

Compact optimized deep learning model for edge: a review

Soumyalatha Naveen¹, Manjunath R. Kounte²

¹School of Computer Science and Engineering, REVA University, Bangalore, India

²School of Electronics and Communication Engineering, REVA University, Bangalore, India

Article Info

Article history:

Received Jan 7, 2023

Revised Apr 23, 2023

Accepted Apr 24, 2023

Keywords:

Deep learning

Edge

Energy efficient

Pruning

Quantization

ABSTRACT

Most real-time computer vision applications, such as pedestrian detection, augmented reality, and virtual reality, heavily rely on convolutional neural networks (CNN) for real-time decision support. In addition, edge intelligence is becoming necessary for low-latency real-time applications to process the data at the source device. Therefore, processing massive amounts of data impact memory footprint, prediction time, and energy consumption, essential performance metrics in machine learning based internet of things (IoT) edge clusters. However, deploying deeper, dense, and hefty weighted CNN models on resource-constraint embedded systems and limited edge computing resources, such as memory, and battery constraints, poses significant challenges in developing the compact optimized model. Reducing the energy consumption in edge IoT networks is possible by reducing the computation and data transmission between IoT devices and gateway devices. Hence there is a high demand for making energy-efficient deep learning models for deploying on edge devices. Furthermore, recent studies show that smaller compressed models achieve significant performance compared to larger deep-learning models. This review article focuses on state-of-the-art techniques of edge intelligence, and we propose a new research framework for designing a compact optimized deep learning (DL) model deployment on edge devices.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Soumyalatha Naveen

School of Computer Science and Engineering, REVA University

Bangalore, India

Email: soumyanaveen.u@gmail.com

1. INTRODUCTION

Edge intelligence [1], [2] pushes intelligence from the cloud to edge devices for faster results prediction for many real-time applications such as self-driving, augmented reality, and intelligent surveillance systems. Due to the minimum latency requirement of real-time applications, there is a shift from an artificial intelligent (AI)-based cloud computing approach to edge/fog [3] devices. Hence edge intelligence [4] reduces the response time and network bandwidth required for data movement from the device to the cloud infrastructure for processing and getting back the result. In our daily lives, more applications [5], [6] based on smart devices for home automation, smart healthcare through wearables devices, intruder detection through the smart camera and smart mobiles, smart parking, and smart factory applications improved people's daily life.

Though intelligent internet of things (IoT) edge computing [7] benefitted from various applications, resource constraint in terms of memory, processing capability, and energy efficiency incredibly imposes challenges in embedding intelligence through deep learning model into IoT devices. Deep learning models continue to grow in their size with an increasing number of layers and nodes, making it difficult to deploy on portable resource constrained (such as memory, central processing unit (CPU), energy, and bandwidth)

devices. Convolutional neural network (CNN) is suitable for computer vision tasks like categorization, object identification, speech recognition, machine translation, augmented reality, picture annotation, autonomous driving, object tracking, unmanned aerial vehicles (UAV) obstacle avoidance, segmentation, robot vision and activity recognition. However, real-time embedded system-based applications or smart devices cannot use CNN-based models due to their high computational and storage requirements. Due to privacy, security, latency, communication bandwidth and memory requirements, processing the data through AI-enabled embedded devices supports processing locally [8] close to the sensor. Although a lot of work focuses on developing edge intelligence; still, the adoption of CNN on embedded devices is challenging due to the high computing resource requirements.

Figure 1 depicts the ultra-low latency real-time interactive surveillance systems consisting of drones or UAV equipped with intelligent cameras for surveillance, aerial photography, or infrastructure assessment. The surveillance system should identify various threats by capturing and analyzing images of surroundings and respond appropriately to make the decision. Identifying, categorizing, and visual object recognition from the pictures and getting quick insights is a fascinating field and a vital task for UAV applications.

