Efficient Neural Network Reduction for AI-on-the-edge Applications through Structural Compression

Adriano Puglisi, Flavia Monti, Christian Napoli and Massimo Mecella
Department of Computer, Control, and Management Engineering Antonio Ruberti
Sapienza Università di Roma, Rome, Italy
Email: {puglisi, monti, napoli, mecella}@diag.uniroma1.it

***Abstract*—Modern neural networks often rely on overparameterized architectures to ensure stability and accuracy, but in many real-world scenarios, large models are unnecessarily expensive to train and deploy. This is especially true in Internet of Things (IoT) and edge computing scenarios, where computational resources and available memory are severely limited. Reducing the size of neural networks without compromising their ability to solve the target task remains a practical challenge, especially when the goal is to simplify the architecture itself, not just the weight space. To address this problem, we introduce ImproveNet, a simple and general method that reduces the size of a neural network, without compromising its ability to solve the original task. The approach does not require any pre-trained model, specific architecture knowledge, or manual tuning. Starting with a standard-sized network and the standard training configuration, ImproveNet verifies the model's performance during training. Once the performance requirements are met, it reduces the network by resizing feature maps or removing internal layers, thus making it ready for AI-on-the-edge deployment and execution.**

***Index Terms*—IoT, Edge AI, Deep Model Optimization, Neural Network Compression**

I. Introduction

The Internet of Things (IoT) concept, first introduced by Ashton in 1999, describes a system in which physical objects equipped with sensors are connected to the Internet and used to collect data from the environment. Since then, the idea has evolved rapidly and today includes billions of devices that can communicate with each other and exchange information in real time. There are currently over seven billion IoT devices in the world, and this number is expected to exceed twenty billion in the coming years.

As the number of connected devices increases exponentially, the amount of data generated is also growing. Although this data may contain useful information, it is often affected by noise, redundancies or errors [1]. As a result, traditional processing methods are no longer sufficient and increasing machine learning techniques are being used to extract knowledge from collected data. However, running machine learning models directly on IoT devices is extremely complex. These systems, also called *edge devices*, are characterized by limited resources, limited memory, low computational power, and stringent energy constraints. For these reasons, running large models locally (*on-device*) is often impractical.

An alternative solution could be to send the data to a remote server (*cloud*), where heavier models can be run without hardware constraints. However, in many real-world scenarios this option is limited or unacceptable, either for latency reasons (such as in real-time applications) or for privacy reasons (in healthcare, industrial, or personal contexts). In these scenarios, keeping the computation local is the only sustainable choice, provided that the model is light enough to be run safely and efficiently on the device.

To address this trend towards *AI-on-the-edge* (a.k.a. Edge AI), we propose ImproveNet, a simple method that reduces the size of a neural network directly during training, while ensuring that the model maintains the required performance. The approach starts with complete architecture, which is then progressively reduced as training progresses.

Block removal and filter reduction are the two structural alterations that lead to reduction. Removing a block means eliminating entire sequential portions of the network, each composed of one or more convolutional or linear layers, with a direct impact on the depth of the model. Filter reduction, on the other hand, consists in decreasing the number of output channels in the convolutional or dense layers, with the effect of reducing the width of the intermediate representations. Both operations allow us to simplify the architecture in a controlled way, keeping the capacity of the model within acceptable thresholds.

In the following sections, we describe in detail how this strategy works and analyze its application in different scenarios. In particular, Section II presents the main existing works dedicated to the reduction of neural networks through pruning, quantization or architectural simplification. Section III introduces the logic of ImproveNet and how reduction operations are integrated into the training cycle. Finally, in Section IV, we report the experimental results obtained by applying the method to the ESA-ADB dataset, using two autoencoders, one linear and one fully convolutional.

