

Encoding of Neural and Deep Learning Networks using generalized bit-planes: fostering sparsity while reducing storage demands and computational complexity

Gabryel Medeiros¹, Lisandro Lovisolo^{1,2}, Helio N. Cunha Neto²,
Leonardo Rey Vega³, Guilherme Lucio Abelha Mota⁴

Abstract—Deep neural networks (DNN) achieve high performance across many tasks. They often require extensive parameter sets and computationally intensive weight representations. The memory footprint, storage demand, and arithmetic complexity directly affect the practical deployment of DNN models, especially in resource-constrained settings. We investigate post-training quantization of DNN models based on sums of powers of two (SOPOT), resulting in model compression. This is done using the Matching Pursuits with Generalized Bit Planes (MPGBP) algorithm. The proposed approach replaces the conventional weight formats with sparse, signed power-of-two (SPTs) representations that are better suited to low-complexity implementation. We compare three approximation granularities: neuron-, layer-, and network-wise. Their trade-offs in terms of accuracy, sparsity, and entropy-based rate on VGG-16 pretrained on ImageNet-1K are evaluated. The results show that the network-wise strategy provides the best compression-performance trade-off.

I. INTRODUCTION

DNN have achieved remarkable results in several application domains, including computer vision, speech processing, and natural language understanding [1]. However, high predictive performance often comes at the cost of large models, many parameters, and high computational demand [2]. Large neural networks require more memory, storage, bandwidth, and arithmetic operations. These requirements make training, deployment, and inference more difficult, especially in environments with limited resources. For this reason, efficiency has become a central issue in modern deep learning. Model compression, parameter reduction, precision scaling, and low-complexity representations have therefore attracted increasing attention as ways to reduce implementation cost while preserving predictive performance [3].

Different strategies have been proposed to reduce the storage and computational burden of deep neural networks. Common approaches include pruning, quantization, low-rank approximation, and architectural simplification [2]. These approaches aim to reduce the number of parameters, the number of arithmetic operations, and the amount of data required to store and process a model [4]. As a result, these approaches can improve efficiency during training, inference, and deployment. In the case of VGG-based models, compression has also been explored through architectural modifications designed to reduce model size and execution cost while preserving recognition performance [5].

In this work, we use sums of powers of two (SOPOT) as a structured representation for neural network coefficients. These are obtained through the Matching Pursuits with Generalized bit-planes (MPGBP) algorithm [6]. MPGBP is a matching-pursuit-based algorithm that iteratively approximates coefficients using a sparse combination of signed power-of-two terms, yielding compact representations suitable for low-complexity implementation. The main idea is to replace conventional coefficient formats with a more compact structure that is better suited to low-complexity arithmetic. At the same time, we seek to preserve the predictive capacity of the original model. Rather than only evaluating compression in terms of reduced precision or parameter count, this paper examines how the distribution of signed power-of-two terms across the network affects the balance between accuracy and representation cost. The analysis explores coefficient approximation at multiple granularities by applying the MPGBP to individual neurons, to individual network layers, and to the full network. This enables a systematic comparison of how different allocations of terms and thus computation effort over the model affect its performance and compression efficiency.

The remainder of this paper is organized as follows. Section II provides an overview of NN compression-related works. Then, Section III discusses the approximation of NN models' parameters using SOPOTs, which we investigate in this work. The methodology employed to evaluate the results is provided in Section IV, while the results follow in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

Reducing the size and computational cost of DNNs has led to works on model compression [1]. Prior work demonstrated that a considerable proportion of network parameters can be eliminated with minimal effect on accuracy [7]–[9]. Specifically, [8] proposes pruning-based compression for DNN, retaining only the most important connections, substantially reducing storage and computation while preserving predictive performance [8]. This approach was further developed in Deep Compression, which integrates pruning, trained quantization, and entropy coding into a comprehensive framework for deep neural network compression [4].