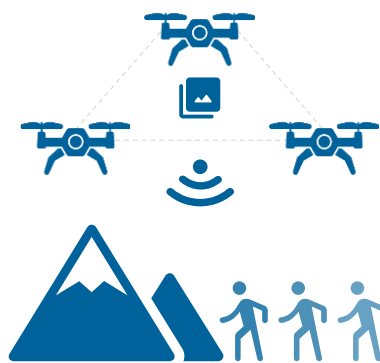


Figure 1. Ultra-low latency real time surveillance system

Our motivation for writing this review article is to exploit the potential of optimized compact deep learning model design for resource-constraint IoT gadgets like smartphones, cameras, and other devices for real-time uses, including intelligent healthcare, precision farming, and surveillance detection. The primary focus of this article is to explore the strategies for creating an optimized energy-efficient deep learning model to deploy on-edge devices with limited resources. These models offer advantages, including decreased latency, greater privacy, increased reliability, and energy efficiency. We have divided the remaining article into the following sections: section 2 provides a quick summary of edge computing and a fair analysis of the research on leveraging different deep learning frameworks for real-time object detection. Then, enabling technologies for edge intelligence are presented in section 3, a novel research methodology for creating a compact model is shown in section 4, and the conclusion is presented in section 5.

2. RELATED WORK

Optimized energy-efficient deep learning models [9]–[11] on edge are possible through model compression. Model compression refers to techniques for reducing the size of a trained model without significantly degrading its performance. As shown in Table 1 (in Appendix), recent efforts towards reducing computation and parameter storage cost through model pruning, quantization, and knowledge distillation are the most used compression approaches to compress the model without hurting original accuracy.

Pruning [11]–[15] removes redundant or unnecessary connections in a neural network. Pruning involves removing individual weights or removing entire neurons or layers. Pruning can be done either during training or after the model training. Other than pruning, eliminating the network redundancy without retraining [16], low rank approximation [17]–[21], fast Fourier transform (FFT) based convolutions [22], [23], quantization [24], binarization [25], [26], pruning [27], [28], sparsity regularization [29], [30], pruning low magnitude weights [31]–[33] are the common approaches. Knowledge distillation [34] refers to the process of training a smaller model to replicate the behavior of a larger, pre-trained model. Quantization [35]–[42] reduces the precision of the weights and activations of a neural network. In addition, quantization dramatically reduces the size of a model. It enhances performance by mapping the values to a smaller range of discrete values, such as 8-bit integers rather than 32-bit floating point values.

According to Liu *et al.* [43], pruning and fine-tuning the pre-trained model increased accuracy by 2.7% and decreased Floating point operations (FLOPs) by 20%. Furthermore, Zhang *et al.* [44] lowered inference latency and memory footprint by up to 5.79X and 14.72X, respectively, by dividing the model and the data and developing a scheduler. According to Stahl *et al.* [45], layer partitioning without fusing resulted in a 1.52X improvement in the inference task. However, much research is going on, on distributing and running a large amount of CNN on devices with limited resources. In this article, we improved the pre-trained model and created an effective inference model by splitting both the model and the data.

The current approach reduces the amount of processing and data on each edge device by partitioning and distributing layer information across numerous edge devices. Zhao *et al.* [46] suggests a system for allocating the CNN-based inference task execution among various IoT devices for concurrent task execution to reduce reaction time. They also added a workload distribution module, scheduling, and work stealing to minimize latency and memory footprint. Model compression techniques such as weight and filter pruning also perform better than uncompressed models. Although the large model structure has significant redundancy, the model's prediction accuracy is relatively unaffected after removing many model parameters. However, a recent study [47] understands that significant reduction in parameters with magnitude-based pruning significantly from the fully connected layers in the pruned networks. Model compression can help deploy machine learning models to devices with limited resources, such as mobile phones or embedded devices.

From the literature, we identify the following research gaps in deep learning at edge devices: i) there is a need for a quicker inference method for distributed heterogeneous IoT devices; ii) the deep learning model deployment on edge mobile and IoT devices consumes significant computational power and leaves a significant memory footprint; iii) more studies on inferencing methods for classification or prediction in distributed heterogeneous IoT clusters are required; iv) current methods completely ignore the possibilities of compression techniques and are limited to layer-based partitioning (to speed up inference); and v) use compression techniques and present existing methods to optimize and make the best use of the resources available to successfully implement edge intelligent systems.

