Propagated Gradient Based Learnable Dynamic Compression and Energy Optimisation for Edge AI on Microcontrollers

MOHAMMED S. ALZIYAD
dept. of Computer Science, Shaqra University
Marconi Lab, International Centre for Theoretical Physics
Trieste, Italy
m.alziyad911@icloud.com

Rytis Paskauskas
Marconi Lab, International Centre for Theoretical Physics
Trieste, Italy
rytis.paskauskas@ictp.it

***Abstract*—ML models demand substantial computation, memory, and energy resources, hindering their deployment on edge devices with limited hardware [1], [2]. To tackle these constraints, we introduce a state-of-the-art technique, "S More Than You Need," a propagated-gradient-based learnable dynamic compression method that obtains a subnetwork performing as well as the full network, reducing size by 35× to 88× without loss of accuracy [3]–[5]. Our method guides the gradient toward convergence, providing optimal accuracy and compact size. We reinforce our work with long-term solutions such as motion sensors to extend battery life and reduce computing load, and lightweight communication protocols such as MQTT and LoRa [2]. Additionally, we assess our methods on several microcontrollers, providing reliable performance across models such as LeNet [6], ResNet [7], MobileNet [8], and LightNN.**

I. Introduction

Neural networks dominate current state-of-the-art edge devices [1]. These neural networks demand substantial computation, memory, and energy resources. For instance, the VGG-19 and ResNet-152 use over 200 MB and require over 12 GFLOPs per inference [7], [9]. Lately, efficient architectures such as MobileNet and ShuffleNet reduce computation for mobile devices [8], [10]. Despite these advances, edge devices are limited to nearly 1 MB, and these models consume power rapidly due to inference computational requirements [2].

To achieve this goal, we introduce **S**, a learnable parameters that can find enable neural networks to identify useful neurons and form optimal subnetwork that achieve comparable performance to the original network, allowing compression and training to proceed in parallel.

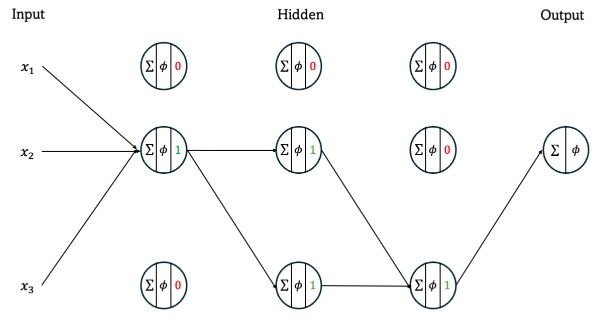

Fig. 1. **SNeural network** works to minimize the loss function and number of neurons. Using S parameter for every neuron.

II. Literature Review

In [3], proposed a method to prune redundant connections from neural networks using a three-step method. First, we train the network to learn which connections are important. Next, we prune the unimportant connections. Finally, we retrain the network to fine tune the weights of the remaining connections. On the ImageNet dataset, the method reduced the number of parameters of AlexNet by a factor of 9×, from 61 million to 6.7 million, without incurring accuracy loss. Similar experiments with VGG-16 found that the total number of parameters can be reduced by 13×, from 138 million to 10.3 million, again with no loss of accuracy.

This is complemented by findings in [4], where they introduce "deep compression", a three-stage pipeline:

pruning, trained quantization and Huffman coding, that work together to reduce the storage requirement of neural networks by 35× to 49× without affecting their accuracy. The pipeline first prunes the network by learning only the important connections. Next, quantizes the weights to enforce weight sharing, and finally, applies Huffman coding. The network is retrained after the first two steps to fine-tune the remaining connections and quantized centroids. Pruning reduces the number of connections by 9× to 13×; quantization then reduces the number of bits that represent each connection from 32 to 5. On the ImageNet dataset, the method reduced the storage required by AlexNet by 35×, from 240MB to 6.9MB, without loss of accuracy. The method also reduced the size of VGG-16 by 49× from 552MB to 11.3MB, again with no loss of accuracy. When benchmarked on CPU, GPU and mobile GPU, the compressed network achieved 3× to 4× layerwise speedup and 3× to 7× better energy efficiency.

Additionally, Frankle and Carbin examine the fundamental principles of network pruning in [5]. Their study confirms that pruning techniques can reduce the parameter counts of trained networks by over 90% without compromising accuracy, thus decreasing storage requirements and improving computational performance of inference. The study finds that a standard pruning technique naturally uncovers subnetworks that, when initialized with their original weights, are capable of training effectively. Based on these results, they articulate the lottery ticket hypothesis: dense, randomly-initialized networks contain subnetworks (winning tickets) that when trained in isolation reach test accuracy comparable to the original network in a similar number of iterations. They present an algorithm to identify these winning tickets, which are consistently found to be less than 10-20% of the size of the original network for several architectures on MNIST and CIFAR10. Above this size, the winning tickets learn faster than the original network and can reach higher test accuracy.

