SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Efficient Model Compression for Resource-Constrained AI Systems Using Complexity-Driven Pruning

Mukesh Kumar[1], Hannah Jessie Rani R[2], G. Karthika[3], R. Suresh[4], Jai Kumar B[5], Arivukkodi R[6], Seethaladevi S[7]

[1] Associate Professor, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Parul Institute of Technology, Parul University, Vadodara, Gujarat, India. Email: mukesh.kumar35946@paruluniversity.ac.in, ORCID: 0009-0005-2673-9553
[2] Assistant Professor, Department of Electrical and Electronics Engineering, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Bengaluru, Karnataka, India. Email: jr.hannah@jainuniversity.ac.in, ORCID: 0000-0002-5449-104X
[3] Assistant Professor, Department of Electronics and Communication Engineering, Saveetha Engineering College, Thandalam, Chennai – 602105, India. Email: mkarthi.ganesan@gmail.com
[4] Assistant Professor, Department of Electronics and Communication Engineering, Sona College of Technology, Salem, Tamil Nadu, India. Email: suresh.ece@sonatech.ac.in
[5] Assistant Professor, Department of Computer Science and Engineering, Presidency University, Bangalore, Karnataka, India. Email: jai.kumar@presidencyuniversity.in, ORCID: 0000-0002-0397-7664
[6] Assistant Professor, Computer Science, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: arivukodir@maher.ac.in
[7] Assistant Professor, Department of Mathematics, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: seethaladevi@maher.ac.in

Abstract
The rapid advancement of edge AI has enabled intelligent Internet of Things (IoT) applications such as smart healthcare, industrial automation, and agriculture. However, deploying deep learning (DL) models on resource-constrained devices is challenging due to high computation, memory, and energy demands. To address this, the research uses a dataset of 6784 layer-level records capturing computational cost, memory usage, and operational impact, and proposes a complexity-driven pruning-based model compression framework for efficient deployment. Unlike traditional approaches that rely on iterative training, pruning, and retraining cycles, the proposed Kookaburra Optimized Stacked Long Short-Term Memory (KO-StackedLSTM) Network performs layer-wise complexity analysis to selectively remove less significant filters, reducing computational overhead without expensive fine-tuning. The KO-StackedLSTM uses a bio-inspired Kookaburra optimization to remove redundant parameters and employs a stacked LSTM structure to improve temporal learning with efficient, accurate inference. To further enhance performance, Min–Max normalization is applied for improved data scaling and convergence, while PCA reduces input dimensionality, preserving essential features and minimizing processing cost. Additionally, the research introduces three adaptive compression modes: FLOPs-aware (FA), parameter-aware (PA), and memory-aware (MA) to enable flexible optimization based on specific resource constraints. It also presents a trade-off analysis between resource use and performance, offering practical insights for real-world deployment. The model achieves a high accuracy of 96.38% with an FLOPs by 83.40% using Python, demonstrating its effectiveness for efficient AI deployment in resource-constrained environments. Overall, the research provides a scalable and efficient solution for real-time inference under limited resources.s

Keywords: Model Compression, Network Pruning, AI, IoT, Resource-Constrained Systems, Efficient Deep Learning.

1. INTRODUCTION

The extensive use of artificial intelligence (AI) technologies in various industries is one of the reasons that intelligent systems should be deployed in constrained environments such as mobiles, IoT, and embedded platforms [1,2]. DL models have demonstrated tremendous accuracy in solving difficult problems such as speech recognition, text processing, and image recognition. However, most DL models are computationally expensive as they require large memory, a powerful central processing unit, and high energy consumption, despite the fact that they deliver good results. Due to constraints such as latency, memory capacity, and energy consumption, the deployment of such models becomes difficult [3, 4]. It is important to note that studies on

model compression have been found helpful in solving the above challenge through the development of methods aimed at reducing the model's size and computation while retaining its accuracy [5, 6]. In order to come up with compressed models to be applied in edge computing applications, various model compression techniques have been extensively studied, including pruning, quantization, knowledge distillation, and low rank approximation [7]. However, achieving an effective balance between model efficiency and performance is not easy in limited-resource AI applications. Over-compression causes the loss of performance, whereas under-optimization cannot satisfy the constraints of the device. Adaptive and hardware-aware compression approaches have received considerable attention due to their ability to optimize models to fit into particular devices. Meanwhile, coupling compression with other optimization approaches, like dynamic inference, model splitting, and edge-cloud cooperation, yield even better results [8, 9]. The emergence of edge intelligence and privacy-preserving AI has increased the relevance of performing inference operations on the devices themselves. Inference using cloud servers alone would result in communication delays and possible data exposure to cyber-attacks [10, 11]. The use of compressed models allows for local processing of data, ensuring fast results and enhanced privacy. Recent advancements in automatic compression, neural architecture search, and co-design of hardware and software hold great promise in developing efficient models with minimal intervention [12, 13]. Overall, model compression methods are useful in creating efficient AI solutions. Reduces cost of calculations, allowing for scalable and efficient task completion. Future research using model compression likely produce increased mathematical algorithms, improved method of finding effective neural architectures, and integrated design of hardware and software.