3. MODEL COMPRESSION TECHNIQUES

Deep learning-based intelligent services are helpful for quick data analysis due to the gradual growth of computing methods, storage devices, IoT devices, and smartphone technology. Edge computing and artificial intelligence are combined to create edge intelligence, often called intelligent edge, to offer superior services. To accelerate the deep learning model many model compression techniques, exist.

3.1. Pruning

Pruning the convolutional layers with an equal percentage of pruning rate and retraining shows the FLOP reduction without significant loss in original accuracy. Reduce computation costs of CNNs by pruning filters with relatively small weight magnitudes without adding erratic sparsity. Pruning the fully connected layer having few parameters is challenging as it also removes weights of batch normalization subsequently.

3.1.1. Criteria for pruning

The criteria for choosing deep learning models to deploy in edge computing applications depends on the application's specific requirements. Various considerations include the number of available computing resources at the edge, the size and complexity of the model, and the required latency for model inference. Several criteria to determine which parameters or connections to prune are listed below.

- **Minimum weight:** The minimum weight criteria in pruning for CNN refers to the threshold value used to determine which weights in a neural network should be pruned or removed. This threshold is set based on the magnitude of the weights, with those that are below the threshold being considered insignificant and removed from the network. The objective of this criterion is to decrease the number of parameters in the network while preserving the overall accuracy of the model. Convolutional kernel reduces unimportant kernel values during training by using l1 and l2 regularization.
- **Activation:** The ability of the rectified linear unit (ReLU) activation function to introduce non-linearity to the network enhances the network's capacity to learn complicated features. Based on the absolute values of the network's weights, L1 regularization adds a penalty term to the loss function. This encourages the network to have fewer non-zero weights, resulting in a sparser network.
- **Mutual information:** Mutual information (MI) criteria in pruning for CNNs is a method for identifying and removing redundant or unnecessary connections in a CNN. MI measures the amount of information exchange between two variables. Mutual information is used to measure the dependence between the input and output of a CNN.

- Validation loss: The validation loss is one of the metrics used to verify the efficacy of the pruning process for optimization. The validation loss measures the variation between a given dataset's predicted and actual values. It helps to assess the network's accuracy and performance during pruning.
- Dropout: Dropout is a regularization technique [48] to set the percentage of neurons in the neural network to zero during training. An extensive neural network trained on a small training set leads to overfitting. Dropout reduces over-fitting during the retraining process by adjusting the dropout ratio. Dropout helps to prevent overfitting by forcing the network to learn more robust features that are not dependent on any specific neuron or combination of neurons.

3.2. Quantization

Quantization accelerates the deep neural network (DNN) and enables easy setup of DNN models on devices with limited resources. Furthermore, combining pruning and quantization reduces the network's inference time and space complexity by maintaining significant accuracy. Perform quantization on convolution layers of the trained model weight; to minimize the memory footprints and speed up the inference of the model. For example, the k-means clustering algorithm followed by 8-bit quantization, makes the model compact and speeds up the integer-valued weights.

3.3. Key performance indicator (metrics)

Several criteria are present for measuring the performance of a deep learning model after pruning and quantization. The commonly used metrics are accuracy, F1 score, confusion matrix, precision, recall, mean squared error, and mean absolute error. Table 2 describes the performance metrics and definitions.

Table 2. Overview of parameters used to measure performance

Metrics	Definitions
Top-1 Error	The percentage of times the classifier does not award the best score to the correct class.
Top-5 Error	The proportion of times the classifier did not use the correct class in one of its top five guesses.
Parameters of the model	Includes the convolutional layer's weights and bias, quantity, and size of kernel, as well as the number of channels of input image.
Compression rate	Rate of reduction in number of parameters of the model (convolutional layer and fully connected layer). This reduces the execution time, memory footprint.
Accuracy	Accuracy describes the correct prediction of a model across all the classes
Floating point operations (FLOPs)	Each layer in the CNN requires massive computations on the number and different sizes of feature maps and convolution kernels. Pruning reduces the least unimportant parameters and hence reduces the number of operations in the network.
Memory utilization (memory footprint)	Amount of memory required to store weights, feature maps and gradients
Time (speed up)	If reduction in computation by FLOPs indicates the decrease in inference time of the network due to the removal of the least unimportant parameters. Convolution operation, parallelization algorithm, hardware, scheduling, and memory transfer, influence the inference time.
Model compression	Model compression describes the methods used to reduce a deep learning model's network's inference time and memory footprint.
F1 score	This statistic combines precision and recall and is frequently used to assess how well a model performs on imbalanced datasets.
Confusion matrix	Measures the number of accurate predictions the model generates, including true positives, true negatives, false positives, and false negatives.
Precision	This indicator shows what percentage of true positive predictions the model generated were accurate.
Recall	This metric counts the number of actual positive occurrences the model correctly predicted would be truly positive.