Another approach involves making neural networks more efficient by changing larger parts of the model, such as filters, channels, or entire layers, instead of just removing individual weights. A linear structure accelerates inference while preserving accuracy in convolutional layers in [10]. Afterwards, structured pruning methods such as filter pruning and ThiNet [11] showed that removing complete filters can reduce both model size and floating-point operations more

¹ – Laboratório de Processamento de Sinais, Aplicações Inteligentes e Comunicações (PROSAICO), FEN, UERJ. ² – Programa de Pós-graduação em Engenharia Eletrônica (PEL), FEN - UERJ. ³ – Universidad de Buenos Aires and CSC-CONICET, Bs.As., Argentina. ⁴ – Programa de Pós-graduação em Ciências Computacionais e Modelagem Matemática, IME, UERJ.
e-mails: mgabryeloliveira@gmail.com, lovisolo@eng.uerj.br, helio.cunha@ime.uerj.br, lrey@fi.uba.ar and guimota@ime.uerj.br.
This work is funded by FAPERJ Call N° 15/2024, CAPES and CNPq.

effectively than irregular sparsity. These works emphasize that parameter reduction alone is insufficient; the form of the compressed representation also determines whether practical computational gains can be achieved.

Transitioning to representations based on powers of two are particularly attractive because they combine coefficient compression with arithmetic simplification [6], [12]. Recent studies have explored similar directions from different perspectives. Geng et al. introduced an efficient non-uniform quantization strategy for deep neural networks [13]. On the other hand, Przewlocka-Rus et al. focused on low-bit-width quantization schemes that are directly compatible with efficient hardware implementations [14]. These works show that power-of-two-based representations are not only compact but also well aligned with low-complexity computation. Quantization methods can be divided into post-training quantization (PTQ) and quantization-aware training (QAT) [15]. In PTQ, a pre-trained model is quantized after training, whereas QAT incorporates quantization effects during training.

This work is related to PTQ, but differs from previous works in two important aspects. First, the paper adopts a structured approximation obtained through MPGBP, rather than a conventional quantizer design. Second, the paper explicitly analyzes network parameters' approximation at different granularities, namely neuron-wise, layer-wise, and network-wise, to study how the allocation of signed power-of-two terms affects the trade-off among accuracy, sparsity, and representation cost.

III. THE SOPOT REPRESENTATION VIA MPGBP

SOPOTs representation of numbers allows multiplierless implementations [12], [16], favoring complexity reduction. Each multiplication requires as many registers as SPTs/active "signed-bits" in the number expression. Such that limiting the number of active "signed-bits" does not require using generic multipliers and restricts the implementation complexity. The number of active "signed-bits" defines the computational complexity of the multiplication operation since it bounds the number of shift operations required for the multiplication process. That is, the greater the number of active bits, the greater the computational burden due to multiplication.

Among the techniques for obtaining SOPOT approximations, we employ the MPGBP [6] since it is fast and low-complexity. It approximates vectors with guaranteed convergence/error bound. This allows imposing both the word length and, if desired, a maximum number of SPTs for the whole vector. The MPGBP [6] rapidly obtains SOPOT representations of vectors with guaranteed convergence given by

$$\mathbf{w} \approx \sum_{i=1}^I \beta^{k_i} \mathbf{c}_{j_i}, \quad (1)$$

where β defines the bit-plane strengths in the approximation, $\mathbf{c}_j$ is the *codeword* of a *codebook* $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_j, \dots, \mathbf{c}_{\#\mathcal{C}}\}$ and $\#\mathcal{C}$ is the cardinality of *codebook*. Besides, j_i is the index for the *codeword* selected in the i -th iteration of decomposition. The *codewords* are composed of coordinates/coefficients with values, i.e., -1, 0, or 1, so $\beta^{k_i} \mathbf{c}_{j_i}$ has coordinates equal to $-\beta^{k_i}$, 0, or β^{k_i} .

SOPOT-based PTQ for an NN using MPGBP was investigated in [16], intending multiplierless implementations. They approximated each neuron of a multilayer perceptron for multiple-input multiple-output detection and indicated that the number of SPTs impacts the trade-off between computational cost and performance. In contrast, the present study examines MPGBP-based SOPOT approximation applied to different network granularities of a deep vision model and their influence on accuracy, sparsity, and representation cost. Besides, in contrast, we apply MPGBP-based SOPOT to a convolutional network.

