Multiply-And-Max/min Neurons at the Edge: Pruned Autoencoder Implementation

This is the author's version of the article that has been presented at IEEE MWSCAS2023
The editorial version of the paper is available at http://dx.doi.org/10.1109/MWSCAS57524.2023.10405867

Multiply-And-Max/min Neurons at the Edge: Pruned Autoencoder Implementation

Philippe Bich [‡], Luciano Prono [‡], Mauro Mangia [*], Fabio Pareschi [‡†], Riccardo Rovatti [*†] and Gianluca Setti [§†]
[‡]DET, Politecnico di Torino, Italy - Email: {philippe.bich, luciano.prono, fabio.pareschi}@polito.it
[*]DEI, [†]ARCES, University of Bologna, Italy - Email: {mauro.mangia, riccardo.rovatti}@unibo.it
[§]CEMSE, King Abdullah University of Science and Technology (KAUST), Saudi Arabia - Email: gianluca.setti@kaust.edu.sa

***Abstract*—In response to the increasing interest in Internet of Things (IoT) applications, several studies explore ways to reduce the size of Deep Neural Networks (DNNs), to allow implementations on edge devices with strongly constrained resources. To this aim, pruning allows removing redundant interconnections between neurons, thus reducing a DNN memory footprint and computational complexity, while also minimizing the performance loss. Over the last years, many works presenting new pruning techniques and prunable architectures have been proposed but relatively little effort has been devoted to implementing and validating their performance on hardware.**

Recently, we introduced neurons based on the Multiply-And-Max/min (MAM) map-reduce paradigm. When state-of-the-art unstructured pruning techniques are applied, MAM-based neurons have shown better pruning capabilities compared to standard neurons based on the Multiply and Accumulate (MAC) paradigm. In this work, we implement MAM on-device for the first time to demonstrate the feasibility of MAM-based DNNs at the Edge. In particular, as a case study, we implement an autoencoder for electrocardiogram (ECG) signals on a low-end microcontroller unit (MCU), namely the STM32F767ZI based on ARM Cortex-M7. We show that the tail of a pruned MAM-based autoencoder fits on the targeted device while keeping a good reconstruction accuracy (Average Signal to Noise Ratio of 32.6 dB), where a standard MAC-based implementation with the same accuracy would not. Furthermore, the implemented MAM-based layer guarantees a lower energy consumption and inference time compared to the MAC-based layer at the same level of performance.

I. Introduction

Deep Neural Networks (DNNs) have recently attracted unprecedented interest because of their ability to unravel complex problems. The number of tasks these models can solve is constantly growing, but at an increasing cost: very large networks are necessary, running solely on clusters that benefit from almost unlimited resources. Nonetheless, with the rise of Internet of Things (IoT), machine learning on low-power, low-end edge devices has become essential, propelling the investigation of techniques able to reduce the size and computational cost of DNN structures.

The implementation of DNNs on IoT devices unlocks many possibilities in many domains such as Natural Language Processing [1], Precision Agriculture [2], Computer Vision [3] and many others. Moreover, on-device computing avoids the constant dependency on an Internet connection since data is treated locally. This significantly reduces the need to transmit information back and forth to the cloud, reducing latency and energy consumption. Unfortunately, DNNs typically rely on a massive number of parameters, making them impossible to be stored on low-end, low memory footprint devices. In this respect, pruning is of great help since it removes unnecessary and highly redundant interconnections reducing the size of a DNN while also minimizing the performance loss.

In literature, many pruning techniques have been presented [4], [5]. Pruning techniques can be divided in two main categories, namely structured [6]–[9] and unstructured pruning [10]–[12]. The former category focuses on the removal of entire neurons, filters or channels, while the latter considers also the removal of individual interconnections. In particular, the general idea behind unstructured pruning is to give to all interconnections a *score* which is then used as an indicator to select whether an interconnection is to be removed or not. For example, the magnitude of the weights associated to the interconnections is a typically employed scoring method. Finally, pruning can be performed in a one-shot fashion [13], [14] or applied iteratively multiple times [15].