1.1 Research Objective and Organization

This research proposes a complexity-driven model compression method based on pruning of the KO-StackedLSTM model, which minimizes computation costs, memory requirements, and energy consumption, and maintains high levels of accuracy in order to achieve efficient implementation of DL models in edge IoT devices with limited resources.

This research is structured as follows: Introduction, Chapter 1, Previous Research, Chapter 2, Methodology, Chapter 3, Performance Evaluation, Chapter 4, Discussion, and Conclusion, Chapter 5.

2. PREVIOUS RESEARCH

In recent research, various methods have been examined to improve the performance, robustness, and scalability of artificial intelligence algorithms in critical environments. In [14], a robust AI architecture using dynamic neural networks with meta-training was suggested, where the results have shown remarkable improvements in resilience (24%), adaptability (~22%), and computational performance (>30%) in the face of faults and attacks; however, the data used in the paper is opaque, and there is no analysis of the application of the algorithm in real-time. Likewise, [15] analyzed the optimization of XR-enhanced Digital Twins in IoT by performing QoE/QoS assessment via survey; the findings suggest latency and resource limitations as the main challenges of implementing such solutions, yet there is no experimental evaluation. Furthermore, in [16], an asymmetric exponent technique was developed for low-precision training, where equivalent accuracy was achieved in 8-bit and 32-bit models on different architectures such as LeCun, AlexNet, and ResNet-18; however, the proposed algorithm can only be applied to certain CNNs, and hardware testing is required. Meanwhile, [17] designed an intrusion detection system based on quantization with an autoencoder, resulting in huge savings in terms of memory usage (70.01%), model size (92.23%), and CPU utilization (27.94%), while maintaining excellent performance in detection accuracy, albeit limited to only one data set. On the other hand, [18] suggested a novel hybrid pruning algorithm that combines dynamic channel-wise pruning and layer-wise fusion, thus saving up to 80.05% in computational costs and reducing accuracy losses to just 0.72%, despite being confined only to the compression of ResNet-110.

Note: AI – Artificial Intelligence, XR – Extended Reality, IoT – Internet of Things, QoE – Quality of Experience
QoS – Quality of Service, CNNs – Convolutional Neural Networks, CPU – Central Processing Unit

3. METHODOLOGY

Methodology involves using a dataset comprising 6784 instances with layer-based features, like the cost of computation, the amount of memory consumed, and the effect on operation. Min–Max normalization and PCA are employed to scale down features and perform dimensionality reduction. The KO-StackedLSTM is used to assess layer-level complexity before applying the pruning technique to remove less significant filters. Lastly, PA,

FA, and MA modes adaptively optimize the model for resource limitations, and Figure 1 demonstrates the process of the research.