A. SOPOTs via the MPGBP

MPGBP approximates a vector $\mathbf{w}$ via ($\beta = 2$ in eq. (1))

$$\mathbf{w} \approx \sum_{i=1}^I 2^{k_i} \mathbf{c}_{j_i}. \quad (2)$$

Good codebook designs and fast decomposition via MPGBP for $\beta = 2$ exist. For $\mathbf{w} \in \mathbb{R}^N$, the most compact representations are obtained if the number of non-zero coordinates P per codeword ($\mathbf{c}_j$) is, following [6], $P = \lfloor \sqrt{N} \rfloor$.

Note that the MPGBP obtains ternary representations in bit-planes. The MPGBP jointly determines the *codeword* (coordinates) and the weight 2^{k_i} in the K -th bit-plane. We apply them to encode the NN weights.

B. NN parameters approximation (SOPOT representation) using MPGBP

The MPGBP algorithm operates on a single input vector $\mathbf{w} \in \mathbb{R}^N$, producing a SOPOT approximation subject to a budget of $\text{SPT}_{\max}$ total number of SPTs. That is, the MPGBP is allowed to run until a given number of non-null weights $\text{SPT}_{\max}$ is reached, bounding the implementation complexity. To apply it to a neural network with multiple weight tensors, we must define *what constitutes the input vector*

We investigate three granularities levels, each defining a different scope for the MPGBP approximation.

1) Neuron Wise Approach (R): Each neuron's incoming weight vector is approximated independently. For a fully connected layer with weight matrix $\mathbf{W} \in \mathbb{R}^{M_{l-1} \times M_l}$, where M_l is the number of neurons in the l -layer (the layer zero corresponds to the network inputs), each row $\mathbf{w}_l \in \mathbb{R}^{M_{l-1}}$ is approximated using the MPGBP. For convolutional layers, each filter is flattened into a single vector. The number of nonzero codeword entries is $P = \lfloor \sqrt{M_{l-1}} \rfloor$. The number of SPT-budget per neuron is $\text{SPT}_{\max} = \lceil m M_{l-1} \rceil$, where M_{l-1} is the number of incoming weights for the neuron and m is a multiplier used to evaluate different tradeoffs. Note that for convolutional layers, P and $\text{SPT}_{\max}$ are derived from the total number of parameters in the convolution filter F_l producing the output instead of the connections from the previous layer.

MPGBP frameworks: The proof for MPGBP convergence rests on being able to compute the norm for $\|\mathbf{w}\|$ [6]. If we join the weights of all the neurons in a layer, we get the matrix

$$\mathbf{W}_l = [\mathbf{w}_{1,l}^T \quad \mathbf{w}_{2,l}^T \quad \dots \quad \mathbf{w}_{M_l,l}^T]^T. \quad (3)$$

Using the Frobenius norm of $\mathbf{W}_l$, the same convergence proof for the MPGBP stems when approximating $\mathbf{W}_l$, and the optimal number of non-null terms per MPGBP iteration is given by $P = \lfloor \sqrt{M_{l-1}M_l} \rfloor$ where $M_{l-1}M_l$ is the number of elements in $\mathbf{W}_l$, considering that there can be weights connecting all the outputs from the previous layer ($l-1$) neurons M_{l-1} to all the neurons M_l in the l -th layer. In the ‘layer-wise’ approach, we apply the MPGBP to approximate the matrix (or vector) of the input parameters of all the neurons in each network layer using SPTs. Similarly, we can approximate the network tensor using a network-wise approach.

2) *Layer-Wise Approach (L)*: All the parameters in a layer are compressed using MPGBP altogether. The weight matrix $\mathbf{W}_l \in \mathbb{R}^{M_{l-1} \times M_l}$ is approximated using MPGBP. The number of nonzero codeword entries is $P = \lfloor \sqrt{M_{l-1}M_l} \rfloor$ and the SPT budget is $\text{SPT}_{\max} = \lceil m M_{l-1} M_l \rceil$. We note that, for convolutional layers, P and $\text{SPT}_{\max}$ are set considering the total number of parameters in the convolution filters producing the layer outputs.

whose weights are already small receives the same proportional budget as a layer with large weights. This motivates the network-wise approach.