It is also possible to select special architectures that are more resistant to the pruning process compared to standard DNNs. In [16] we introduced neurons based on the Multiply-And-Max/min (MAM) map-reduce paradigm that are naturally prone to pruning. Indeed, neurons based on this paradigm perform better compared to standard Multiply-and-ACcumulate (MAC) neurons when pruned with any unstructured pruning technique already available in the literature. The functionality of MAM neurons have been proven in a large variety of different setups to solve classic computer vision tasks (CIFAR-10, CIFAR-100, ImageNet) [16]. Despite this, no actual on-device implementation of MAM has ever been tested. This is of interest because *i)* MAM inevitably introduces a computational overhead as the existing hardware is not optimized for this map-reduce paradigm (unlike MAC) and *ii)* unstructured pruning – despite typically resulting in good model compression rates – introduces a memory overhead when a sparse representation of the weights matrices is used [17], [18], which may be alleviated by the high pruning capabilities of MAM-based structures.

In this paper, we employ a pruned MAM-based layer at the Edge in order to ensure the feasibility of MAM-based implementation on actual resource-constrained devices. As a typical case study, we chose an autoencoder for electrocardiogram (ECG) signals and we implement its tail on an STM32F767ZI ARM Cortex-M7 based microcontroller unit (MCU) to evalu-

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

This is the author's version of the article that has been presented at IEEE MWSCAS2023
The editorial version of the paper is available at http://dx.doi.org/10.1109/MWSCAS57524.2023.10405867

ate the actual memory footprint, energy consumption and the inference time of MAM neurons. We employ both an efficient sparse representation for the pruned weight matrices and an 8-bit quantized representation of the weights. This is the first time the performance of MAM-based layers is measured in terms of actual inference time, energy consumption and memory footprint (i.e., considering the overhead due to the representation of the sparse weight matrix).

The paper is structured as follows. In Section II, we present a brief summary of the structure of MAM neurons. Then, in Section III, we introduce the ECG autoencoder. In Section IV, we choose an efficient representation for the pruned weight matrices and the memory footprint and the inference time of the MAM layer is measured. Finally, the conclusion is drawn.

II. MAM Layer Description

The following is a brief summary of MAM-based layers. Given the input column vector $\boldsymbol{x} \in \mathbb{R}^N$ and the output $\boldsymbol{z} \in \mathbb{R}^M$ the MAC operation in a standard fully connected (FC) layer can be represented as

$$z_j = \sum_{i=1}^{N} w_{ji} x_i + b_j \quad \text{for } j = 1, \dots, M \tag{1}$$

where w_{ji} is the element at row j and column i of the weight matrix $\boldsymbol{W} \in \mathbb{R}^{M \times N}$ and b_j is the j-th element of the bias vector $\boldsymbol{b} \in \mathbb{R}^M$.

The idea behind a MAM-based neuron is that the summation in (1) may be very roughly approximated by taking into consideration only the maximum and the minimum values, which are typically two of the most significant contributes. This increases pruning capabilities of neurons since they intrinsically rely on less weights. We highlight that this is not the trivial process of keeping only two interconnections per neuron, as the maximum and minimum operations *dynamically* select the interconnections, which are not necessarily the same for different inputs. In this way, (1) becomes

$$y_j = \max_{i \in \{1,\dots,N\}} w_{ji} x_i + \min_{i \in \{1,\dots,N\}} w_{ji} x_i + b_j \quad \text{for } j = 1, \dots, M \tag{2}$$

which describes the behavior of a MAM-based layer. This new type of layer has good pruning capabilities and, in some cases, can be used in place of MAC-based layers with a very low impact on the performance of the DNN.

In order to effectively train MAM layers, the *vanishing contributes* method we described in [16] is employed, consisting in a gradual transition of the layer from MAC to MAM.

III. Case Study: Autoencoder for ECG Signals

In this section we present the performance of a large autoencoder (AE), exploiting a MAM-based layer (AE-MAM), which is pruned in order to fit the network on devices with limited resources.