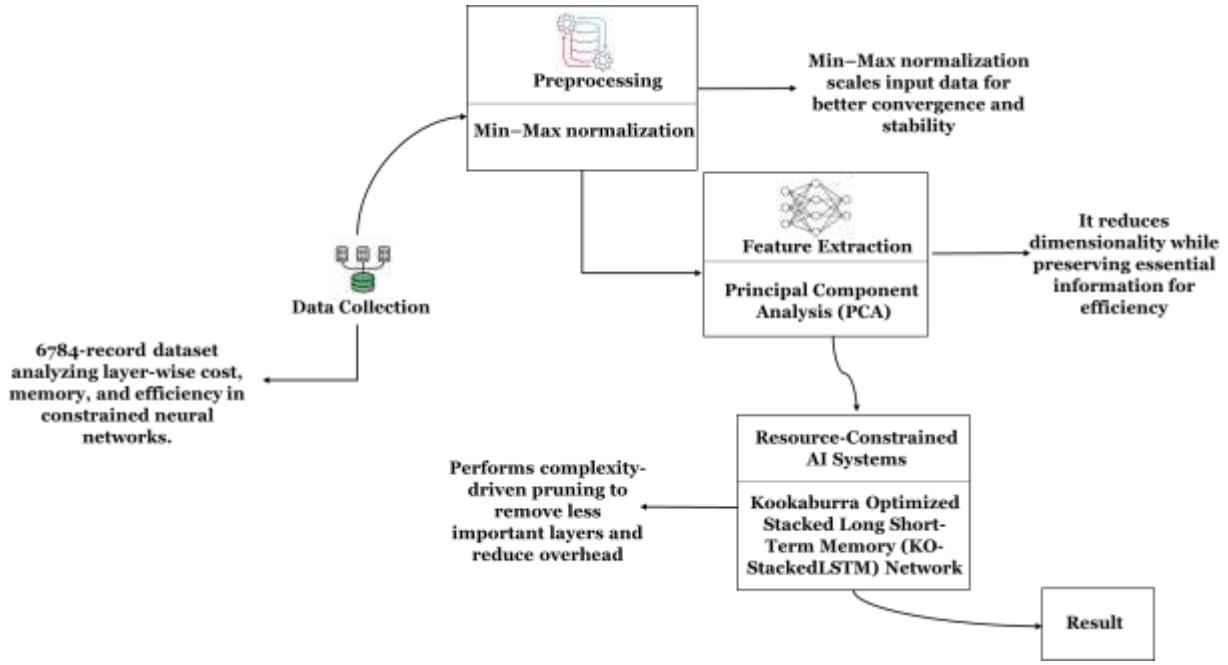

Figure 1: Layer-Wise Optimization and Compression Framework

3.1 Data Description

The 6784 instances contain details regarding certain attributes of the architecture utilized by the neural network in scenarios where there are limitations in resources. It aims to quantify the effect of different operations, memory usage, and computational cost incurred by different parts of the network. The purpose of this dataset is to make it simpler to investigate the effects of different kinds of resource restrictions on optimization approaches and the contribution of each layer to the complexity of the model. Data Score: (https://www.kaggle.com/datasets/colabsss/complexity-aware-model-pruning-dataset).

3.2 Data preprocessing using Min–Max normalization

Min–Max normalization is applied to scale input data into a fixed range, improving convergence and training stability. This reduces computational overhead and supports efficient model compression for resource-constrained environments. The normalization process is defined as:

$$Z_{new} = \frac{Z - \min(Z)}{\max(Z) - \min(Z)} \quad (1)$$

In Equation (1), Z is the original data value, Z_{new} represents the normalized value, and the dataset's minimum and maximum values are shown by min (Z) and max(Z), respectively. Besides the process of normalization, the technique of pruning is used for the removal of any irrelevant and less important parameters in the network. It results in a reduction of the complexity of the model as well as the use of memory and computations while retaining the necessary accuracy of the system.

3.3 Principal Component Analysis (PCA) for Feature Extraction

Using PCA and pruning techniques in this research is to minimize the dimensionality of features and avoid redundancy in parameters while retaining necessary information, thus making the system more efficient, less complex, and deployable in constrained artificial intelligence environments. The average of the data in each dimension is calculated as:

$$\mu_n = \frac{1}{M}\sum_{m=1}^{M} Y_{mn} \quad (2)$$

In Equation (2) M is the total number of features, Y_{mn} indicates the value of the m^{th} feature for the n^{th} sample, μ_n denotes the mean value of the n^{th} sample, and $\sum_{m=1}^{M}$ is the summation over all features. The covariance matrix represents the variance of a feature (or input variable) at a particular time and captures its relationship with other features. It is computed as shown in equation (3):

$$D = \frac{1}{M} AA^T \quad (3)$$

A is the zero-mean data matrix, A^T means the transpose of matrix A, D stands for covariance (or dispersion) matrix, and M is the number of features (or normalization factor). By ordering the eigenvalues in decreasing order of value, it is possible to identify the components providing the most contribution to the variation of the set of data. Achieving this objective maximize the information that can be obtained from, reduce the amount of redundant information, and generate high-quality feature representations suitable for efficient processing in limited-resource environments.

3.4 KO-StackedLSTM: Complexity-Aware Model Optimization Framework

The KO-StackedLSTM model is an integrated model that combines Kookaburra optimization with a stacked LSTM. The model's complexity has been used to eliminate redundancies in network parameters through a process called pruning; therefore, the model can perform optimally without the need to retrain the network.