4. PROPOSAL FOR COMPACT MODEL DESIGN

The need for optimized energy-efficient deep learning models on edge devices has grown in tandem with the demand for IoT applications because they provide faster prediction and better privacy because data are not transmitted over the Internet. As a result, researchers are working on optimizing deep learning models for edge devices. As shown in Figure 2, we propose a framework for optimizing a pre-trained CNN model for accelerating the inference to deploy on embedded devices. In our framework, the pre-trained model is considered, and then pruning is used to reduce the model's size and complexity by removing any unnecessary weights or filters. Quantization also speeds up and uses less memory by reducing the precision of the model's weights and activations.

Furthermore, since we combine pruning and quantization, it further reduces the model parameters. To send the task to each IoT device for parallel execution, the fused tile partitioning and distribution module involves fusing the model layers and dividing fused layers vertically. To deliver the results, the gateway device will finally combine the output from IoT devices. Hence the proposed work improves the performance

and reduces the inference latency, memory space, and communication cost. The proposed model designs an energy-efficient pre-trained CNN model, but they can also potentially impact the model’s accuracy, so it is essential to evaluate the trade-offs carefully.

The joint potential of pruning and quantization and the essence of fused tile partitioning enable compact model design. Hence the compact optimized model is suitable for distributed heterogenous embedded resource constraint edge devices as they enable real-time processing while minimizing resource usage. Furthermore, adopting federated learning and device fault tolerance mechanisms improves hardware-software ecosystems’ scalability and reliability.

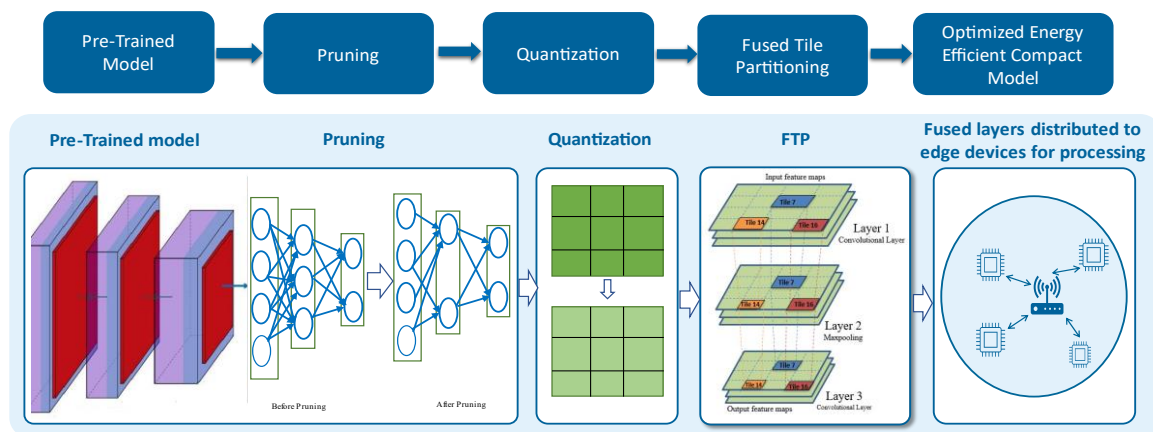


Figure 2. Proposed model of optimized energy efficient pretrained compact model to deploy on IoT devices

