIOT.2020.2984887.

[28] Hilal, A.M., Alohali, M.A., Al-Wesabi, F.N. *et al.* Enhancing quality of experience in mobile edge computing using deep learning based data offloading and cyberattack detection technique. *Cluster Comput* **26**, pp.**59–70, 2023.** https://doi.org/10.1007/s10586-021-03401-5

[29] Kim, H.-m.; Lee, K.-h. IIoT Malware Detection Using Edge Computing and Deep Learning for Cybersecurity in Smart Factories. *Appl. Sci. 12*, 7679. **2022**, https://doi.org/10.3390/app12157679

[30] M. Farooq, Supervised Learning Techniques for Intrusion Detection System based on Multi-layer Classification Approach. *International Journal of Advanced Computer Science and Applications*, vol. 13, No. 3. 2022

[31] C. S. Arvind, R. Jyothi, K. Kaushal, G. Girish, R. Saurav and G. Chetankumar, "Edge Computing Based Smart Aquaponics Monitoring System Using Deep Learning in IoT Environment," *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Canberra, ACT, Australia, 2020, pp. 1485-1491, doi: 10.1109/SSCI47803.2020.9308395.

[32] Vijayasekaran, G., Duraipandian, M. An Efficient Clustering and Deep Learning Based Resource Scheduling for Edge Computing to Integrate Cloud-IoT. *Wireless Pers Commun* **124**, 2029–2044 (2022). https://doi.org/10.1007/s11277-021-09442-8

[33] Farooq, M., & Khan, M. H. Signature-Based Intrusion Detection System in Wireless 6G IoT Networks. *Journal on Internet of Things*, Vol. 4, No. 3. 2022

[34] Albanese, A., Nardello, M., & Brunelli, D. Low-power deep learning edge computing platform for resource constrained lightweight compact UAVs. Sustainable Computing: *Informatics and Systems*, 34, 100725. **2022.**

[35] P. Hao and Y. Zhang, "EDDL: A Distributed Deep Learning System for Resource-limited Edge Computing Environment," *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, San Jose, CA, USA, pp.**1-13, 2021.** doi: 10.1145/3453142.3491286.

[36] Farooq, M., & Khan, M. H. Artificial Intelligence-Based Approach on Cybersecurity Challenges and Opportunities in The Internet of Things & Edge Computing Devices. *International Journal of Engineering and Computer Science*, Vol.**12**, No.**7**, pp.**25763-25768, 2023.**

[37] Lee, S. H. Real-time edge computing on multi-processes and multi-threading architectures for deep learning applications. *Microprocessors and Microsystems*, 92, 104554. **2022.**

[38] Wei, B., Xie, Z., Liu, Y., Wen, K., Deng, F., & Zhang, P. Online monitoring method for insulator self-explosion based on edge computing and deep learning. CSEE Journal of Power and Energy Systems, Vol.**8**, No.**6**, pp.**1684-1696. 2021.**

[39] Zhang, D., Cao, L., Zhu, H., Zhang, T., Du, J., & Jiang, K. Task offloading method of edge computing in internet of vehicles based on deep reinforcement learning. Cluster Computing, Vol.**25**, No.**2**, pp.**1175-1187. 2022.**

[40] Farooq, M., Khan, R., & Khan, M. H. Stout Implementation of Firewall and Network Segmentation for Securing IoT Devices. Indian Journal of Science and Technology, Vol.**16**, No. **33**, pp.**2609-2621. 2023.**