A. Autoencoder architecture

The autoencoder we consider in this work is composed of three one-dimensional convolutional layers followed by three FC layers and three one-dimensional transposed convolutional layers. This architecture has a latent space dimension equal to 64 while both input and output have a dimension equal to 256. Fig. 1 shows the complete DNN.

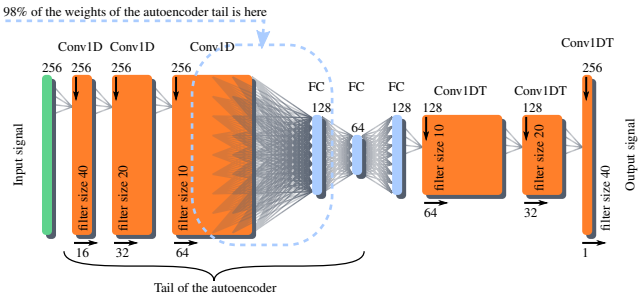

Fig. 1. Structure of the autoencoder. The first FC layer contains more than 98% of the total number of weights in the tail of the autoencoder and it is implemented both as a MAC layer and as a MAM layer. All the layers use a ReLU activation function.

In this work we focus on the tail of the autoencoder (i.e., the part of the model in which the signal is compressed, as highlighted in Fig. 1). The tail of the autoencoder contains more than 2 millions trainable parameters, of which more than 98% are contained in the first FC layer, which is the part of the system that most benefits from pruning. We implement on-device this layer both as a MAC layer and as a MAM layer, and then we prune it to reduce the size of the DNN. We refer to the structure containing the MAC layer as AE-MAC, while we refer to the structure containing the MAM layer as AE-MAM.

B. Dataset

The autoencoder is trained on a synthetic ECG dataset, generated as described in [19] with the same setup of [20], [21]. It is composed of 100 000 ECG windows of size $n = 256$, with 80 000 windows used for training, 10 000 for validation and 10 000 for test. The ECG signal has been scaled in the interval $[0, 1]$. The error on the decoded output of the autoencoder is evaluated with the Reconstruction Signal to Noise Ratio (RSNR), which is a typical metric for ECGs [20], [22], defined as

$$\text{RSNR} = 20 \log_{10} \frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2} = \left(\frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2} \right)_{\text{dB}} \tag{3}$$

where $\boldsymbol{x}$ is the input ECG signal on a window and $\hat{\boldsymbol{x}}$ is the output of the autoencoder, i.e., the reconstructed signal. To evaluate the average performance over a batch of inputs, we use the Average RSNR (ARSNR), defined as the average value of RSNR.

C. Quantization and training

When dealing with the implementation of DNNs at the Edge, quantization of the parameters is fundamental in order to minimize the memory footprint of the structure. In order to minimize the effect of parameters quantization on the accuracy of the network, we perform quantization-aware training. In particular, we employ fake-quantization [23], i.e., we quantize parameters and activations during the forward pass phase of training in order to take into account the quantization error.

We train both the AE-MAC and the AE-MAM for 200 training epochs with full-precision parameters (i.e., 32-bit floating point). Then we fine tune the models with additional quantization-aware training using uniform quantization for another 10 epochs. Weights are quantized to 8 bits in the range

This is the author's version of the article that has been presented at IEEE MWSCAS2023
The editorial version of the paper is available at http://dx.doi.org/10.1109/MWSCAS57524.2023.10405867

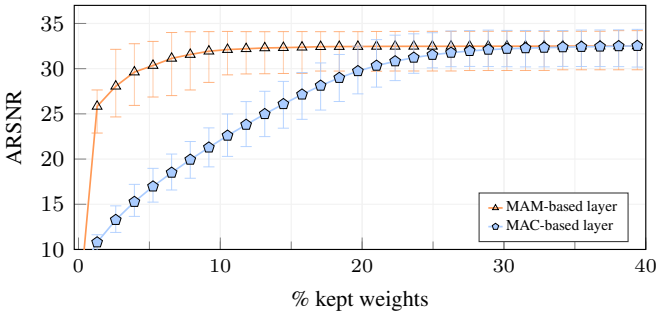

Fig. 2. ARSNR vs the percentage of remaining weights in the first FC layer of the autoencoder. The mean performance over 10 trainings is shown, with the maximum and minimum ARSNR indicated by the error bar.