5. CONCLUSION

In convolutional neural networks, fully connected layer parameters highly contribute to the model size. The DNN-based model requires significantly more computation than the traditional model, which leads to high energy consumption. Unfortunately, deploying DNNs on resource-constrained devices having limited resources such as memory, computing power, and energy makes it challenging. There is a growing demand for energy-efficient deep learning models on edge devices, as these devices often have limited power and computational resources. One way to optimize the energy efficiency of a deep learning model is through pruning and quantization. In this review article, an extensive survey is carried out to understand the model reduction to reduce energy consumption during training and inference. Based on the research gap, we propose a framework to design an optimized model and work distribution strategy for lowering the energy consumption, inference time and memory footprint. The joint potential of pruning and quantization techniques followed by fused tile partitioning can significantly improve the performance of deep learning models on edge devices, making them more suitable for deployment on distributed heterogenous resource-constrained IoT devices.

APPENDIX

Table 1. State of art of CNN based model compression approaches

Papers	Year	Technique	Limitations
Pruning weights			
Murray <i>et al.</i> [12]	2015	Sparsity-inducing regularizers for pruning neurons which enables automatic sizing of neural networks.	Focus on pruning neurons via sparsity-inducing regularizers for small networks rather than parameter pruning
Srinivas and Babu [13]	2015	The author removed one redundant neuron at a time instead of removing individual weights. Applied on networks with fully connected layers to produce smaller networks.	Wiring together pairs of neurons with similar input weights, however this pruning may not reduce computation.
Naveen <i>et al.</i> [14], [15]	2021	For distributing the workload to each IoT edge device, the pre-trained model is pruned to lower the model parameter, and then fused tile partitioning is employed.	Hardware-aware hyperparameter tuning is required, along with fault-tolerant and deep neural network (DNN) containerization.

Table 1. State of art of CNN based model compression approaches (*continue*)

Papers	Year	Technique	Limitations
Eliminate the network redundancy without retraining			
Mariet and Sra [16]	2016	Create a sparse network by identifying a subset of neurons that do not require retraining and reducing network redundancy while considering the fully connected layers.	Pruning the fully connected layer may not reduce computation cost and time.
Computation cost reduction for convolutional layers			
Denil <i>et al.</i> [17]	2013	Vector quantization is used reduce the redundant parameters of the model	Complementary to dropout, maxout however cannot be used for large scale deep networks for industry.
Jaderberg <i>et al.</i> [18]	2014	Focuses on CNN evaluation speed up by using cross channel to construct low rank basis filter.	Further investigation on Separable filters in layers and filter reconstruction, model approximation during training can be explored.
Denton <i>et al.</i> [19]	2014	Explored low rank approximation and filter clustering for weight approximation and achieved speedup with marginal drop in accuracy.	Further work can be explored by considering regularization during or before training to reduce the number of parameters.
Zhang <i>et al.</i> [20]	2015	Accelerates CNN computation considering nonlinear filters with low-rank decomposition without stochastic gradient descent (SGD).	Instead of filter weight consideration, channel pruning results reduces inference time.
Ioannou <i>et al.</i> [21]	2016	Low rank representation of CNN with weight initialization novel scheme is used reduce the computation.	Channel redundancy reduction, low dimensional embedding further enables the development of computationally efficient CNN model.
FFT based convolutions to reduce the convolutional overheads			
Mathieu <i>et al.</i> [22]	2013	FFT based convolutions	Need to explore performance for the impact of image and kernel size. For FFT based implementation input image should be a power of 2 else padded to next power.
Lavin and Gray [23]	2016	Winograd algorithm for FFT-based fast convolutions	Winograd algorithm needs specialized processing based on the size of the filter and block
Techniques for reducing model size and computing overheads			
Han <i>et al.</i> [24]	2016	Quantization	Performs quantization, due to libraries can't access matrix lookup and may require software and hardware solution for architectural design.
Rastegari <i>et al.</i> [25]	2016	Binarization	Though developed a small efficient network can focus on specific resource constraint such as latency, memory for applications.
Coubariaux and Bengio [26]	2016	Binarization reduces the model size and lowers the computation overhead	Multiplication during training can reduce and computation and needs to generalize other classification and detection models for other datasets.
Removing the feature maps from well-trained network			
Anwar <i>et al.</i> [27]	2015	The candidates' weights and locations are pruned three times using particle filtering, which selects the best combination of many randomly generated masks.	Knowledge based pruning further may reduce computational complexity.
Polyak and Wolf [28]	2015	Detecting less frequently activated feature maps with sample input data and reduces feature map from trained network.	Analyzing magnitude-based filter weights and pruning the filter and feature map reduces the complexity.
Training CNN with sparse constraints			
Lebedev and Lempitsky [29]	2016	Group-wise elimination of convolution kernel to leverage on achieving group-sparsity of the convolutional filters	Regularizers based normalization to have sparse network than pruning to accelerate in practice, the actual speed-up depends on implementation
Zhou <i>et al.</i> [9]	2016	During the training process, introducing group-sparse regularization on convolutional filters to reduce the filters.	Regularization may need a greater number of iterations to converge
Wen <i>et al.</i> [31]	2016	To eliminate unnecessary filters, channels, or even layers, this article adds structured sparsity regularization to each layer.	Layer wise parameter regularization increases complexity and can be avoided to speedup computation
Pruning low magnitude weights			
Maxwell <i>et al.</i> [32]	2014	Using regularizers during training of CNNs to remove the connections of convolution and the fully connected layer having non-zero values leading to a sparse deep network.	The refined sparsity process can further reduce the parameters of the model to optimize memory conservation.
Han <i>et al.</i> [33]	2015	The author performed magnitude-based weight trimming on filters with weights below the specified threshold. To create a sparse network, carefully tune the threshold to determine how many filters must be pruned.	Parameter pruning and sharing can be applied for various DNNs, for energy consumption the number of weights alone cannot considered.
The knowledge distillation technique			
Hinton <i>et al.</i> [34]	2015	As the deployment of small student models is simple on resource constrained internet of things hardware devices, the knowledge distillation technique transfers knowledge from the large teacher model to the student model to reduce computing costs.	Lacks knowledge distillation when neural network is very big.