3.4.1 SLSTM for Resource-Constrained AI Systems

Through the utilization of stacked layers of long short-term memory (LSTM) networks, the proposed SLSTM model increases performance for scenarios with time series data by creating more accurate temporal dependencies than conventional recurrent neural networks (RNNs). The new model work for low-computing resources due to the use of parameter compression and pruning techniques. By employing various regularization techniques such as dropout, and early stopping, stacked LSTMs and dense stochastic models provide better fit than non-stochastic models given their independent data. Additionally, training optimization and fine-tuning of hyperparameters help to achieve better efficiency and scalability. Figure 2 shows data processing, reduction, and pruning to produce an efficient compact model

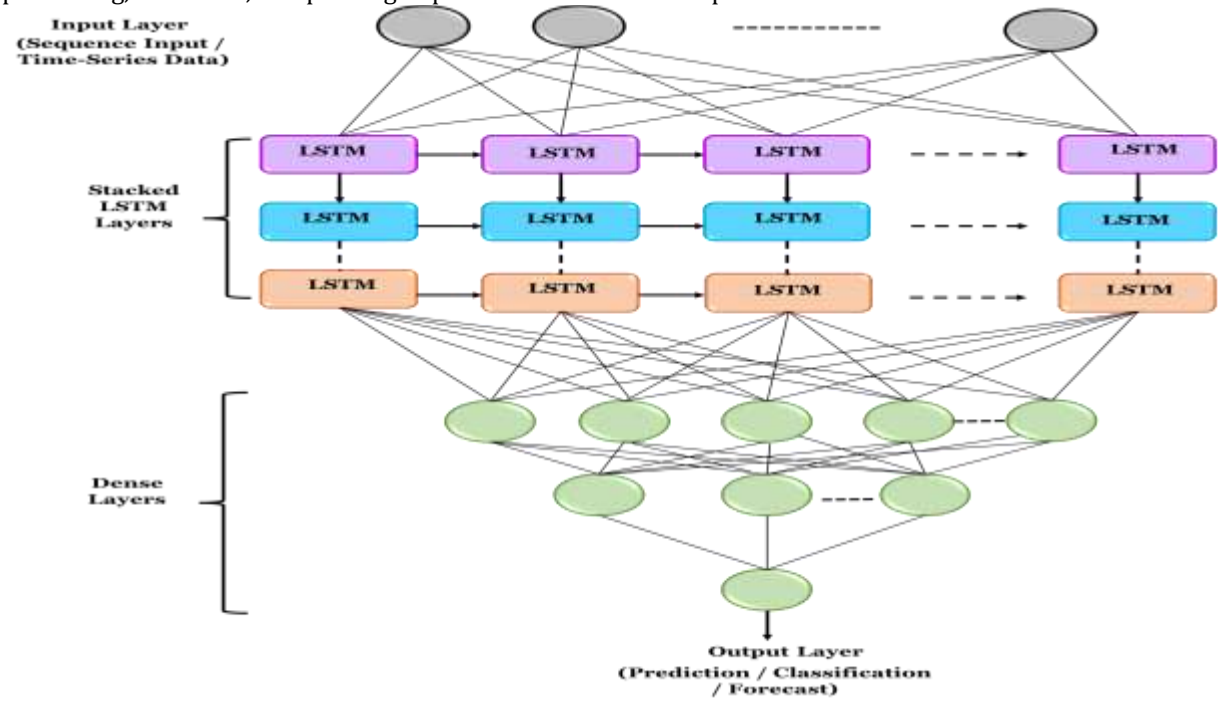

Figure 2: Layered Structure of the Stacked LSTM Model

3.4.2 Kookaburra Optimization Algorithm (KOA) for Model Compression

In this section, a mathematical formulation of the KOA is scaled down to be efficiently compressed in resource-constrained AI systems. Following the hunting patterns of kookaburra birds, the algorithm is modeled based on cooperation in exploration, rapid exploitation of promising areas, and relocation when new, better solutions are located. Within the suggested scheme, every kookaburra can be seen as a possible compression setup, including ratios pruning and choice of filter layers. The algorithm is searching to settle on an optimum configuration to reduce the complexity of the computation and minimize the model accuracy.

- **Initialization**

Candidate solutions are randomly generated within bounds and evaluated using an objective function to measure compression efficiency.

- **Exploration Phase**

Solutions move toward better candidates (prey), enabling global search and avoiding local minima.

$$GP_{i,d}^{1} = g_{i,d} + \text{rand}() \cdot (SCP_{i,d} - I \cdot g_{i,d}) \quad (4)$$

In Equation (4) $GP_{i,d}^{1}$ is the new (updated) value of the d^{th} decision variable for the i^{th} solution after exploration, $g_{i,d}$ is the current position of that variable. The function rand() $\in [0,1]$ introduces randomness, ensuring diverse search behavior. Where $I \in \{1, 2\}$ and $SCP_{i,d}$ is the selected prey position.