II. Related Works

In our application context, oriented to industrial scenarios and constrained by efficiency and compatibility requirements with edge systems, requirements analysis has led to narrowing the focus to fully convolutional or linear networks. Despite this choice, the comparison with existing compression techniques

Manuscript received May 19, 2025; revised August 27, 2025; accepted July 25, 2025. Published September 2, 2025.
Issue category: Special Issue on DSD/SEAA 2025 on Works in Progress (WiP) Session, Salerno, Italy, Sept. 2025.
Paper category: Short
DOI: doi.org/10.64552/wipiec.v11i1.89

TABLE 1 - COMPARISON OF MODEL COMPRESSION TECHNIQUES

Property	**ImproveNet**	**Structured Pruning** [7] [8]	**Unstructured Pruning** [9] [10]	**Distillation** [11]
Reduction type	Structural	Structural	Sparse	Knowledge transfer
Granularity	Blocks, Channels, Neuron	Filters	Weights	-
Inference time	Reduced	Reduced	Same	Reduced
Memory footprint	Reduced	Reduced	Same	Reduced
Compression ratio	High	High	Same	Reduced
Performances retention	Preserved	Not guaranteed	Not guaranteed	Teacher-dependent
Loss stability	Controlled	Requires retraining	Requires retraining	Regularized
Training Time	High	High	Low	Low
Architecture agnostic	Yes	No	No	Yes
Self contained	Yes	Yes	Yes	No
Repeatability	Yes	[7] Available in Caffe (Python) [8] Available in PyTorch	[9] Not available [10] Official not available (3rd part)	[11] Official not available (3rd part)
Neural Network Supported	Linear & Fully Convolutional	CNN	CNN	Any

will be conducted in a fair way, highlighting for each approach the context of validity and the reference architectures to which it applies. Several methods have been proposed to reduce the size and complexity of neural networks, particularly in applications with limited computational or memory resources, such as embedded or IoT devices. Most existing techniques are based on static approaches, including quantization [2], pruning [3], and knowledge distillation [4] [5], and are typically applied after training. Table 1 summarizes the main differences between ImproveNet and other model compression techniques, i.e., Structured Pruning, Unstructured Pruning, and Distillation.

III. IMPROVENET

Unlike traditional techniques based on neural importance, induced sparsity in weights, or post-training strategies, ImproveNet takes a completely different approach. The method acts directly during the training process, progressively reducing the network only when performance reaches predefined thresholds. This reduction occurs without requiring a fully trained model or the use of external heuristics.

The method takes as input the initial model together with all the components needed for training, such as the dataset, the optimizer, the metrics estimator, the loggers, and the performance constraints to be achieved such as the loss $\mathcal{L}_{target}$ and the accuracy $\mathcal{M}_{target}$. At regular intervals, the optimizer checks whether the current model M satisfies the target requirements. These requirements are expressed as constraints on global quantities, such as the loss $\mathcal{L}(M)$ and the accuracy metric $\mathcal{M}(M)$, as formalized in the following equation

$$\mathcal{L}(M) \leq \mathcal{L}_{target} \land \mathcal{M}(M) \geq \mathcal{M}_{target}$$

If the conditions are satisfied, ImproveNet applies a structural transformation to the network, choosing between reducing the number of channels and removing an internal block.

The type of reduction applied is managed dynamically based on the size of the current model compared to the initial one. The first two attempts apply a channel reduction and a simple block removal respectively. After this, the method computes the reduction ratio between the current network and the original one. If this ratio is still higher than the first threshold, priority is given to block removal, alternating every three attempts with a filter reduction. When the ratio drops below the first but above the second threshold, the two types of reduction are alternated more frequently (once every two attempts). Finally, only channel reduction is performed below the second threshold, preventing further excessive structural eliminations.

Using this technique, the model can gradually reduce while preserving its structural balance and avoiding excessive compression in subsequent training phases. To avoid repeated or harmful activities, the system also considers the number of reductions performed previously.

An additional protection mechanism is activated in case the network starts to stagnate. If the model does not converge within a certain number of iterations and the reduction attempts exceed half of the maximum expected number, ImproveNet performs a controlled reallocation of the architecture. The goal is to prevent excessive compression from trapping the model in non-ideal local minima.

