Examining and Mitigating the Impact of Crossbar Non-idealities for Accurate Implementation of Sparse Deep Neural Networks

Abhiroop Bhattacharjee[1], Lakshya Bhatnagar[2], and Priyadarshini Panda[1]
[1]Department of Electrical Engineering, Yale University, USA
[2]Indian Institute of Technology, Delhi, India
{abhiroop.bhattacharjee, priya.panda}@yale.edu

***Abstract*—Recently several structured pruning techniques have been introduced for energy-efficient implementation of Deep Neural Networks (DNNs) with lesser number of crossbars. Although, these techniques have claimed to preserve the accuracy of the sparse DNNs on crossbars, none have studied the impact of the inexorable crossbar non-idealities on the actual performance of the pruned networks. To this end, we perform a comprehensive study to show how highly sparse DNNs, that result in significant crossbar-compression-rate, can lead to severe accuracy losses compared to unpruned DNNs mapped onto non-ideal crossbars. We perform experiments with multiple structured-pruning approaches (such as, C/F pruning, XCS and XRS) on VGG11 and VGG16 DNNs with benchmark datasets (CIFAR10 and CIFAR100). We propose two mitigation approaches - Crossbar-column rearrangement and Weight-Constrained-Training (WCT) - that can be integrated with the crossbar-mapping of the sparse DNNs to minimize accuracy losses incurred by the pruned models. These help in mitigating non-idealities by increasing the proportion of low conductance synapses on crossbars, thereby improving their computational accuracies.**

***Index Terms*—Structured-pruning, crossbars, non-idealities, crossbar column-rearrangement, weight-constrained-training**

I. Introduction

The previous decade has seen the rise of Deep Neural Networks (DNNs) to solve various real world problems. To this end, memristive crossbar architectures have received significant attention to realize DNNs in an analog manner on hardware with high degree of parallelism, great compactness and energy-efficiency [1], [2]. Several Non-Volatile-Memory (NVM) devices such as, Resistive RAM (ReRAM), Phase Change Memory (PCM) and Spintronic devices have been explored as synapses to implement DNNs on crossbars [3].

In the recent years, several crossbar-aware pruning techniques have been devised that yield sparse DNN models [4]–[7]. Owing to their high sparsity, these models require significantly lower number of crossbars to be mapped, thereby introducing hardware resource-efficiency not only in terms of crossbars but also peripheral circuits interfacing the crossbars. Pruning algorithms such as, [4]–[7], produce structured sparsity in DNNs that fit into crossbars as dense weight matrices [8]. These structured pruning algorithms claim to preserve the accuracy of the pruned DNNs, after implementation on crossbars, with minimal or no noticeable loss, while bringing in high energy- and area-efficiencies. However, analog crossbars possess several non-idealities such as, interconnect parasitics, non-linearities/variations in the synapses, etc. [2], [9], [10]. These non-idealities result in imprecise dot-product currents in the crossbars, leading to performance (accuracy) degradation on mapping DNNs. Many works have modelled these non-idealities and studied their repercussions on the performance and robustness of crossbar-mapped DNNs, and have suggested mitigation techniques [2], [9]–[12]. Though the above-mentioned crossbar-aware structured pruning algorithms have claimed to preserve the performance of the pruned DNNs, none of them have included the impact of the non-idealities during inference on crossbars. For a realistic hardware evaluation of the performance of increased structured sparsity in DNNs mapped on crossbars, the inclusion of hardware non-idealities is critical. Thus, this work is designed to draw the focus of the research community towards a non-ideality aware evaluation of the various existing structured pruning algorithms and showing how increased sparsity can degrade the performance of DNNs on non-ideal crossbars. In the end, we also introduce two hardware-centric non-ideality mitigation strategies, namely crossbar-column rearrangement and *Weight-Constrained-Training* (WCT), to help improve the performance of the sparse DNNs on crossbars.

The key contributions of this work are as follows:

- We find that DNNs with high degree of structured sparsity suffer from greater performance degradation during inference on non-ideal crossbars with respect to unpruned DNN models.
- There exists a *trade-off*: as structured sparsity increases in DNNs, their mappings onto non-ideal crossbars become more resource-efficient in terms of area and energy, however with loss of performance (inference accuracy).
- We propose a hardware-friendly non-ideality mitigation strategy called crossbar-column rearrangement, that increases the feasibility of low conductance synapses in crossbars, thereby reducing the impact of non-idealities.
- Finally, we integrate the crossbar-aware pruning techniques with *Weight-Constrained-Training* (WCT), and show its effectiveness in non-ideality mitigation in sparse DNNs, especially on larger crossbars.