Despite the advancements discussed in [3]–[5], there are several gaps remain, such as computation cost of compression, misunderstanding the lottery ticket hypothesis, where we applied the concept and empower the backpropagation to reconstruct the architecture to obtain the winning-lottery.

III. SNeuron

Traditional neurons perform a weighted summation followed by an activation function to break linearity. In **SNeuron**, an additional component $H(\sigma(S))$ defines the operational state of the neuron. This component produces a binary value that, when multiplied with the activation term, acts as a gate controlling information flow.

The SNeuron is applied to the entire network to minimize the number of active neurons. Consequently, some neurons handle multiple tasks, while removing those associated with useless tasks helps the network become more generalized and compact.

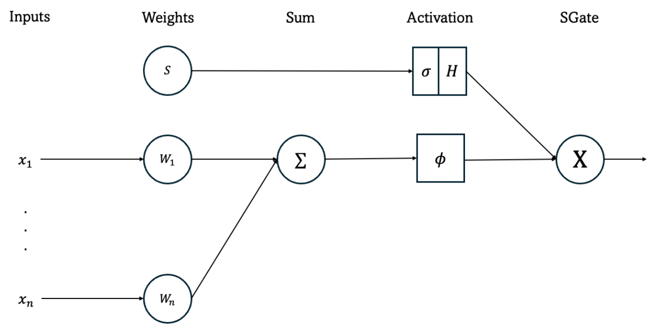

Fig. 2. **SNeuron** includes a term that acts as a binary gate, selecting the state of the current neuron and allowing the gradients to activate beneficial neurons.

IV. SLoss

The loss function of S consists of two terms, the original loss and the S loss. These terms are combined to compute gradients that minimize the network:

$$L_{\text{total}} = T_{\text{original}} \cdot \left(\frac{1}{N} \sum H(\sigma(S)) \right) + T_{\text{original}}$$

This formulation encourages both factors to cooperate in identifying an optimal subnetwork that corresponds to the winning ticket. The non-differentiability issue of H is addressed in PyTorch by detaching H during backpropagation, which prevents gradient flow through this operation.

V. Energy Optimisation

Energy efficiency is a critical requirement for deploying AI models on microcontrollers. We implemented two strategies to extend device lifetime and reduce dependence on external charging.

A. Motion Sensor

Fig. 3. **Mini PIR motion detector** used for motion sensing and power optimization.

We evaluated the impact of adding a Passive Infrared (PIR) motion sensor to trigger the microcontroller only when activity was detected. The following assumptions were used:

- Active mode current: 120 mA
- Sleep mode current: 5 mA

- Daily activity: 2 hours active, 22 hours sleep

Without the PIR sensor, the device remains active for 24 hours, leading to:

$$E_{\text{no PIR}} = 120\,\text{mA} \times 24\,\text{h} = 2880\,\text{mAh/day}$$

With the PIR sensor:

$$E_{\text{with PIR}} = (120\,\text{mA} \times 2\,\text{h}) + (5\,\text{mA} \times 22\,\text{h})$$

$$= 240 + 110 = 350\,\text{mAh/day}$$

The energy saving is therefore:

$$\Delta E = 2880 - 350 = 2530\,\text{mAh/day}$$

$$\%\text{Reduction} = \frac{2530}{2880} \times 100 \approx 88\%$$

This demonstrates that motion-triggered operation reduces daily consumption by nearly $9\times$, achieving close to 85–90% savings.

B. Solar Panel

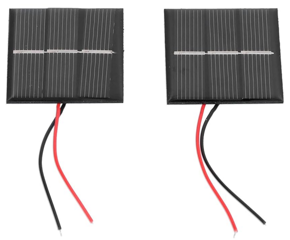

Fig. 4. **Solar panel** used for recharging the microcontroller battery

A compact solar panel was added to recharge the battery during operation. The panel provides an average output of 200 mA for about 3 hours of effective sunlight each day, supplying roughly 600 mAh of energy. The daily device consumption with motion-triggered operation is about 350 mAh. Since the harvested energy exceeds the consumption, the battery remains charged. This balance allows the system to operate continuously without the need for external charging or frequent battery replacement, making it suitable for long-term use in remote or off-grid environments.

VI. Lightweight Communication Protocols

Efficient communication is critical for edge AI systems deployed on microcontrollers. We implemented two lightweight protocols, MQTT and LoRa, to enable reliable data transfer while maintaining low power consumption.