3) *Network-Wise Approach (N)*: At last, the network approach extends the two previous approaches. We construct a tensor $\mathbf{W}$ from the network parameters. Since a norm can also be defined in this case, the convergence for approximating the tensor using MPGBP also follows. In this case, the optimal number of non-null terms per MPGBP iteration is given by

$$P = \left\lfloor \sqrt{\sum_{l=1}^L M_{l-1}M_l} \right\rfloor, \quad (4)$$

where M_{l-1} is the number of inputs to the network. Note that the total number of network parameters is given by $W = \sum_{l=1}^L M_{l-1}M_l$. And, the SPT budget is $\text{SPT}_{\max} = \lceil mW \rceil$. Note that the convolution filters are considered as in the approaches before; that is, if the network has W parameters, considering all filters, they are organized in a tensor and the MPGBP is applied to this tensor to obtain the SOPOT approximation, leading to $P = \lfloor \sqrt{W} \rfloor$.

This mode provides *global* budget allocation. At each iteration, the P weights with the largest residual magnitude are selected from the entire network. If one layer has already converged to a low residual, while another still has a large approximation error, the algorithm automatically redirects SPTs to where they are most needed.

IV. EVALUATION METHODOLOGY

Once the different approximations of an NN have been obtained, we assess their performance using different criteria.

A. Sparsity

We define sparsity as the fraction of the model parameters set to exactly zero after compression (i.e., after expressing it using approximations). The sparsity is given by $S = 1 - \frac{W_{\text{nonzero}}}{W}$ where W is the total number of network parameters and W_{nonzero} is the number of parameters that are not zeroed. Unlike pruning methods that explicitly target sparsity, MPGBP

induces sparsity as a natural consequence of its SPT greedy allocation: the algorithm concentrates representation budget on the largest-magnitude residual components, leaving small weights unrepresented.

B. Rate

We measure the information-theoretic rate of the compressed representation through binary plane decompositions. Each compressed weight/parameter is decomposed into a hierarchy of binary planes. Let k be the bit-plane index; each weight, independently of applying the vector-, layer-, or neuron-wise strategy, is approximated using SOPOTs. We arrange the approximated weights using a sequence of matrices. Firstly, we define a parameter mask; for each network weight/parameter, it is zero or one. When zero, it means that the corresponding weight is zero and does not take part in the compressed/approximated network. Thus, we code only the non-zero parameters using two matrices.

Let $\mathbf{b}$ be the parameter-mask. As noted, $b_i = 1$ indicates whether the bit-plane weight at position i is nonzero. This mask determines which parameters exist in the compressed network. The exponent existence matrix is evaluated at each bit-plane for the active parameters. It encodes if the k -th bit-plane¹ is used to approximate the i' -th weight ($e_k^{(i')}$, for $k = 0, 1, \dots, K_{\max}$: $e_{i'}^{(k)} = 1$ if weight i' is present in the k -th bit-plane. It exists only where $b_i = 1$. The signs at the k -th bit-plane are arranged in $\mathbf{s}^{(k)}$, for $k = 0, 1, \dots, K_{\max}$. $s_{i''}^{(k)} = 1$ if the 2^{-k} term of weight i'' is positive. Note that it exists only if $e_{i'}^{(k)} = 1$.

This structure aims at exploring sparsity. It explores zeroed parameters, and also the SOPOT representation sparsity. The SPT existence bit exists only for a non-zero parameter if the bit-plane takes part in the approximation, and the sign bits, if the existence bit is not zero in the bit-plane. As a result, the necessary rate to encode the network is (in bits/parameter)

$$R = \frac{1}{W} \left[WH(\mathbf{b}) + \sum_{k=0}^K \left(W_{\text{nz}} H(\mathbf{e}_k | \mathbf{b}) + W_{\text{exist},k} H(\mathbf{s}_k | \mathbf{e}_k) \right) \right] \quad (5)$$

where $H(\mathbf{b})$ is the binary entropy of the parameter-mask, $H(\mathbf{e}_k | \mathbf{b})$ is the conditional entropy of the existence plane for exponent k computed only considering the nonzero weights, W_{nz} is the number of nonzero weights, and $H(\mathbf{s}_k | \mathbf{e}_k)$ is the conditional entropy of the weight sign computed considering only the weights whose approximations have a non-zero weight for the exponent k . This rate represents the theoretical minimum number of bits per parameter achievable by a lossless entropy coder applied to the plane decomposition.