Table 1. State of art of CNN based model compression approaches (*continue*)

Papers	Year	Technique	Limitations
Pruning-Quantization			
Benoit <i>et al.</i> [35]	2017	Proposed integer-only arithmetic based quantization scheme than floating point inference.	The complete inference task is carried out with integer arithmetic without considering floating point dequantization and limited to ReLU activation functions.
Shaokai <i>et al.</i> [36], [37]	2018, 2019	Developed a framework of DNN performing weight pruning and clustering/quantization along with iterative weight clustering training, centroid update, and weight clustering to enhance the model's performance.	Pruning and quantization is considered in optimized framework, however progressive weight pruning may yield better results.
Yuan <i>et al.</i> [38]	2019	A memristor-based framework is proposed that considers weight quantization and pruning while training a DNN.	Weight pruning followed by quantization on VGG-16 and ResNet-18 reduces power reduction, however framework focuses on classification/detection model ignoring generative models.
Muhamad <i>et al.</i> [39]	2020	Used DeepLIFT for pruning DNN to prune filters and the weights of the fully convolutional layer, followed by clustering-based quantization of DNN weights. Also, integer-based mixed-precision quantization for varying number of integer bits of each layer of DNN.	For pruning and quantization using Explainable Artificial Intelligence to process the dataset may further improves the network performance
Hu <i>et al.</i> [40]	2021	Proposed one-shot pruning-quantization, which compresses through pre-trained wight parameters. During fine-tuning weight parameters are updated. Further, channel-wise quantization for each layer having common codebook still reduces bitrate.	DNN model compression and practical implementation on custom hardware platforms helps to validate the inference efficiency.
Zeng <i>et al.</i> [41]	2022	Proposed 8-bit quantization technique using tanh on dense layer weights followed by linear quantization on rest of the network.	Graphics processing unit (GPU) based computation may further reduce the computation speed and needs to perform practical hardware implementation.
Ma <i>et al.</i> [42]	2022	Proposed improved ADMM-NN, a joint weight pruning and quantization framework.	AI based applications are benefited with network pruning than block based DNN pruning using regularization.

ACKNOWLEDGEMENTS





The authors are grateful to REVA University for supporting the research facilities.