II. Background and Motivation

A. Analog crossbars and their non-idealities

Memristive crossbars are used to realize Multiply-and-Accumulate (MAC) operations on hardware in an analog manner. Crossbars receive the input activations of a DNN as analog voltages and produce currents analogous to the outputs of MAC operations. Ideally, the MAC operations occur using

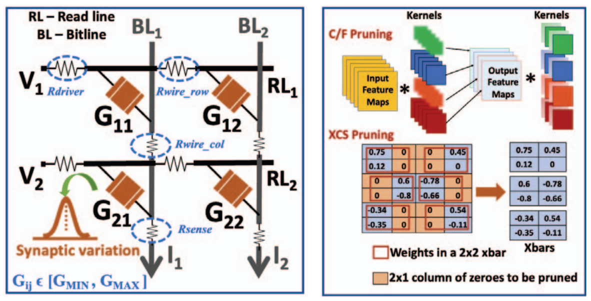

Fig. 1: (a) Non-ideal crossbar with input voltages V_i, synaptic conductances G_{ij} and output currents $I_j = \sum_i G_{ij} * V_i$. The interconnect and synaptic non-idealities, that lead to imprecise dot-product currents, are annotated; (b) **Top -** A representation of channel/filter pruning (C/F pruning). The blurred channels/filters correspond to DNN weights pruned in a structured manner. **Bottom -** A representation of XCS pruning

Ohm's Law and Kirchoff's current law with the interaction of the input voltages and the memristive conductances for the synapses (programmed between G_{MIN} and G_{MAX}). However, the analog nature of the computation leads to various non-idealities, such as, circuit-level interconnect parasitics and synaptic-level non-linearities or variations [2], [10], [11], [13].

Fig. 1(a) describes the equivalent circuit for a memristive crossbar consisting of various circuit-level and device-level non-idealities, *viz.* $Rdriver$, $Rwire\_row$, $Rwire\_col$ and $Rsense$ (interconnect parasitics), modelled as parasitic resistances and variations/non-linearities in the memristive synapses. The impact of these non-idealities can be incorporated by transforming the ideal memristive conductances $G_{ij(ideal)}$ to non-ideal conductances $G_{ij(non-ideal)}$. Consequently, the net output current sensed at the end of the crossbar-columns ($I_{non-ideal}$) deviates from its ideal value (I_{ideal}). This manifests as accuracy degradation for DNNs mapped onto crossbars. The relative deviation of $I_{non-ideal}$ from its ideal value is measured using *non-ideality factor* (NF) [11]. It is defined as $NF = (I_{ideal} - I_{non-ideal})/I_{ideal}$. NF is a direct measure of crossbar non-idealities, *i.e.* increased non-idealities induce a greater value of NF, degrading the performance of the DNN mapped onto them.

B. Crossbar-aware structured pruning of DNNs

In the recent years, there have been numerous works on structured pruning of DNNs, such as *channel/filter pruning* or C/F pruning (see Fig. 1(b)-Top) wherein the unimportant filters and channels in a DNN (corresponding to rows and columns in the weight matrix of the DNN) are pruned to obtain a sparse 2D weight matrix [4], [5]. These pruned models result in significant hardware savings in terms of reduced number of crossbars for mapping, thereby bringing in energy- and area-efficiency for DNN implementation. Likewise, other crossbar-aware pruning strategies include *Crossbar Column Sparsity* (XCS) [6] or *Crossbar Row Sparsity* (XRS) [7] (XCS shown in Fig. 1(b)-Bottom), that exploit fine-grained sparsity by respectively pruning columns or rows of weights within a crossbar [8]. Additionally, these works have claimed to preserve the inference accuracy of the structure-pruned networks on crossbars with minimal or no discernible performance loss with respect to the unpruned ones. *However, none of the previous works have accounted for the non-idealities inherent in crossbar arrays which raises concerns about the claimed performance of the highly pruned models in the real scenario.*

III. Methodology and Evaluation Framework

In this work, we structure-prune DNNs at initialization with a given sparsity ratio (s) for each layer [14], using the crossbar-aware techniques specified in Section II-B, and then train the respective pruned models on software. Traditionally, standard methods of obtaining sparse neural networks include training from scratch, followed by structured pruning and finally fine-tuning the pruned models [8]. However, structured pruning at initialization followed by training reduces training overheads since it requires only one round of training as opposed to the traditional approach requiring two rounds [14], [15].