C. Budget Equivalence

For trustworthy and meaningful comparisons, all three approaches use the same total SPT budget (or at least a similar one). In the Neuron approach, each neuron is approximated using $\lceil m M_{l-1} \rceil$ SPTs, summing for all neurons to approximately $m \cdot W$ SPTs in total. One uses $\lceil m M_{l-1} M_l \rceil$ SPTs for layer l , in the Layer approach, summing up to the same total. That is, all three approaches allocate approximately $\lceil mW \rceil$

¹Note that we use i' instead of i because the indices are different from above, since we operate only on the non-zero weights now.

SPTs among the network elements. Therefore, differences in the accuracy among the approaches reflect impacts of the *allocation strategy* – how the SPT budget is distributed across parameters – and not on complexity, measured by the number of SOPOTs arithmetic operations.

Although above we discuss fully connected networks, the same reasoning applies to DL having convolutional layers, since the SPT budget is set as a function of the number of parameters composing the entities (neuron/filter, layer, network).

V. RESULTS

We evaluate MPGBP compression on VGG-16 [17], a convolutional neural network with 138.4M parameters, pretrained on ImageNet-1K [18]. The pretrained weights are obtained from the TorchVision library [19] with *IMAGENET1K_V1* weights. All results reflect post-training quantization (PTQ), as no retraining is performed after compression.

For each of the three granularities (Neuron, Layer, and Net), we apply multipliers $m \in \{0.25, 0.50, 0.75, 1.00\}$ to control the total SPT budget SPT_{max} . In all modes, the number of nonzero codeword entries P also follows the rationale in Section III-B. The resulting models are used to classify 2000 randomly selected images from ImageNet, and we report average accuracies on 5-folds. The results are also compared against the baseline accuracy of the uncompressed VGG-16 on our evaluation subset, which is 72.18%.

A. Accuracy vs. Sparsity

Fig. 1 relates sparsity to accuracy for each approach. At $m = 0.50$, the Network-wise approach (N) zeroes 53.3% of the parameters with no accuracy loss. The Layer-wise approach achieves similar overall sparsity (51.2%) but suffers 10.65% accuracy degradation. This indicates that the distribution of parameters equal to zero across layers determines the impact on accuracy. In the Network approach, the MPGBP greediness zeroes parameters that contribute less to the network's output. Layers with smaller weights are more strongly pruned, while most of the SPT terms are used for layers with larger weights. In contrast, the Layer approach allocates SPTs in each layer independently of other layers; this potentially leads to zeroing critical weights in possibly sensitive layers while wasting SPTs in less important layers. That is, the Network approach leads to a weight-based sparsity across the network, while the Layer approach forces the same sparsity at all layers. At high sparsity levels ($S > 0.70$), only the Network approach leads to accuracy close to the baseline, whereas the Neuron and Layer approaches collapse to approximately 0.10 and 0.00, respectively. This demonstrates that global budget allocation is essential for aggressive compression, where the margin for misallocation is small.

B. Accuracy vs. SPT Budget

Fig. 2 presents the trade-off between SPT budget and accuracy for the three granularities. The network approach is consistently better at every multiplier level, reaching baseline accuracy (72.18%) at $m = 0.50$ with only 0.50 SPTs per parameter. Since 53.3% of weights are zeroed, the remaining nonzero weights use approximately one SPT each on average.