Exploitation Phase (Local Search)

After identifying promising regions, the algorithm performs fine-tuning:

$$GP_{i,d}^{2} = g_{i,d} + (1 - 2 \cdot \text{rand}()) \cdot \frac{ub_d - lb_d}{t} \quad (5)$$

In Equation (5) $GP_{i,d}^{2}$ is the updated value of the d^{th} variable for the i^{th} solution, and $g_{i,d}$ is its current value. The term rand() $\in \{1, 2\}$ introduces randomness, so $(1 - 2 \cdot \text{rand}())$ generates values in the range [-1, 1], generates values in the range $ub_d - lb_d$ defines the search range, and t is the current iteration number. The solution is updated as:

$$G_i = \begin{cases} GP_i^2, & \text{if} F_i^2 < F_i \\ G_i, & \text{otherwise} \end{cases} \quad (6)$$

In Equation (6), G_i is the current solution, F_i is the current fitness, and F_i^2 is the new fitness. GP_i^2 is the updated solution from the exploitation phase. If the new solution is better ($F_i^2 < F_i$), it replaces the current one; otherwise, the original solution is retained. As the iterations of the search the search area can decrease, allowing for more local optimization of the solution space and graduating the solution towards the global optimum of that model.

The KOA-SLSTM approach is an integration of the Kookaburra Optimization algorithm with SLSTM for modeling compression. Kookaburra Optimization selects and eliminates insignificant parameters based on their impact on the degree of complexity and accuracy. The optimized version of SLSTM achieves high learning capacity by using a few computational, memory, and energy resources. Algorithm 1 shows the KOA-assisted KO-StackedLSTM approach used to pre-process the data, reduce its dimensionality, and eliminate insignificant parameters.

Algorithm 1: KO-StackedLSTM-Based Model Compression

Input: Dataset D
Output: Compressed and optimized KO − StackedLSTM model
1: Load dataset D
2: Apply Min − Max Normalization on D
3: Apply PCA for dimensionality reduction → D_reduced
4: Initialize Stacked LSTM (SLSTM) model
5: Extract layer − wise features (FLOPs, memory, importance scores)
6: Initialize KOA parameters:
  − Population size N
  − Max iterations T
  − Lower bound lb, Upper bound ub
7: Generate initial population G randomly
8: Evaluate fitness ObF(G_i) for each solution:
  (based on accuracy + complexity reduction)
9: For t = 1 to T do
10: For each candidate solution G_i do
11: // Exploration Phase
12: Identify better solutions (prey) DP_i
13: Update position:
  $GP\_i^1 = G\_i + \text{rand}() * (\text{selected\_prey} - I * G\_i)$
14: If fitness improves:
  $G\_i = GP\_i^1$
15: // Exploitation Phase

16: Update position locally:
$GP\_i^2 = G\_i + (1 - 2 * \text{rand}()) * (ub - lb)/t$
17: If fitness improves:
$G\_i = GP\_i^2$
18: End For
19: End For
20: Select best solution G_best
21: Apply pruning based on G_best:
− Remove less important layers/filters
22: Train optimized KO − StackedLSTM model
23: Output compressed model

4. PERFORMANCE EVALUATION

This section is concerned with the evaluation of the KO-StackedLSTM model implemented using the means of the Python programming language. The efficiency of the proposed approach is demonstrated via reduction in terms of computational complexity, memory consumption, as well as in terms of decreasing model sizes while preserving performance for various types of adaptive compression methods.

4.1 Data exploratory feature analysis

Figure 3 (a-e) describes complexity characteristics, along with performance parameters of models, including variation in inference times, number of FLOPs per layer, and scores for determining layer significance in pruning approaches. Moreover, the figure indicates the dependency between FLOPs, energy, and memory requirements for models, revealing certain variability of computational resources for training purposes and highlighting the complexity-oriented goals of compression methods.