Next, we use a simulation framework in Pytorch (see Fig. 2) to map the trained DNNs onto non-ideal memristive crossbars and investigate the cumulative impact of the circuit and device-level non-idealities on their performance during inference. In the platform, a Python wrapper is built that unrolls each and every convolution operation in the software DNN into MAC operations. This yields 2D weight matrices for each DNN layer which are to be partitioned into numerous crossbar instances of a given size (16×16, 32×32 or 64×64). Before partitioning, based on the structured pruning approach we apply the following transformations T on the sparse weight matrices W:

1) $T(W)$ **for C/F pruning:** Here, for a given 2D weight matrix of a DNN layer, all the columns bearing zero values are eliminated. Further, we also eliminate rows of the weight matrix of the next DNN layer that interact with the output feature maps corresponding1 to the columns of zero values in the previous layer.
2) $T(W)$ **for XCS (or XRS):** Here, within a given 2D weight matrix of a DNN layer, there are chunks of successive zero weight vectors of the size of crossbar-column (or crossbar-row) (see Fig. 1(b)-Bottom) which are eliminated.

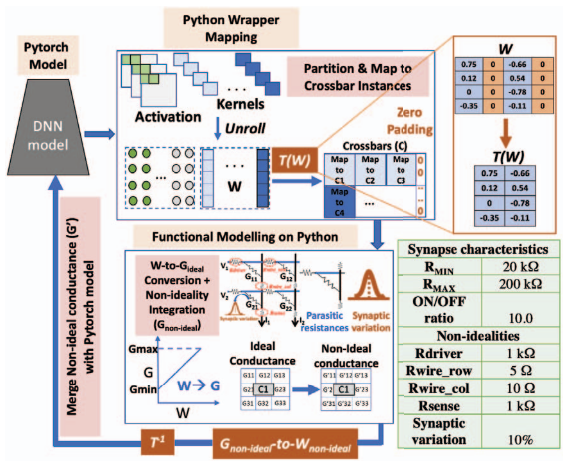

Fig. 2: Hardware evaluation framework in Python to map structure-pruned DNNs on non-ideal crossbars, followed by inference on the crossbar-mapped DNN models

Note, for standard unpruned DNNs, no transformation $T(W)$ is needed. The resulting transformed weight matrices are then partitioned into multiple crossbar instances. The next stage (functional modelling) of the platform converts

TABLE I: Table showing software accuracies and crossbar-compression-rates (with 32×32 crossbars) for the various DNN models with CIFAR10 and CIFAR100 datasets

Dataset: CIFAR10	**Software Accuracy (%) || Crossbar-compression-rate**			
Network	**Unpruned**	**C/F** ($s = 0.8$)	**XCS** ($s = 0.8$)	**XRS** ($s = 0.8$)
VGG11	83.6 || –	83.5 || 19.69×	83.28 || 4.26×	82.67 || 4.88×
VGG16	84.48 || –	83.65 || 19.60×	82.06 || 5.57×	83.47 || 4.89×
Dataset: CIFAR100	**Software Accuracy (%)**			
Network	**Unpruned**	**C/F** ($s = 0.6$)		
VGG11	53.29 || –	52.72 || 5.64×		
VGG16	51.83 || –	50.55 || 4.20×		

the weights in the crossbars to suitable conductances G (between G_{MIN} and G_{MAX}). Thereafter, the circuit-level non-idealities are integrated as parasitic resistances via circuit laws (Kirchoff's laws and Ohm's law) and linear algebraic operations written in Python [2], [9]. Further, the variations/non-linearities in the synapses are included with Gaussian profiling. In this work, we follow a *device agnostic* approach for analyzing the impact of intrinsic circuit-level and synaptic crossbar non-idealities on DNNs during inference. The various synapse parameters (e.g., R_{MIN}, R_{MAX}, device ON/OFF ratio) and values of the non-idealities used for our experiments have been listed in the table shown in Fig. 2. The non-ideal synaptic conductances G' are then converted back into non-ideal weight values and finally, for each DNN layer, we recombine all the crossbars and apply the inverse transformation T^{-1} to obtain 2D matrices of non-ideal weights (W'). W's are integrated into the original Pytorch based DNN model to conduct inference.