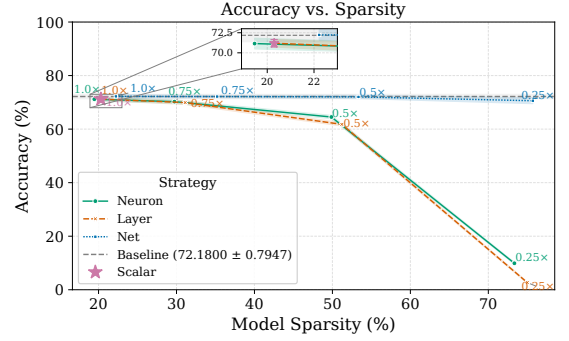


Fig. 1: Accuracy vs. Model Sparsity.

The Layer and Neuron approaches require higher budgets to reach comparable accuracy. At $m = 0.25$, the Network-wise approach compressed model hits 72.25% of the instances, while the Layer- and Neuron-approach models only hit 2.45% and 9.45%, respectively. That is, at low SPT budgets, the allocation strategy is critical. While the Network approach distributes SPTs among the globally most significant weights, the other Layer and Neuron approaches distribute SPTs uniformly across layers or neurons, regardless of their relative importance. At higher budgets ($m = 1.00$), the performance gap narrows, meaning that there are SPTs to represent the important weights with sufficient precision. The accuracy for the Neuron approach is 72.23%, for the Layer approach, it reaches 71.19%, while the Network approach maintains 72.55%, all within 1.2 percentage points w.r.t the baseline.

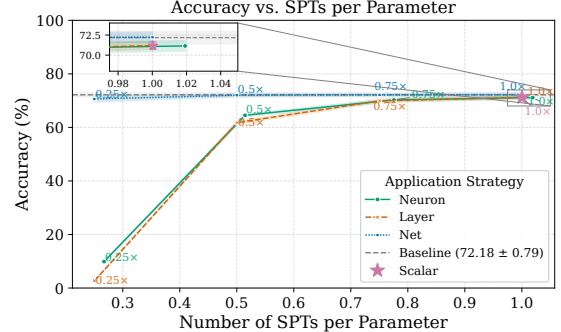


Fig. 2: Accuracy vs. Number of SPTs per Parameter.

C. Entropy vs. Accuracy

Fig. 3 shows accuracy as a function of the entropy-based rate in bits per parameter, computed via the conditional binary plane decomposition as in equation (5). This plot captures the information-theoretic cost of each compressed representation, independent of any specific coding scheme. The Network approach achieves the most favorable rate-accuracy trade-off. At 2.63 bits/parameter, it matches the baseline accuracy, corresponding to a compression ratio of 12.2× relative to the original 32-bit representation. The Layer approach requires 4.35 bits/parameter to reach 71.19% accuracy (7.4× compression), and the Neuron approach needs 4.42 bits/parameter for 71.15% (7.2× compression). The advantage of the Network approach arises from its induced sparsity pattern.

D. Per-Layer Sparsity Distribution

Fig. 4 reveals the mechanism behind the Network-wise approach's superior accuracy. At $m = 0.50$, all three modes pro-

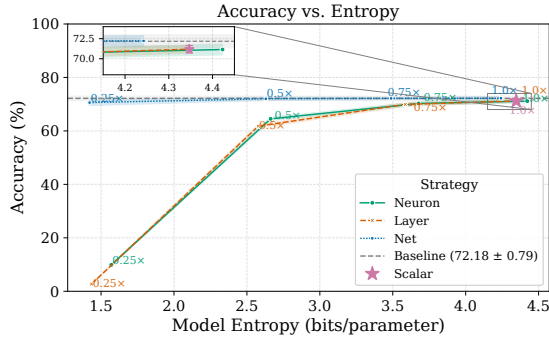
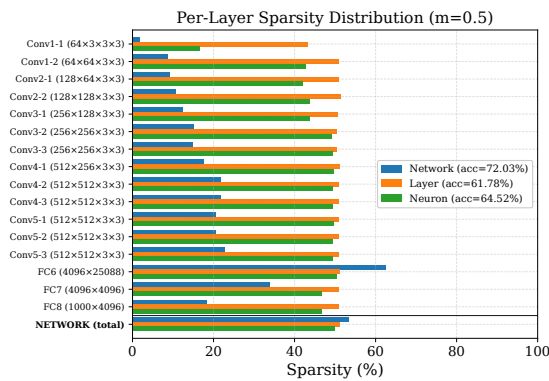