REFERENCES





- [1] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: the confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020, doi: 10.1109/JIOT.2020.2984887.
- [2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019, doi: 10.1109/JPROC.2019.2918951.
- [3] S. Naveen and M. R. Kounte, "Distributing the cloud into fog and edge: new weather in IoT based deep learning," in *Proceedings of the 2nd International Conference on Recent Trends in Machine Learning, [IoT], Smart Cities and Applications*, Springer Nature Singapore, 2022, pp. 749–758.
- [4] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: multiaccess edge computing for 5G and internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020, doi: 10.1109/JIOT.2020.3004500.
- [5] A. Singh, S. C. Satapathy, A. Roy, and A. Gutub, "AI-based mobile edge computing for IoT: applications, challenges, and future scope," *Arabian Journal for Science and Engineering*, vol. 47, no. 8, pp. 9801–9831, Aug. 2022, doi: 10.1007/s13369-021-06348-2.
- [6] A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque, "Artificial intelligence-driven mechanism for edge computing-based industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4235–4243, Jul. 2019, doi: 10.1109/TII.2019.2902878.
- [7] X. Chen, Q. Shi, L. Yang, and J. Xu, "ThriftyEdge: resource-efficient edge computing for intelligent IoT applications," *IEEE Network*, vol. 32, no. 1, pp. 61–65, Jan. 2018, doi: 10.1109/MNET.2018.1700145.
- [8] S. Naveen and M. R. Kounte, "Machine learning at resource constraint edge device using bonsai algorithm," in *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAIECC)*, Dec. 2020, pp. 1–6, doi: 10.1109/ICAIECC50550.2020.9339514.
- [9] M. Kumar, X. Zhang, L. Liu, Y. Wang, and W. Shi, "Energy-efficient machine learning on the edges," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2020, pp. 912–921, doi: 10.1109/IPDPSW50202.2020.00153.
- [10] F. Daghero, D. J. Pagliari, and M. Poncino, "Energy-efficient deep learning inference on edge devices," in *Advances in Computers*, Elsevier, 2021, pp. 247–301.
- [11] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6071–6079, doi: 10.1109/CVPR.2017.643.
- [12] K. Murray and D. Chiang, "Auto-sizing neural networks: with applications to n-gram language models," *arXiv preprint arXiv:1508.05051*, Aug. 2015.
- [13] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *arXiv preprint arXiv:1507.06149*, Jul. 2015.
- [14] S. Naveen, M. R. Kounte, and M. R. Ahmed, "Low latency deep learning inference model for distributed intelligent IoT edge clusters," *IEEE Access*, vol. 9, pp. 160607–160621, 2021, doi: 10.1109/ACCESS.2021.3131396.
- [15] S. Naveen and M. R. Kounte, "Memory optimization at edge for distributed convolution neural network," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 12, Dec. 2022, doi: 10.1002/ett.4648.