A. MQTT

MQTT is a publish–subscribe messaging protocol designed for constrained devices and low-bandwidth networks. We used MQTT to connect the microcontroller to mobile and cloud applications, enabling real-time data transfer with minimal overhead. Its lightweight structure reduces both computational and energy requirements, making it suitable for continuous IoT communication.

B. LoRa

LoRa provides long-range, low-power wireless communication, which is particularly valuable for remote or off-grid deployments. By integrating LoRa, the microcontroller can be placed virtually anywhere, without relying on Wi-Fi or cellular networks. The protocol consumes significantly less power compared to traditional wireless communication standards, extending device lifetime while maintaining reliable connectivity over several kilometers.

Together, MQTT and LoRa provide complementary communication options: MQTT for efficient device-to-cloud integration and LoRa for energy-efficient long-range connectivity. These protocols support scalable and sustainable deployment of edge AI systems in diverse environments.

VII. Experiments

We applied S compression on several models, including LeNet and GN. The method achieved strong parameter reduction without noticeable accuracy loss. Table 1 shows that the compression pipeline reduced network storage by 25× to 88× across different models. For example, the size of LeNet decreased from 170 KB to 1.97KB, small enough to fit into on-chip SRAM and eliminate the need for energy-consuming DRAM.

TABLE I
Compression Results using S Method

Network	**Acc. (%)**	**Params**	**Size (KB)**	**Comp.**	**Acc. Ret. (%)**
LeNet	98.13	44,426	170	-	-
LeNet via S	98.08	3,477	13	13×	92.2
LeNet via S	95.25	505	1.97	88×	98.86
GN	99.9	3,719	14.53	-	-
GN via S	99.9	168	0.66	22×	95.48
LightNN	81.09	1,186,986	2319	-	-
LightNN via S	80.17	214,190	418	6×	81.95

VIII. Future work

In future work, we aim to extend this approach to a broader set of algorithms such as RNNs and Transformers to increase its applicability. We will also focus on improving the robustness of the loss function to ensure stable performance across different architectures. Another direction is addressing the issue of local minima to enhance model convergence. In addition, we plan to evaluate the method on larger datasets and real-world deployment scenarios to validate scalability and reliability.

IX. Conclusion

This work introduced a propagated gradient based learnable dynamic compression method, S, and applied it to multiple neural networks. The approach achieved compression rates of up to 88× while maintaining accuracy, reducing model size to a level suitable for deployment on microcontrollers with limited memory. The proposed SNeuron and SLoss formulations enabled subnetworks to emerge during training, producing compact and efficient models without requiring retraining.

In parallel, we addressed the challenge of power consumption in edge AI systems. By integrating a PIR motion sensor and a solar panel, we reduced daily energy use by nearly 90% and achieved energy self-sufficiency. These optimisations make continuous operation feasible in remote or off-grid environments.

The combined contributions of model compression and energy optimisation demonstrate a pathway toward practical, sustainable, and scalable deployment of AI models on microcontrollers.

Acknowledgment

The author would like to thank the International Centre for Theoretical Physics (ICTP), Marconi Lab, and the Multidisciplinary Laboratory for providing the resources and support that enabled this work. Special thanks are extended to the TRIL Fellowship program sponsored by KAUST for offering the opportunity to conduct this research. The guidance and contributions of colleagues and mentors during the fellowship were invaluable in shaping the outcomes of this study.

References

[1] M. H. M. Noor and A. O. Ige, "A survey on state-of-the-art deep learning applications and challenges," 2025. [Online]. Available: https://arxiv.org/abs/2403.17561

[2] A. M. P. G. C. F. F. K. Nicholas D. Lane, Sourav Bhattacharya, "Squeezing deep learning into mobile and embedded devices," 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7994570

[3] J. T. W. J. D. Song Han, Jeff Pool, "Learning both weights and connections for efficient neural networks," 2015. [Online]. Available: https://arxiv.org/abs/1506.02626

[4] W. J. D. Song Han, Huizi Mao, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016. [Online]. Available: https://arxiv.org/abs/1510.00149

[5] M. C. Jonathan Frankle, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1803.03635

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," 1998. [Online]. Available: https://ieeexplore.ieee.org/document/726791

[7] S. R. J. S. Kaiming He, Xiangyu Zhang, "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385

[8] B. C. D. K. W. W. T. W. M. A. H. A. Andrew G. Howard, Menglong Zhu, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[10] M. L. J. S. Xiangyu Zhang, Xinyu Zhou, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," 2017. [Online]. Available: https://arxiv.org/abs/1707.01083