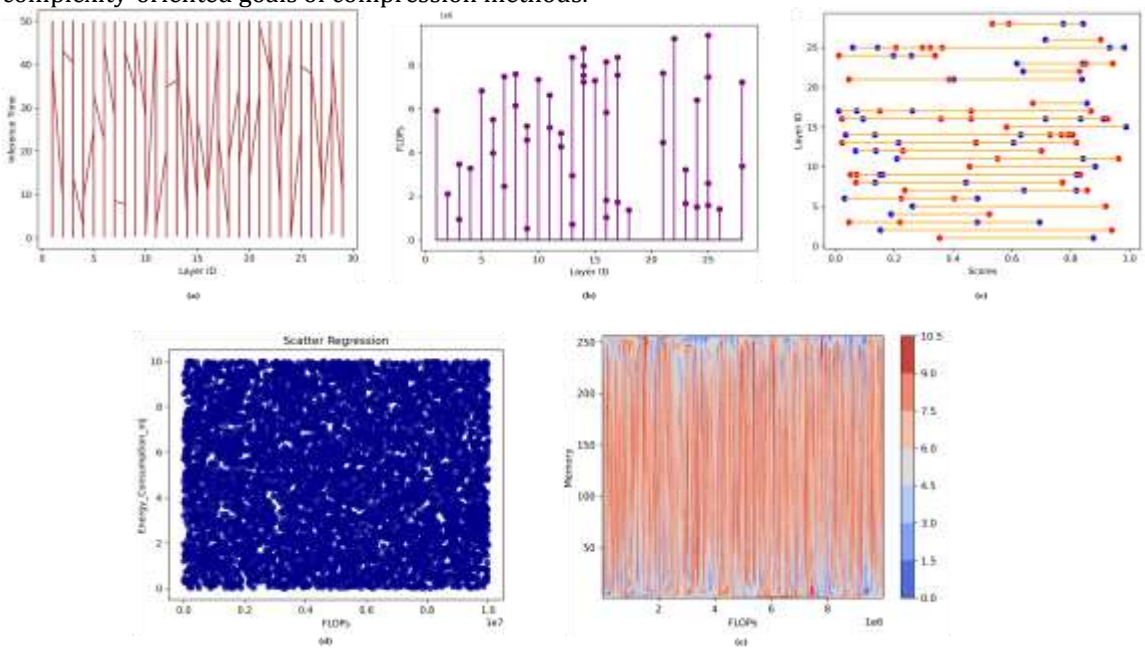

Figure 3: The visualization of (a): Inference time variation across layers, (b): Layer-wise FLOPs distribution, (c): Score-based layer importance for pruning, (d): FLOPs versus energy consumption relationship, and (e): Memory utilization trends in complexity-aware model compression

Figure 4 (a-c) depicts the pattern of distribution of layer importance scores, memory consumption parameters, as well as the patterns of function responses to parameter variations in PA, FA, and MA compression methods based on complexity considerations.

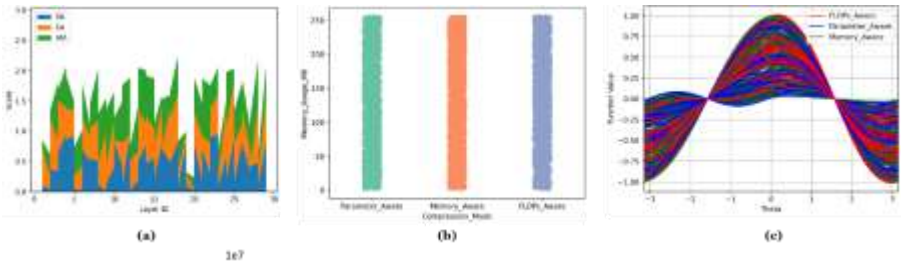

Figure 4: (a) Layer-wise complexity scores; (b) memory usage across PA, FA, and MA modes; (c) optimization behavior under different compression modes

4.2 Comparison Phase

CIFAR-10 [18] and ImageNet50 [18] are existing datasets and are commonly used to test the performance of DL models, including ResNet and DenseNet. ImageNet50 offers 50 classes with high-resolution 224x224 images, while CIFAR-10 has 50k training and 10k test images that are 32x32. In Table 1, the proposed KO-StackedLSTM was trained on such datasets, with a better compressed accuracy and much more FLOPs reduction than current models, thus showing better performance and efficiency.