$[-1, +1]$ and inputs and outputs are quantized to 16 bits in the range $[-1, +1]$. Biases are kept as-they-are and converted a-posteriori to 32-bit fixed point to maximize accuracy, as their influence on the memory footprint is almost negligible. The Mean Squared Error (MSE) is selected as loss function and Adam Optimizer [24] is used with a starting learning rate of 0.001 and a batch size of 128.

D. Pruning the autoencoder

In this work we focus on one-shot unstructured pruning, i.e. the model is trained and then a set of selected weights is removed without taking into account the spatial disposition of the remaining interconnections. We prune the first FC layer both in AE-MAC and AE-MAM, as its impact on the memory footprint is dominant compared to the other layers. We select the Magnitude Pruning technique (MP) as pruning algorithm, consisting in scoring weights based on their absolute magnitude. The idea behind this approach is based on the assumption that the smaller the magnitude of a weight, the less its influence on the output of the network. Fig. 2 summarizes the resulting ARSNR vs the number of non-zeroed weights, where the mean performance over 10 trained autoencoders is shown.

Results show that AE-MAM is more robust to pruning compared to AE-MAC. With 5% of remaining weights in the first FC layer, the mean ARSNR over 10 trained models of AE-MAM is still over 30 dB while it is only about 17 dB in the case of AE-MAC. All the pruned weights are put to zero and since we are using an unstructured pruning technique to remove interconnections, the result is a sparse weight matrix.

IV. On-device performance of a pruned MAM-based layer

In this section, we evaluate the actual performance of the MAM-based layer implemented on an edge device. Firstly, a representation for the sparse matrix (used for both the MAC and the MAM layer) is to be defined, then the memory footprint, the inference time and the energy consumption of the layer are computed and finally the tails of the AE-MAC and AE-MAM autoencoders are implemented on an ARM Cortex-M7 based MCU, namely STM32F767ZI, and their performance is compared.

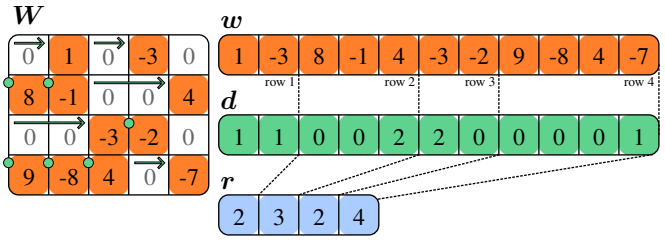

Fig. 3. Example of representation for the description of sparse weight matrices.

A. Sparse Matrix Representation

It is important to notice that the sparsity represented in Fig. 2 does not directly translate into an advantage in terms of memory footprint since, in order to be exploited, it is necessary to employ a sparse representation of the matrix which inevitably introduces a computational and memory overhead. Accordingly, in order to benefit from the sparsity of the pruned weight matrix, we use the Compressed Sparse Row (CSR) approach [25]. In particular, being $\boldsymbol{W} \in \mathbb{R}^{M \times N}$ the sparse weight matrix, we encode it with three vectors:

- $\boldsymbol{w}$ is the collection of $w_{ij} \in \boldsymbol{W} | w_{ij} \neq 0$, arranged row after row;
- $\boldsymbol{d}$ indicates for each $w_{ij} \neq 0$ the number of elements $w_{ij} = 0$ separating it from the previous element $w_{ij} \neq 0$ or from the beginning of the row;
- $\boldsymbol{r}$ contains the number of weights $w_{ij} \neq 0$ in each row of matrix $\boldsymbol{W}$.

Note that we introduced a slight modification to standard CSR – as $\boldsymbol{d}$ encodes offsets, and not directly the indexes. Fig. 3 shows an example of representation of a sparse matrix where the remaining weights of matrix $\boldsymbol{W}$ are highlighted in orange.