Finally, if at the end of a reduction cycle the model fails to stably maintain the convergence conditions, the system restores

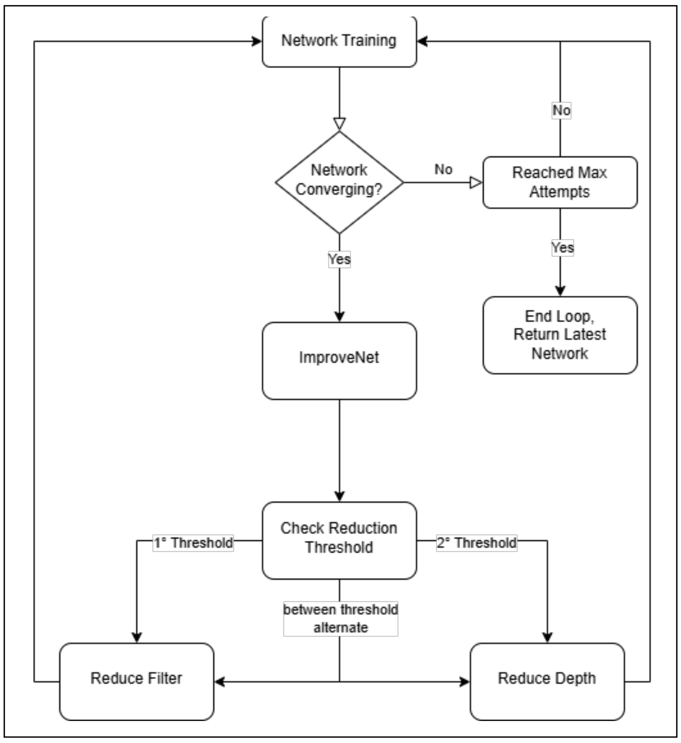

Figure 1 - Schematic representation of the ImproveNet workflow.

the last effective configuration, i.e. the last architecture that satisfied the performance constraints. An alternative strategy is then adopted, for example switching from block removal to filter reduction, or vice versa. This rescue mechanism ensures that the compression process does not irreversibly compromise the optimization capacity of the network.

The described structural transformations are based on the reduction of the number of channels (i.e. of input and output channels) and on the removal of entire blocks, which include sequences of convolutional or linear layers. Both operations take into account both the number of attempts already made and the size of the current model compared to the original one, Figure 1 shows the full workflow underlying the ImproveNet procedure, highlighting how the system monitors the training progress and applies structural reductions when the target conditions are met.

IV. Preliminary Results

To evaluate the performance of ImproveNet, one approach could have been a comparison with the methods in Table 1, but since most of them did not provide official source code or were not implemented in PyTorch or the code was available but did not work properly, we decided to test our method using ESA-ADB dataset [6], a recognized benchmark for multivariate time-series anomaly detection based on real data from space missions. We chose this dataset because it is representative of an edge environment where there is a need for small and compact models, suitable for resource constrained environments and a real-time operational context. The dataset is composed of three missions from which we selected Mission 1, composed of 76 channels, 58 of which are target channels and are splitted into 4 subsystems. Mission 1 includes 200 annotated events where 118 are anomalies, 78 nominal rare events (atypical but expected telemetry variations), and 4 communication gaps. We conducted experiments on a lightweight subset consisting of channels 41 to 46 as suggested by [6]. These channels were selected because they contain interesting but manageable anomalies, are useful for monitoring the health of the satellite, are relatively easy to visualize and analyze manually, and are independent of other channels or subsystems. The data were normalized in the range [0, 1] using a Min-Max scaling channel by channel, to ensure uniformity between the signal scales and avoid distortions in the calculation of the loss function.