Table 1: Accuracy and Computational Efficiency Comparison on CIFAR-10 and ImageNet50 Datasets

Model	Original Acc. (%)	Acc. ↓ (%)	Compressed Acc. (%)	Original FLOPs (M)	FLOPs ↓ (%)	Compressed FLOPs (M)
	CIFAR-10 Dataset					
ResNet-56 [18]	93.52	0.77	92.75	127.62	72.11	35.59
ResNet-110 [18]	93.76	0.72	93.04	257.09	80.05	51.28
DenseNet-40 [18]	94.53	0.87	93.66	292.56	39.80	176.1
KO-StackedLSTM [Proposed]	95.20	0.65	94.80	300.78	81.70	34.25
	ImageNet50 Dataset					
DenseNet-121[18]	90.21	0.92	89.29	59.20	36.48	37.60
KO-StackedLSTM [Proposed]	92.40	0.91	91.25	60.78	37.65	38.17

The proposed KO-StackedLSTM and existing models, such as ResNet and DenseNet, were trained and evaluated on the Complexity-Aware Model Pruning Dataset [Proposed]. Table 2 and Figure 5 clearly demonstrate that our model not only performed better in terms of achieving higher compressed accuracy but also minimized the loss of accuracy while obtaining substantial improvements in FLOPS reduction, thus exhibiting better trade-off capabilities than the state-of-the-art methods.

Table 2: Complexity-Aware Compression and Performance Analysis on the Proposed Dataset

Model	Original Acc. (%)	Acc. ↓ (%)	Compressed Acc. (%)	Original FLOPs (M)	FLOPs ↓ (%)	Compressed FLOPs (M)
ResNet-56	94.10	0.76	93.14	126.40	73.34	34.67
ResNet-110	94.25	0.71	94.16	256. 13	81.20	51.42
DenseNet-40	95.56	0.86	94.56	291.70	40.34	175. 8
DenseNet-121	91.12	0.91	90.78	60.15	37.05	36.57
KO-StackedLSTM [Proposed]	97.17	0.68	96.38	294.80	83.40	32.15

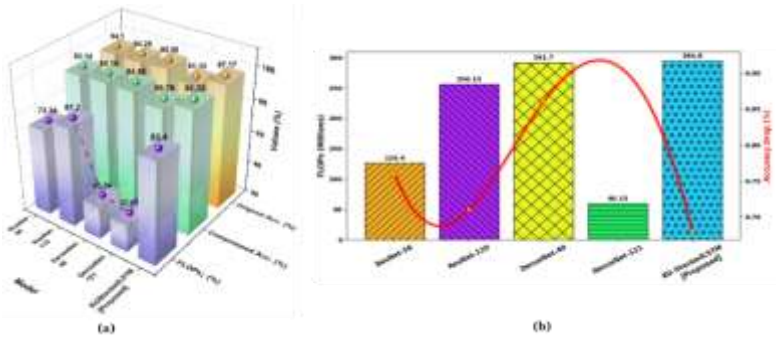

Figure 5: Efficiency–Accuracy Trade-off Analysis of Compressed Deep Learning Models

The proposed model presents a novel compression technique utilizing normalization, PCA for feature extraction, and pruning in the KO-StackedLSTM architecture. It allows for removing unnecessary layers without further training, and it stands out from other existing methods. However, the research is confined only to the ResNet-110 structure [18]. Also, the sensitivity of the method to training parameters needs to be considered. Moreover, pruning can be done excessively, thus resulting in loss of crucial features, and it still needs to measure the improvement in speed.

5. CONCLUSION

The research focuses on addressing the challenge involved in the adoption of DL models to limited-resource devices because of the massive computational, memory, and energy demands. The study proposes the KO-StackedLSTM model, a complexity-aware model compression technique by pruning layers, Min-Max normalization, PCA, and adaptive mode (PA, FA, MA) without the need for iterative training. The KO-StackedLSTM model gives significant performance on the provided data set, showing an accuracy rating at 96.38% and FLOPs by 83.40%, thus demonstrating its effectiveness for deployment. However, the KO-StackedLSTM model has some limitations with respect to specific architecture designs, being sensitive to parameter choices, susceptibility to over-pruning, and having no ability to validate results in real-world applications. Many of the limitations concerning the KO-StackedLSTM model would need to be evaluated in future research.