- [16] Z. Mariet and S. Sra, "Diversity networks: neural network compression using determinantal point processes," *arXiv preprint arXiv:1511.05077*, Nov. 2015.
- [17] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," *Advances in neural information processing systems*, vol. 26, 2013.
- [18] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, May 2014.
- [19] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *Advances in neural information processing systems*, vol. 27, 2014.
- [20] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1943–1955, Oct. 2016, doi: 10.1109/TPAMI.2015.2502579.
- [21] Y. Ioannou, D. Robertson, J. Shotton, R. Cipolla, and A. Criminisi, "Training CNNs with low-rank filters for efficient image classification," *arXiv preprint arXiv:1511.06744*, Nov. 2015.
- [22] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," *arXiv preprint arXiv:1312.5851*, Dec. 2013.
- [23] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 4013–4021, doi: 10.1109/CVPR.2016.435.
- [24] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, Oct. 2015.
- [25] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Computer Vision ECCV 2016*, Springer International Publishing, 2016, pp. 525–542.
- [26] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.
- [27] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 3, pp. 1–18, Jul. 2017, doi: 10.1145/3005348.
- [28] A. Polyak and L. Wolf, "Channel-level acceleration of deep face representations," *IEEE Access*, vol. 3, pp. 2163–2175, 2015, doi: 10.1109/ACCESS.2015.2494536.
- [29] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2554–2564, doi: 10.1109/CVPR.2016.280.
- [30] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: towards compact CNNs," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2016, pp. 662–677.
- [31] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *arXiv:1608.03665*, Aug. 2016.
- [32] M. D. Collins and P. Kohli, "Memory bounded deep convolutional networks," *arXiv preprint arXiv:1412.1442*, Dec. 2014.
- [33] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *Advances in Neural Information Processing Systems*, pp. 1135–1143, Jun. 2015.
- [34] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, Mar. 2015.
- [35] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 2704–2713, doi: 10.1109/CVPR.2018.00286.
- [36] S. Ye *et al.*, "A unified framework of DNN weight pruning and weight clustering/quantization using ADMM," *arXiv preprint arXiv:1811.01907*, Nov. 2018.
- [37] S. Ye *et al.*, "Progressive weight pruning of deep neural networks using ADMM," *arXiv preprint arXiv:1810.07378*, Oct. 2018.
- [38] G. Yuan *et al.*, "An ultra-efficient memristor-based DNN framework with structured weight pruning and quantization using ADMM," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Jul. 2019, pp. 1–6, doi: 10.1109/ISLPED.2019.8824944.
- [39] M. Sabih, F. Hannig, and J. Teich, "Utilizing explainable AI for quantization and pruning of deep neural networks," *arXiv preprint arXiv:2008.09072*, Aug. 2020.
- [40] P. Hu, X. Peng, H. Zhu, M. M. S. Aly, and J. Lin, "OPQ: compressing deep neural networks with one-shot pruning-quantization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, pp. 7780–7788, May 2021, doi: 10.1609/aaai.v35i9.16950.
- [41] L. Zeng *et al.*, "Sub 8-bit quantization of streaming keyword spotting models for embedded chipsets," in *Text, Speech, and Dialogue*, Springer International Publishing, 2022, pp. 364–376.
- [42] X. Ma *et al.*, "BLCR: towards real-time DNN execution with block-based reweighted pruning," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, Apr. 2022, pp. 1–8, doi: 10.1109/ISQED54688.2022.9806237.
- [43] B. Liu, Y. Cai, Y. Guo, and X. Chen, "TransTailor: pruning the pre-trained model for improved transfer learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, pp. 8627–8634, May 2021, doi: 10.1609/aaai.v35i10.17046.
- [44] S. Zhang, S. Zhang, Z. Qian, J. Wu, Y. Jin, and S. Lu, "DeepSlicing: collaborative and adaptive CNN inference with low latency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2175–2187, Sep. 2021, doi: 10.1109/TPDS.2021.3058532.
- [45] R. Stahl, A. Hoffman, D. Mueller-Gritschneider, A. Gerstlauer, and U. Schlichtmann, "DeeperThings: fully distributed CNN inference on resource-constrained edge devices," *International Journal of Parallel Programming*, vol. 49, no. 4, pp. 600–624, Aug. 2021, doi: 10.1007/s10766-021-00712-3.
- [46] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018, doi: 10.1109/TCAD.2018.2858384.
- [47] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," *arXiv preprint arXiv:1608.08710*, Aug. 2016.
- [48] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, Jul. 2012.

BIOGRAPHIES OF AUTHORS

Soumyalatha Naveen     is awarded with the MTech degree from Visvesvaraya Technological University, India. She is a Research Scholar at School of Computer Science and Engineering, REVA University, Bengaluru, India. Edge computing, internet of things, deep learning, and intelligent IoT systems are some of her key research interests. She can be reached at email: soumyanaveen.u@gmail.com.



Manjunath R. Kounte     graduated from VTU in Belagavi, Karnataka, India, with a Bachelor of Engineering in Electronics and Communication Engineering in 2007, a Master of Technology in Computer Network Engineering in 2010, and a Ph.D. from JAIN University in 2017. He is presently employed with REVA University in Bangalore, India, where he serves as Associate Professor and Head of the Electronics and Computer Engineering Department. His areas of interest in research include blockchain technologies, internet of vehicles, edge computing for the internet of things, and machine learning. He can be reached at email: kounte@reva.edu.in.