Fig. 3: Accuracy vs. Entropy (bits/parameter).

duce similar total sparsity (approximately 50%), yet their accuracies differ substantially, while the Network mode achieves 72.03%, Layer and Neuron modes reach only 61.78% and 64.52% respectively, as seen in Figure 3. In the Layer and Neuron approaches, each layer receives an SPT budget proportional to its size, and the convolutional layers are pruned with the same intensity as the fully connected layers. In Fig. 4, the Layer and Neuron approaches present similar sparsities at all layers, while the Network approach leads to low sparsity in the convolutional layers and high sparsity in the fully connected classifier.

The fully connected layers FC6, FC7, and FC8 account for 123.6M of the network's 138.4M parameters (89.3%). The Network-wise MPGBP aggressively sparsifies these layers, zeroing more than 60% of FC6's parameters, while more SPTs per parameter are used in the convolutional layers. The convolutional layers, despite representing only 10.7% of the parameters, are responsible for features extraction, which feeds classification. The results indicate that preserving these parameters is crucial for network performance, requiring finer representation. Classifier layers are highly redundant and tolerate aggressive quantization. The network approach greedy pursuit naturally produces this adaptive allocation without any explicit layer-importance configuration.


 Fig. 4: Per-layer sparsity distribution at $m = 0.50$ for the three MPGBP granularity modes.

VI. CONCLUSION

This paper investigated the application of the MPGBP algorithm for post-training weight compression of deep neural networks using SOPOT representation. The key findings are as follows. First, joint network compression consistently

outperforms per-layer and per-neuron strategies at equal total SPT budget. At $m = 0.50$, the network approach achieves baseline accuracy (72.18%) with 0.50 SPTs per parameter and an entropy-based rate of 2.63 bits/param, corresponding to $12.2\times$ compression. Second, MPGBP naturally induces sparsity without explicit pruning: 53.3% of weights are zeroed at $m = 0.50$ with no accuracy loss, reducing both storage requirements and the number of multiply-accumulate operations during inference. Third, the SOPOT representation ensures that all surviving weights are sums of powers of two, enabling multiplierless inference through shift-and-add operations only.

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [2] Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, p. 60, 2023.
- [3] J.-y. Li, Y.-k. Zhao, Z.-e. Xue, Z. CAI, and Q. LI, "A survey of model compression for deep neural networks," *Chinese Journal of Engineering*, vol. 41, no. 10, pp. 1229–1239, 2019.
- [4] 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*, 2015.
- [5] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*, pp. 169–175, IEEE, 2018.
- [6] E. A. B. DA SILVA, L. Lovisolo, A. J. Dutra, and P. S. Diniz, "Fir filter design based on successive approximation of vectors," *IEEE transactions on signal processing*, vol. 62, no. 15, pp. 3833–3848, 2014.
- [7] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," *Advances in neural information processing systems*, vol. 29, 2016.
- [8] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.
- [9] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.
- [10] 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.
- [11] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.
- [12] L. F. d. S. Coelho, L. Lovisolo, and M. P. Tcheou, "Adaptive filtering with reduced computational complexity using sopot arithmetic," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.
- [13] X. Geng, S. Liu, J. Jiang, K. Jiang, and H. Jiang, "Compact powers-of-two: An efficient non-uniform quantization for deep neural networks," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2024.
- [14] D. Przewlocka-Rus, S. S. Sarwar, H. E. Sumbul, Y. Li, and B. De Salvo, "Power-of-two quantization for low bitwidth and hardware compliant neural networks," *arXiv preprint arXiv:2203.05025*, 2022.
- [15] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–37, 2023.
- [16] L. F. D. S. Coelho, P. S. Diniz, D. Le Ruyet, and L. Lovisolo, "Multiplierless mlp using successive vector approximation in post-training quantization," in *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2025.
- [17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.
- [18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, IEEE, 2009.
- [19] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," 2010. <https://pytorch.org/vision/stable/models.html>. Pretrained weights: VGG16_Weights.IMAGENET1K_V1.