IV. Experiments

In this work, we train VGG11 and VGG16 DNNs with structured sparsity (via C/F pruning, XCS or XRS) using benchmark datasets such as, CIFAR10 and CIFAR100. For the experiments with CIFAR10 dataset, the sparsity is set as $s = 0.8$, while with CIFAR100 dataset, the sparsity is $s = 0.6$. The unpruned and pruned DNN models are trained to have nearly equal software accuracies to conduct a fair comparison of the impact of non-idealities when the models are mapped onto non-ideal crossbars (see Table I). The crossbar-compression-rates for the pruned DNNs on 32×32 crossbars are also shown in Table I.

V. Impact of Non-idealities on Pruned DNNs

The sparse DNNs have greatly reduced number of parameters than their unpruned counterparts which results in significantly lesser number of crossbars on hardware (see Table I). However, the fewer parameters remaining in the sparse DNNs are crucial for the model's performance. Thus, any non-ideality interfering with the fewer parameters of the sparse DNNs would have huge impact on the performance. In Fig. 3(a), we find that for the VGG11/CIFAR10 model, the DNNs with structured sparsity (via C/F pruning, XCS, XRS with $s = 0.8$) suffer greater accuracy degradation than their unpruned counterparts for crossbar sizes ranging from 16×16 to 64×64. Further, as we increase the crossbar size, both the accuracies of unpruned and pruned networks decline owing to increase in crossbar non-idealities [9], [11]. Specifically, on 64×64 crossbars, the inference accuracy of the unpruned model reduces by $\sim 21\%$ with respect to the software baseline while, for the sparse DNNs pruned via C/F pruning, XCS and XRS, the decline is $\sim 39\%$, $\sim 24\%$ and $\sim 30\%$, respectively. Also, in Fig. 3(b), we find that on reducing the extent of sparsity in the C/F pruned DNNs from $s = 0.8$ to $s = 0.5$, the performance degradation suffered by the pruned DNNs is reduced. This validates the fact that greater sparsity, although leads to energy- and area-efficient mappings on crossbars, increases the interference of crossbar non-idealities, thereby hampering the performance of the pruned networks.

In Fig. 3(c), for the VGG16 DNN with CIFAR10 dataset, the trends are similar to the case of the VGG11 DNN for XCS, XRS and C/F pruning ($s = 0.8$) in case of 16×16 and 32×32 crossbars. However, in case of a larger 64×64 crossbar, we find that the performance of the network pruned by C/F pruning exceeds that of the unpruned network. This is because unpruned DNNs require a larger absolute number of crossbars for mapping than pruned ones. As a result, the value of NF is expected to increase at a higher rate for unpruned DNN on moving from 32×32 to 64×64 crossbars (see Fig. 3(d)). So, for larger crossbars, the accuracy degradation for structure-pruned DNNs would decelerate compared to their unpruned counterparts, which can even lead to better absolute accuracy of the pruned networks than the unpruned ones.

VI. Non-Ideality Mitigation Strategies & Results

A. *Crossbar-Column rearrangement (R)*

For the sparse DNNs obtained via C/F pruning, we propose a simple hardware-friendly transformation of column rearrangement R after the transformation T (as discussed in Section III) before partitioning and mapping weights onto non-ideal crossbars. *This transformation is inspired from the fact that the impact of non-idealities (or non-ideality factor NF) reduces for crossbars with higher proportion of low conductance synapses* [10], [13]. Additionally, this approach of column rearrangement does not have any training overhead and is applied during the mapping of the DNNs onto crossbars.

Let us consider a 4×6 weight matrix W, after applying the transformation T, to be mapped onto 2×2 crossbars (see Fig. 3(e)). During the R transformation, we first compute the value of $(\mu \times \sigma)^{\frac{1}{2}}$ for each column from I-VI, where μ and σ respectively denote the mean and standard deviation of the absolute values of weights in each column. Thereafter, based on the increasing order of $(\mu \times \sigma)^{\frac{1}{2}}$, we rearrange columns I-VI in the manner shown. Now, in Fig. 3(f), we visualize the impact of R transformation on the weight matrices of the 3^{rd} and 5^{th} convolutional layers of the VGG16/CIFAR10 DNN (C/F pruned with $s = 0.8$) using heatmaps. Before applying the transformation, the lighter (low conductance synapse) and darker (high conductance synapse) points in the heatmaps are intermixed. Post transformation, the lighter points are concentrated at the center of the heatmaps and darker points are mostly near the peripheries. Thus, post R transformation and partitioning, individual crossbars have greater proportions of low conductance synapses, thereby mitigating the impact of crossbar non-idealities.