The data were split respecting the temporal order of the observations where 70% was used for training, while the remaining 30% was divided into equal parts for validation and testing. The anomalous pattern is present exclusively in the test set, to train the model on the reconstruction of normal conditions. The time series were then transformed into fixed-length windows of 50 samples, with a stride of 50, obtaining sequences of the type (batch, 50, 6), where 6 represents the number of channels.

Training was conducted for a maximum of 100 epochs, using the Adam optimizer with a learning rate of 0.0001. The objective function used is a weighted combination of mean squared error (MSE) and mean absolute error (MAE), defined as

$$\mathcal{L} = \alpha \cdot MSE + (1 - \alpha) \cdot MAE$$

where ($\alpha = 0.5$) represents the balance between the two components. This formulation allows to penalize both large point errors (that the MAE effectively intercepts) and moderate diffuse errors (captured by the MSE), resulting particularly suitable for anomaly detection tasks.

We considered a Fully Convolutional Autoencoder (FCAE) and a Linear Autoencoder (LAE). ImproveNet was applied to these models, which operated during training by progressively reducing their structural complexity through functional criteria, generating compressed versions capable of maintaining comparable performance in terms of predictive accuracy.

All the experiments were run using an Intel core I5-13600KF, 32 GB of RAM and an RTX 3060 with 12 GB of VRAM. The results obtained show a significant reduction in the size of the models, in the case of the convolutional autoencoder, as shown in Table 2, the number of parameters goes from 130,886 in the original version to only 6,734 in the compressed network, with a reduction ratio of 94.85%, a reduction in the inference time from 3.4 ms to about 1 ms (3.4x faster) and a reduction in memory footprint of 94.07%. Similarly, in the linear model, as shown in Table 2, the parameters drop from 244,972 to 15,284 (93.76% reduction), and the inference time is reduced from 1.56 ms to about 0.15 ms (10.4x faster) and a memory footprint reduced of 93.37%.

Figure and Figure display anomaly detection results on a test sequence for linear autoencoder and convolutional models, respectively. In both situations, it is noted that the smaller

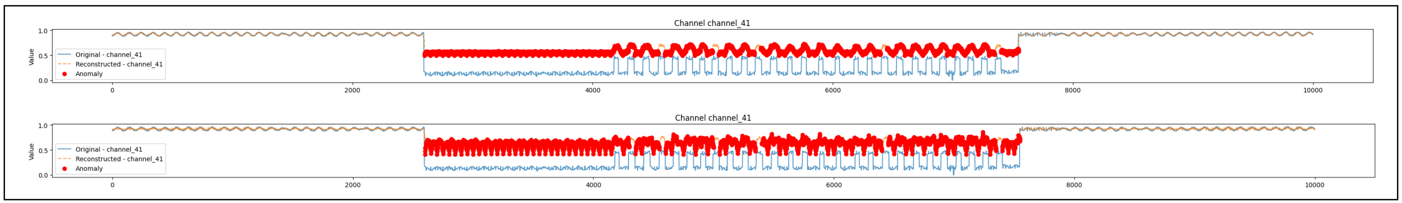

Figure 2 - Anomaly detection on a test sequence by the original convolutional model (the upper one) and reduced one (the lower one).

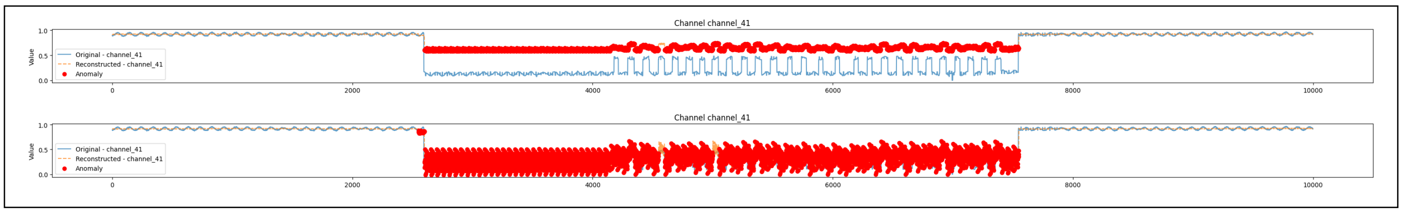

Figure 3 - Visual comparison between the original linear autoencoder (the upper one) and the reduced linear autoencoder model (the lower one).