In order to minimize the memory footprint, both the values in $\boldsymbol{w}$ and $\boldsymbol{d}$ are encoded with 8 bits. This means that the maximum distance between two non-pruned weights is 255, i.e., the maximum value that can be represented with $\boldsymbol{d}$. If the distance between two non pruned weights is bigger, one or more dummy-elements with weight value zero are introduced in the list in convenient positions. On the contrary, vector $\boldsymbol{r}$ contains 16-bit integer values. This is typically not a problem since the overhead caused by this vector is normally negligible compared to the size of $\boldsymbol{w}$ and $\boldsymbol{d}$.

Fig. 4 shows the actual memory footprint of the MAC-based and MAM-based FC layer vs the ARSNR. Compared to the curves in Fig. 3, memory footprint is about twice as expected due to the sparse representation. Being the overhead proportional to the number of weights kept, the advantage of MAM over MAC is accentuated. The maximum available memory on the device for the first FC layer is also shown and it is computed taking into account the size of the weights of the other layers that are implemented and the 32.8 kB of the SRAM that must remain free for the output buffer with the highest size (i.e., the input of the first FC layer). We highlight that the size of the original non-pruned autoencoders (both AE-MAC and AE-MAM) is almost 2.2 MB (with 8-bit weights and 32-bit biases) which is more than the size of the SRAM of the employed device.

With standard MAC neurons, considering the memory overhead caused by the sparse matrix representation, the benefit

This is the author's version of the article that has been presented at IEEE MWSCAS2023
The editorial version of the paper is available at http://dx.doi.org/10.1109/MWSCAS57524.2023.10405867

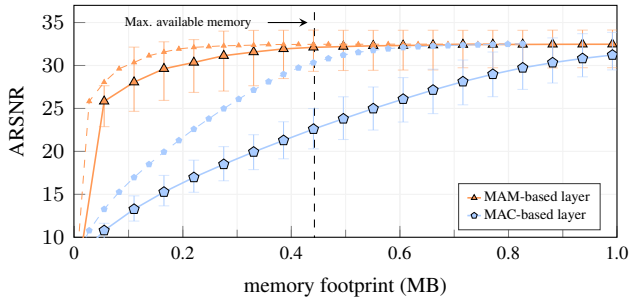

Fig. 4. ARSNR vs the actual memory footprint of the first FC layer for the sparse representation of the weight matrix. Also, the curves showing the memory footprint with no sparse representation overhead are highlighted (i.e., the dashed lines take into account only the size of vector $\boldsymbol{w}$). This shows that the overhead is worse for MAC, being it proportional to the number of kept weights.

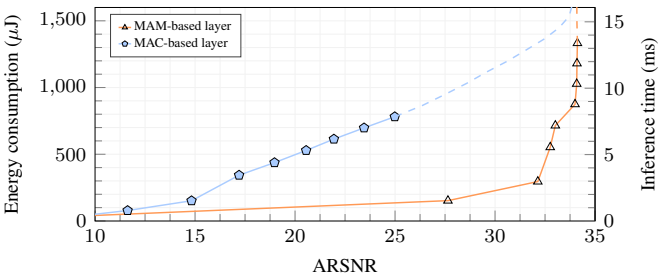

Fig. 5. ARSNR, inference time and energy consumption of the MAC and the MAM layers implemented on-device containing an increasing number of weights. For large layers who could not fit on device, an estimate is made and it is shown with a dashed line.

introduced by unstructured pruning is very limited. Conversely, MAM neurons offer a higher pruning rate that makes unstructured methods a viable option.

B. On-device implementation

The tail of the autoencoders is implemented on an ARM Cortex-M7 based MCU, namely STM32F767ZI. The device is characterized by an SRAM of 512 kB and a maximum operative frequency of 216 MHz. Custom C code is used to implement the different layers of the network, compiled with gcc with optimization options *-Ofast* and *-loop-unroll*. AXI interface, instruction cache and memory cache are enabled. The first FC layer, originally containing most of the weights, is pruned as in Section III-D and encoded as in Section IV-A. We take the best AE-MAC and AE-MAM model in terms of accuracy obtained over the 10 trainings and we implement them with the MAC and the MAM layers containing the number of remaining weights shown in Figure 4, where the available memory allowed it.

