STAR: Sparse Thresholded Activation under partial-Regularization for Activation Sparsity Exploration

Zeqi Zhu[*†], Arash Pourtaherian[*], Luc Waeijen[*], Egor Bondarev[†], Orlando Moreira[*]
[*]GrAI Matter Labs, [†]Eindhoven University of Technology
{z.zhu, e.bondarev}@tue.nl, {apourtaherian, lwaeijen, omoreira}@graimatterlabs.ai

Abstract

*Brain-inspired event-driven processors execute deep neural networks (DNNs) in a sparsity-aware manner. Specifically, if more zeros are induced in the activation maps, less computation will be performed in the succeeding convolution layer. However, inducing activation sparsity in DNNs remains a challenge. To address this, we propose a training approach STAR (**S**parse **T**hresholded **A**ctivation under partial-**R**egularization), which combines activation regularization with thresholding, to overcome the barrier of a single threshold- or regularization-based method in sparsity improvement. More precisely, we employ the sparse penalty on the near-zero activations to fit the activation learning behaviour in accuracy recovery, followed by thresholding to further suppress activations. Experimental results with SOTA networks (ResNet50/MobileNetV2, SSD, YOLOX and DeepLabV3+) on various datasets (Cifar-100, ImageNet, KITTI, VOC2007 and CityScapes) show that STAR can reduce on average 54% more activations compared to ReLU suppression. It outperforms the state-of-the-art by a significant margin of 35% in activation suppression without compromising accuracy loss. Additionally, a case study for a commercially-available event-driven hardware architecture, Neuronflow [29], demonstrates that the boosted activation sparsity in ResNet50 can be efficiently translated into latency reduction by up to 2.78×, FPS improvement by up to 2.80×, and energy savings by up to 2.09×. STAR elevates event-driven processors as a superior alternative to GPUs for Edge computing.*

1. Introduction

Deep neural networks (DNNs) have become ubiquitous in AI applications as their related architectures have dominated various AI challenges in the past decade. The great success of DNNs in academia attracts considerable interest from the industry. However, the general trend in AI research has been to make deeper and more complicated DNNs to achieve better performance [5, 45]. This strongly defies the core requirements of industrial applications – low resource consumption and high efficiency in power and latency – especially for deployment on edge platforms. Consequently, various novel AI computing architectures have been proposed in recent years to improve inference efficiency in terms of power and latency. Among them, the event-driven architecture is one of the most innovative paradigms for reducing the latency and computational burden of DNNs by mimicking the working mechanisms of the human brain.

A plethora of event-driven architectures have been proposed by the academia, e.g., DYNAPs [28], SpiNNaker 2 [20], and industry, e.g., TrueNorth [1], Akida [43], Loihi 1, 2 [8, 9], Neuronflow [29], to overcome the unavoidable problem of high latency and energy consumption on conventional processors. Those architectures emulate the brain's energy and compute efficiency by executing the networks in an asynchronous and parallel event-driven manner, which promotes and simultaneously exploits sparse computations. Specifically, in event-driven processing, skipping one event eliminates the accompanying computations and memory accesses. With a fraction of events skipped, the overall compute and memory-access requirements decrease proportionally [48]. This reduction in compute and memory-access can be directly transferred to latency and energy efficiency on the silicon. Thus, an effective event suppression approach is essential to maximize the efficiency of event-driven processing.

To suppress events, it is critical to understand the execution mechanism of DNNs on event-driven architectures. In general, an activated neuron is equivalent to the generation of an event with a binary, integer, or floating-point value. No event needs to be fired once the activation value is zero. Therefore, inducing zeros in DNN's activation is beneficial for the event reduction in inference. Remarkably, the majority of network architectures adopt ReLU [18] as their activation function, which statistically suppresses half of the network activations. This degree of activation sparsity improves the inference performance by roughly 2 times through event-driven processing. However, it is still inadequate to thoroughly demonstrate the efficiency of event-

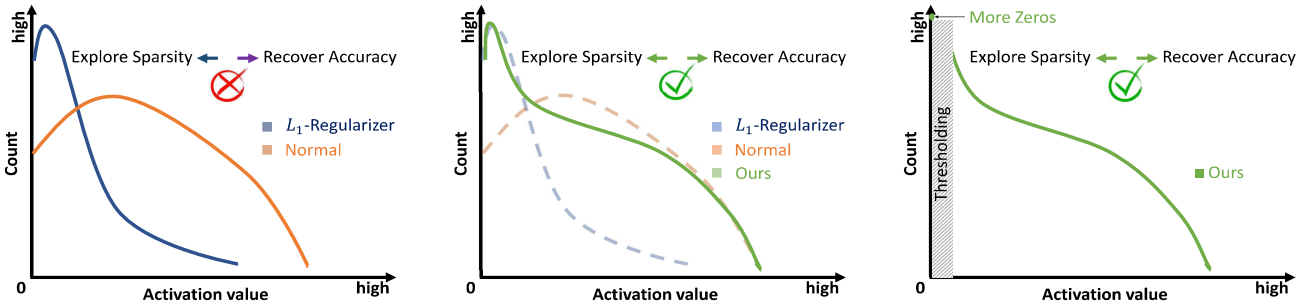

Figure 1. Our proposed STAR improves the activation distribution of a sparsified model and further boosts its sparsity through thresholding. The activation distributions from actual data are illustrated in Fig. 10 and Fig. 11 in Appendix B.2

driven architectures.