CIFAR10 & CIFAR100 Results: Fig. 4(a-b) & Fig. 4(c-d) show that R transformation improves the performance of the C/F pruned VGG11 and VGG16 DNNs. Specifically, $\sim 9\%$ ($\sim 6\%$) improvement in accuracy is observed for VGG11 (VGG16) DNN on 64×64 (32×32) crossbars with CIFAR10 dataset. We also find that on 32×32 crossbars, the accuracy of the pruned VGG16/CIFAR100 DNN post R transformation is $\sim 3\%$ greater than the unpruned counterpart.

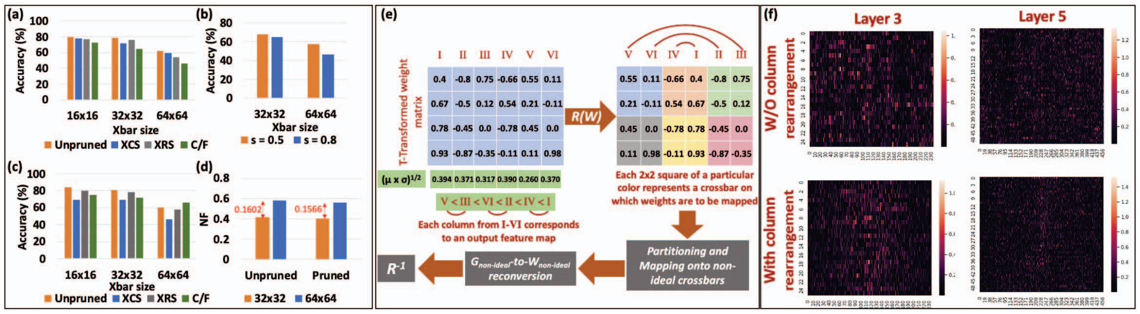

Fig. 3: Plot of inference accuracy versus crossbar size for- (a) unpruned and structure-pruned ($s = 0.8$) VGG11/CIFAR10 DNN; (b) different values of sparsity (s) of a C/F pruned VGG11/CIFAR10 DNN; (c) unpruned and structure-pruned ($s = 0.8$) VGG16/CIFAR10 DNN; (d) Plot showing the variation in average NF for unpruned and C/F pruned weight matrices on increasing the crossbar size from 32×32 to 64×64; (e) Pictorial representation of R transformation integrated with our hardware evaluation framework. Note, in our hardware evaluation framework, when R transformation is applied before partitioning and mapping the weights onto non-ideal crossbars, an inverse transformation R^{-1} is applied after the integration of crossbar non-idealities to correctly carry out inference with the DNN model; (f) Heatmaps to visualize the impact of R transfromation on the weight matrices of 3^{rd} & 5^{th} layers of the VGG16/CIFAR10 DNN trained with C/F pruning with $s = 0.8$

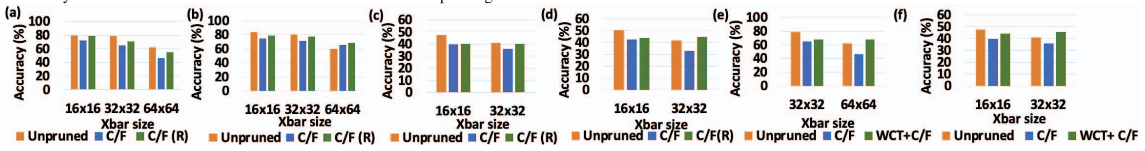

Fig. 4: A plot of inference accuracy versus crossbar size for unpruned, C/F pruned and- (a) C/F pruned with transformation R ($s = 0.8$) VGG11/CIFAR10 DNNs; (b) C/F pruned with transformation R ($s = 0.8$) VGG16/CIFAR10 DNNs; (c) C/F pruned with transformation R ($s = 0.6$) VGG11/CIFAR100 DNNs; (d) C/F pruned with transformation R ($s = 0.6$) VGG16/CIFAR100 DNNs; (e) WCT+C/F pruned ($s = 0.8$) VGG11/CIFAR10 DNNs; (f) WCT+C/F pruned ($s = 0.6$) VGG11/CIFAR100 DNNs