For all these models, we compute the accuracy as well as the execution time and the energy consumption of the first FC layer. Energy consumption is estimated as power multiplied by execution time, where power is evaluated from typical electrical characteristics of STM32F767ZI indicated on datasheet, with a clock frequency of 180 MHz and a supply voltage of 1.7 V, resulting in a typical power value of 99.5 mW.

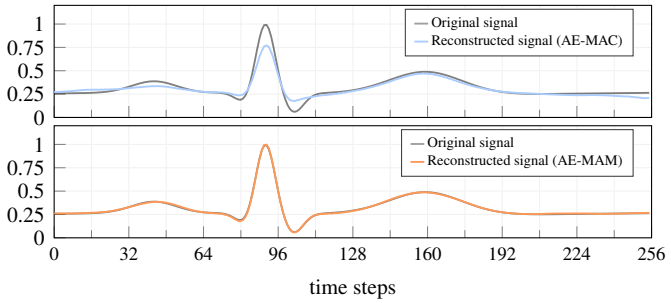

Fig. 6. Example of the original ECG signal and of the reconstructed signal at the output of AE-MAC (top) and AE-MAM (bottom).

In Figure 5, we show the energy consumption and the inference time of the MAC and MAM layers implemented on device. For models with a large number of weights that could not fit on the device, we make an estimate, shown in the plot with a dashed line, assuming that the inference time increases linearly with the number of weights – as this is the trend observed with measured data. This estimate does not take into account that an increase in the maximum available memory of a computational system might also result in a higher energy consumption.

It must be noted that the implementation of MAM-based neurons introduces a computational overhead, mainly due to a branch structure (i.e., an "if" structure) required for the evaluation of both the maximum and the minimum values in a neuron while MAC-based layers sequentially accumulate all the values. However, since MAM-based layers can be aggressively pruned and still achieve good results with few parameters, the inference time and the energy consumption of MAM layers are lower than those of a MAC layer at the same level of performance.

As an example, the accuracy of the reconstructed ECG sample by AE-MAM is of 32.6 dB with the MAM layer having an inference time of 2.97 ms and consuming 295 µJ while the accuracy of the largest AE-MAC model that can be implemented on the device is only 17.6 dB with an inference time for the MAC layer of 7.85 ms and an energy consumption of 781 µJ. Fig. 6 visually illustrates the quality of reconstruction of an ECG sample by these two models.

V. Conclusion

In this work we have analyzed the performance of Multiply-And-Max/min neurons on hardware through the implementation of a DNN-based autoencoder for ECG signals whose largest layer has been substituted with a MAM-based layer. The tail of the DNN has been implemented on an ARM Cortex-M7 based MCU. The network, trained with a quantization-aware approach, has been quantized and the MAM-based layer has been pruned. The pruned structure has been encoded by means of a sparse matrix representation and fit on the MCU device, where the actual memory footprint, inference time and energy consumption of the pruned layer have been evaluated. The MAM-based pruned layer guarantees a far better ECG reconstruction quality compared to the one with the MAC layer, while also having lower energy consumption and inference time.

REFERENCES

[1] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network," in *32nd International Conference on Neural Information Processing Systems (NIPS'18)*, Dec. 2018, pp. 9031–9042.

[2] D. Aghi, S. Cerrato, V. Mazzia, and M. Chiaberge, "Deep Semantic Segmentation at the Edge for Autonomous Navigation in Vineyard Rows," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 3421–3428. doi:10.1109/IROS51168.2021.9635969

[3] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The Design and Implementation of a Wireless Video Surveillance System," in *21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*, Sep. 2015, pp. 426–438. doi:10.1145/2789168.2790123

[4] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the State of Neural Network Pruning?" *Proceedings of Machine Learning and Systems*, vol. 2, pp. 129–146, Mar. 2020.

[5] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the Value of Network Pruning," in *7th International Conference on Learning Representations (ICLR 2019)*, Sep. 2018.