REFERENCE

1. Pazmiño Ortiz, L.A., Maldonado Soliz, I.F. and Guevara Balarezo, V.K., 2025. Advancing TinyML in IoT: A Holistic System-Level Perspective for Resource-Constrained AI. Future Internet, 17(6), p.257.https://doi.org/10.3390/fi17060257
2. Mughal, F.R., He, J., Das, B., Dharejo, F.A., Zhu, N., Khan, S.B. and Alzahrani, S., 2024. Adaptive federated learning for resource-constrained IoT devices through edge intelligence and multi-edge clustering. Scientific Reports, 14(1), p.28746.https://doi.org/10.1038/s41598-024-78239-z
3. Piechocki, M., Kraft, M., Pajchrowski, T., Aszkowski, P. and Pieczynski, D., 2022. Efficient people counting in thermal images: the benchmark of resource-constrained hardware. IEEE Access, 10, pp.124835-124847.https://doi.org/10.1109/ACCESS.2022.3225233
4. Xu, T., Wei, Z., Xu, T. and Zheng, G., 2024. A low-cost multi-band waveform security framework in resource-constrained communications. IEEE Transactions on Wireless Communications, 23(8), pp.9190-9205.https://doi.org/10.1109/TWC.2024.3360130
5. Chaudhri, S.N., Rajput, N.S., Alsamhi, S.H., Shvetsov, A.V. and Almalki, F.A., 2022. Zero-padding and spatial augmentation-based gas sensor node optimization approach in resource-constrained 6G-IoT paradigm. Sensors, 22(8), p.3039.https://doi.org/10.3390/s22083039
6. Ravi, N. and El-Sharkawy, M., 2022. Real-time embedded implementation of improved object detector for resource-constrained devices. Journal of Low Power Electronics and Applications, 12(2), p.21.https://doi.org/10.3390/jlpea12020021
7. Wang, J., Du, H., Niyato, D., Kang, J., Xiong, Z., Rajan, D., Mao, S. and Shen, X., 2024. A unified framework for guiding generative AI with wireless perception in resource constrained mobile edge networks. IEEE Transactions on Mobile Computing, 23(11), pp.10344-10360.https://doi.org/10.48550/arXiv.2309.01426

8. Wen, D., Jeon, K.J. and Huang, K., 2022. Federated dropout—A simple approach for enabling federated learning on resource constrained devices. IEEE wireless communications letters, 11(5), pp.923-927.https://doi.org/10.1109/LWC.2022.3149783
9. Shaushenova, A., Kuznetsov, O., Nurpeisova, A. and Ongarbayeva, M., 2025. Implementation of kolmogorov–arnold networks for efficient image processing in resource-constrained internet of things devices. Technologies, 13(4), p.155.https://doi.org/10.3390/technologies13040155
10. Gad, G., Farrag, A., Aboulfotouh, A., Bedda, K., Fadlullah, Z.M. and Fouda, M.M., 2024. Joint self-organizing maps and knowledge-distillation-based communication-efficient federated learning for resource-constrained UAV-IoT systems. IEEE Internet of Things Journal, 11(9), pp.15504-15522.https://doi.org/10.1109/JIOT.2023.3349295
11. Shao, J., Wang, G., Yi, L., Wang, C., Lan, T., Xu, X., Guo, J., Deng, T., Liu, D., Chen, B. and Yi, Z., 2022. Deep learning empowers lung cancer screening based on mobile low-dose computed tomography in resource-constrained sites. Frontiers in Bioscience-Landmark, 27(7), p.212.https://doi.org/10.31083/j.fbl2707212
12. Lin, Z., Qu, G., Wei, W., Chen, X. and Leung, K.K., 2025. Adaptsfl: Adaptive split federated learning in resource-constrained edge networks. IEEE Transactions on Networking, 14(8).https://doi.org/10.1109/TON.2025.3577790
13. Huang, Q., 2022. Weight-quantized squeezenet for resource-constrained robot vacuums for indoor obstacle classification. AI, 3(1), pp.180-193. https://doi.org/10.3390/ai3010011
14. Moskalenko, V., Kharchenko, V. and Semenov, S., 2024. Model and method for providing resilience to resource-constrained AI-system. Sensors, 24(18), p.5951.https://doi.org/10.3390/s24185951
15. Kamdjou, H.M., Baudry, D., Havard, V. and Ouchani, S., 2024. Resource-constrained extended reality operated with digital twin in industrial internet of things. IEEE Open Journal of the Communications Society, 5, pp.928-950.https://doi.org/10.1109/OJCOMS.2024.3356508
16. Pietrołaj, M. and Blok, M., 2024. Resource constrained neural network training. Scientific Reports, 14(1), p.2421.https://doi.org/10.1038/s41598-024-52356-1
17. Sharmila, B.S. and Nagapadma, R., 2023. Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset. Cybersecurity, 6(1), p.41.https://doi.org/10.1186/s42400-023-00178-5
18. Li, Q., Li, H. and Meng, L., 2023. Deep learning architecture improvement based on dynamic pruning and layer fusion. Electronics, 12(5), p.1208.https://doi.org/10.3390/electronics12051208