B. *Weight-Constrained-Training (WCT)*

We introduce *Weight-Constrained-Training* (WCT) of the structure-pruned DNN models on software before mapping onto crossbars, motivated by a recent work called Non-linearity Aware Training (NEAT) [10]. In WCT, based on the weight distributions of all the layers of a trained DNN, we heuristically determine a cut-off value W_{cut} for its weights, and then apply the following transformation on the weights of the DNN: $W = min\{|W|, W_{cut})\} * sign(W)$. This transformation constrains the DNN weights in the interval $[-W_{cut}, W_{cut}]$. With the above transformation, the DNN is iteratively trained for 2 epochs, to maintain nearly iso-accuracy with baseline. Note, the iterative training via WCT does not add any computational overhead to the overall training time, thereby making it a viable choice. The WCT-DNN results in greater proportion of low conductance states on the crossbars, thus, reducing the impact of crossbar non-idealities. The resultant sparse WCT-DNNs are then mapped onto crossbars. In Fig. 4(e-f), we find that the WCT-DNNs maintain their performance even on increasing the crossbar size, making them resilient against crossbar non-idealities. Further, WCT-DNNs have better accuracy than the C/F pruned DNNs on crossbars. Specifically, with CIFAR10 (CIFAR100) dataset, the WCT-DNN has $\sim 6\%$ ($\sim 7\%$) higher accuracy than the unpruned model on 64×64 (32×32) crossbars.

VII. Conclusion

This work elucidates that DNNs with high degree of structured sparsity can lead to poor performance on non-ideal analog crossbars, albeit being more hardware-efficient. In our study, we have considered structured-pruning techniques, such as C/F pruning, XCS and XRS, and have shown that increased sparsity leads to loss in inference accuracy of the pruned DNNs on crossbars (considering non-idealities) with respect to their unpruned counterparts. To mitigate the non-ideality induced performance degradation, we introduce two hardware-centric strategies (crossbar-column rearrangement and WCT) that can be integrated with the mapping of the DNNs onto non-ideal crossbars to reduce the impact of non-idealities, thereby enhancing the performance of the sparse DNNs on crossbars.

Acknowledgement

This work was supported in part by C-BRIC, a JUMP center sponsored by DARPA and SRC, the National Science Foundation (Grant#1947826) and the DARPA AI Exploration (AIE) program.

References

[1] P. Chi *et al.*, “Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory,” *ACM SIGARCH Computer Architecture News*, 2016.
[2] S. Jain *et al.*, “Rxnn: A framework for evaluating deep neural networks on resistive crossbars,” *IEEE TCAD*, 2020.
[3] Chakraborty *et al.*, “Pathways to efficient neuromorphic computing with non-volatile memory technologies,” *Applied Physics Reviews*, 2020.
[4] W. Wen *et al.*, “Learning structured sparsity in deep neural networks,” *arXiv preprint arXiv:1608.03665*, 2016.
[5] P. Wang *et al.*, “Snrram: An efficient sparse neural network computation architecture based on resistive random-access memory,” *DAC*, 2018.
[6] L. Liang *et al.*, “Crossbar-aware neural network pruning,” *IEEE Access*, 2018.
[7] J. Lin *et al.*, “Learning the sparsity for reram: Mapping and pruning sparse neural network for reram based accelerator,” *ASP-DAC*, 2019.
[8] C. Chu *et al.*, “Pim-prune: Fine-grain dcnn pruning for crossbar-based process-in-memory architecture,” *DAC*, 2020.
[9] A. Bhattacharjee, A. Moitra, and P. Panda, “Efficiency-driven hardware optimization for adversarially robust neural networks,” *DATE*, 2021.
[10] A. Bhattacharjee *et al.*, “Neat: Non-linearity aware training for accurate, energy-efficient and robust implementation of neural networks on 1t-1r crossbars,” *IEEE TCAD*, 2021.
[11] I. Chakraborty *et al.*, “Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks,” *DAC*, 2020.
[12] B. Liu *et al.*, “Reduction and ir-drop compensations techniques for reliable neuromorphic computing systems,” *ICCAD*, 2014.
[13] A. Bhattacharjee *et al.*, “Switchx: Gmin-gmax switching for energy-efficient and robust implementation of binary neural networks on reram xbars,” *arXiv preprint arXiv:2011.14498*, 2021.
[14] J. Frankle *et al.*, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *arXiv preprint arXiv:1803.03635*, 2018.
[15] E. Malach *et al.*, “Proving the lottery ticket hypothesis: Pruning is all you need,” *ICML*, 2020.