[6] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 1398–1406. doi:10.1109/ICCV.2017.155

[7] Z. Chen, T.-B. Xu, C. Du, C.-L. Liu, and H. He, "Dynamical Channel Pruning by Conditional Accuracy Change for Deep Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 799–813, Feb. 2021. doi:10.1109/TNNLS.2020.2979517

[8] Y. He, P. Liu, L. Zhu, and Y. Yang, "Filter Pruning by Switching to Neighboring CNNs With Good Attributes," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022. doi:10.1109/TNNLS.2022.3149332

[9] K. Persand, A. Anderson, and D. Gregg, "Taxonomy of Saliency Metrics for Channel Pruning," *IEEE Access*, vol. 9, pp. 120 110–120 126, 2021. doi:10.1109/ACCESS.2021.3108545

[10] C. Wang, G. Zhang, and R. Grosse, "Picking Winning Tickets Before Training by Preserving Gradient Flow," in *11th International Conference on Learning Representations (ICLR)*, Feb. 2022.

[11] X. Dai, H. Yin, and N. K. Jha, "NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1487–1497, Oct. 2019. doi:10.1109/TC.2019.2914438

[12] C. Louizos, M. Welling, and D. P. Kingma, "Learning Sparse Neural Networks through L0 Regularization," in *6th International Conference on Learning Representations (ICLR 2018)*, Feb. 2018.

[13] S. Zhang and B. C. Stadie, "One-Shot Pruning of Recurrent Neural Networks by Jacobian Spectrum Evaluation," in *International Conference on Learning Representations (ICLR 2020)*, Mar. 2020.

[14] T. Chen *et al.*, "Only Train Once: A One-Shot Neural Network Training And Pruning Framework," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 19 637–19 651.

[15] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," in *7th International Conference on Learning Representations (ICLR 2019)*, Sep. 2018.

[16] L. Prono, P. Bich, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "A Multiply-And-Max/min Neuron Paradigm for Aggressively Prunable Deep Neural Networks," Apr. 2023, preprint available on TechRxiv.

[17] K. Wu, Y. Guo, and C. Zhang, "Compressing Deep Neural Networks With Sparse Matrix Factorization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 3828–3838, Oct. 2020. doi:10.1109/TNNLS.2019.2946636

[18] X. He and T. Liu, "Sparse Matrix Reconstruction Based on Sequential Sparse Recovery for Multiple Measurement Vectors," in *2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*, Apr. 2021, pp. 480–483. doi:10.1109/ICCCS52626.2021.9449312

[19] P. E. McSharry, G. D. Clifford, L. Tarassenko, and L. A. Smith, "A dynamical model for generating synthetic electrocardiogram signals," *IEEE Trans. on Biom. Eng.*, vol. 50, no. 3, pp. 289–294, Mar. 2003. doi:10.1109/TBME.2003.808805

[20] M. Mangia, L. Prono, A. Marchioni, F. Pareschi, R. Rovatti, and G. Setti, "Deep Neural Oracles for Short-window Optimized Compressed Sensing of Biosignals," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 3, pp. 545–557, Jun. 2020. doi:10.1109/TBCAS.2020.2982824

[21] L. Prono, M. Mangia, A. Marchioni, F. Pareschi, R. Rovatti, and G. Setti, "Deep Neural Oracle With Support Identification in the Compressed Domain," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 4, pp. 458–468, Dec. 2020. doi:10.1109/JETCAS.2020.3039731

[22] D. Craven, B. McGinley, L. Kilmartin, M. Glavin, and E. Jones, "Compressed sensing for bioelectric signals: A review," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 2, pp. 529–540, Mar. 2015. doi:10.1109/JBHI.2014.2327194

[23] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8BERT: Quantized 8Bit BERT," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, Dec. 2019, pp. 36–39. doi:10.1109/EMC2-NIPS53020.2019.00016

[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017. doi:10.48550/arXiv.1412.6980

[25] N. Bell and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, Nov. 2009, pp. 1–11. doi:10.1145/1654059.1654078 ISSN: 2167-4337.