TABLE 2 – COMPARISON BETWEEN THE LARGE AND THE REDUCED FULLY CONVOLUTIONAL AUTOENCODER (FCAE) AND LINEAR AUTOENCODER (LINEAR AE) ARCHITECTURES

	Total Param #	**CPU Inference Time (ms)**	**Size (MB)**
FCAE	130,886	3.4073	0.5046
Reduced FCAE	6,734	1.019	0.0299
Linear AE	244,972	1.5676	0.9397
Reduced Linear AE	15,284	0.1508	0.0623

version of the network (the lower time series in both images), parameters. For the convolutional model, the compressed obtained using ImproveNet, retains the ability to accurately detect abnormal patterns, despite the reduction in the number of network shows a slightly lower reconstruction quality than the original network, but the ability to detect anomalies remains unchanged, with comparable predictive performances. Similarly, in the case of the linear autoencoder, a slight degradation of the reconstruction is observed, but the anomaly is still correctly identified.

V. CONCLUSIONS

In this paper, we demonstrated the effectiveness of our approach for convolutional and linear networks in AI-on-the-edge scenarios where the size of the model is a central constraint. The ability of ImproveNet to progressively reduce architectural complexity while maintaining stable performance makes it particularly suitable for use in systems where the trade-off between accuracy and computational efficiency is essential.

In addition to the space/satellite, similar applications are found in sectors such as autonomous robotics, distributed industrial monitoring systems, wearable biomedical devices, and IoT infrastructures, all of which share the need to run neural models under limited computation and energy constraints.

A future development consists of directly integrating the compressed models generated by ImproveNet into real devices, evaluating their behavior on embedded hardware and low-power microcontrollers, in unsimulated operating conditions.

ACKNOWLEDGMENT

The work of Adriano Puglisi is supported by Thales Alenia Space and Regione Lazio, through the fellowships (CUP B83C23001060009), (SIGEM 22066DP000000035) *AI-based digital industrial processes for space systems production workflow.*

REFERENCES

[1] H. N. W. R. C. W. W. H. Z. Z. &. V. A. V. Dai, "Big data analytics for large-scale wireless networks: Challenges and opportunities," *ACM Computing Surveys (CSUR),* pp. 1-36, 2019.

[2] B. K. S. C. B. Z. M. T. M. H. A. .. &. K. D. Jacob, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 2704-2713, 2018.

[3] H. K. A. D. I. S. H. &. G. H. P. Li, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710,* 2016.

[4] T. G. I. &. S. J. Chen, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641,* 2015.

[5] R. C. a. A. N.-M. C. Buciluˇa, "Model compression," *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining,* p. 535–541, 2006.

[6] K. H. C. A. J. R. B. N. J. L. D. .. &. D. C. G. Kotowski, "European space agency benchmark for anomaly detection in satellite telemetry," *arXiv preprint arXiv:2406.17826,* 2024.

[7] J. H. Z. H. Z. H. Y. X. C. W. W. J. &. L. W. Luo, "ThiNet: Pruning CNN filters for a thinner net," *IEEE transactions on pattern analysis and machine intelligence,* vol. 41, no. 10, pp. 2525-2538, 2018.

[8] Y. K. G. D. X. F. Y. &. Y. Y. He, "Soft filter pruning for accelerating deep convolutional neural networks," *arXiv preprint arXiv:1808.06866,* 2018.

[9] S. &. B. R. V. Srinivas, "Data-free parameter pruning for deep neural networks," *arXiv preprint arXiv:1507.06149,* 2015.

[10] S. P. J. T. J. &. D. W. Han, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems,* vol. 28, 2015.

[11] G. V. O. &. D. J. Hinton, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531,* 2015.