Previous studies on activation suppression have mostly been conducted in two directions: threshold-based and regularizer-based. Existing works on threshold-based suppression [2, 27, 34] use hard thresholds solely in inference to omit low-magnitude activation nodes. The increase of threshold values can result in more sparsity, yet erode model performance. To recover accuracy from aggressive suppression, we propose network retraining to adjust the weights to higher thresholds. This is a simple heuristic from weight suppression [46], and yet we prove that it also works well in activation suppression.

Regularizer-based approaches, as another direction, have drawn much attention in recent studies [17,24,48]. The idea is to penalize the activations with a sparse regularizer so that they are driven towards zero. However, the problem is that increasing the weight of regularization loss in total training loss can result in more zeros in activation while compromising accuracy [30, 48]. We hypothesize that this substantial decrease in accuracy is due to the fact that the excessive sparse penalty disrupts the output distribution of activations and inhibits the model from learning effective features, as illustrated in Fig. 1-left. More precisely, existing suppression techniques (i.e., L_1-regularizer) do the opposite to normal training in activation distribution learning. To improve this, we consider applying the regularization on those small-magnitude activations, named as partial-regularization, so that the output distribution of high-magnitude activations can be restored to the high-accuracy level, as portrayed in Fig. 1-middle. More details of activation behaviour are revealed in Sec. 2.

To compensate for the deficiencies of a single threshold- or regularizer-based method in sparsity exploration, we propose STAR (Sparse Thresholded Activation under partial-Regularization), which combines partial-regularization with thresholding to significantly suppress the activations in DNNs while maintaining accuracy (see Fig. 1-right).

STAR presents several significant advantages. Firstly, it surpasses existing activation suppression techniques by a considerable margin in achieving sparsity while maintaining accuracy. Secondly, it employs a single power-of-2 threshold, simplifying hardware implementation and minimizing the efforts needed for threshold searching. Thirdly, it ensures stable sparsity in activations across frames. Furthermore, the method seamlessly integrates with other optimization techniques, such as weight pruning [14] and quantization [39], to further reduce the model size and MAC operations. Finally, given its specific design for the benefits of event-driven architectures, the use of STAR for event-driven processors results in a notable reduction in latency and energy consumption, leading to a substantially superior power/throughput when compared to GPUs.

To demonstrate the generalization and efficiency of STAR, we evaluate our approach on image classification using ResNet [19] and MobileNetV2 [36]. Given the likelihood of autonomous vehicles and surveillance systems being used in real-world scenarios, we broaden the scope of our analysis to semantic segmentation [6] and object recognition [15] tasks, that have more sophisticated yet under-explored network architectures.

In this paper, we bring four main contributions:

1. We introduce a novel mixed training approach STAR to effectively suppress non-zero activations in DNNs.
2. We simplify the threshold searching and hardware implementation by employing a power-of-2 global threshold.
3. We reveal the efficiency of STAR on multiple neural network architectures and popular vision applications.
4. We demonstrate the actual latency reduction, FPS improvement, and power savings by implementing STAR on a real-life event-driven processor GrAI-VIP (see Appendix C).

The remainder of this paper is structured as follows. Sec. 2

investigates the activation behaviour during activation suppression and accuracy recovery. Sec. 3 provides a detailed description of our approach, STAR. Sec. 4 presents the experimental setup and results. Sec. 5 gives an overview of the related work and the positive impact on the community. Sec. 6 concludes the paper.

2. Motivation

Our motivation for this work is based on an empirical observation that the activation learning pattern has a significant impact on the model's accuracy and its activation density throughout training. We study the activation pattern of a ResNet18 model [19] by tracking the movements of the average and maximum of its non-zero activations under various training conditions, as illustrated in Fig. 2. The first training phase (normal training) increases the maximum of activations along with the epochs while maintaining its average at a high level. In the second training phase (L_1-reg), both average and maximum of activations decline considerably due to the effect of L_1 penalty on the activation outputs. Finally, the third training phase (retrain) successfully recovers the accuracy by retraining the weights without L_1 constraint. We have observed that a substantial improvement in accuracy recovery is concomitant with an increase in both the mean and maximum values of non-zero activations, rather than their overall quantity. We use the term "activation renaissance" to refer to this occurrence.

We observe two opposing phenomena when comparing the results of the aforementioned experiments: 1) Improving accuracy necessitates an increase in non-zero activation values. 2) Suppressing activations requires the compression of activation values. Therefore, we deduce that this conflict prevents the prior regularizer-based methods from suppressing activations further while preserving accuracy

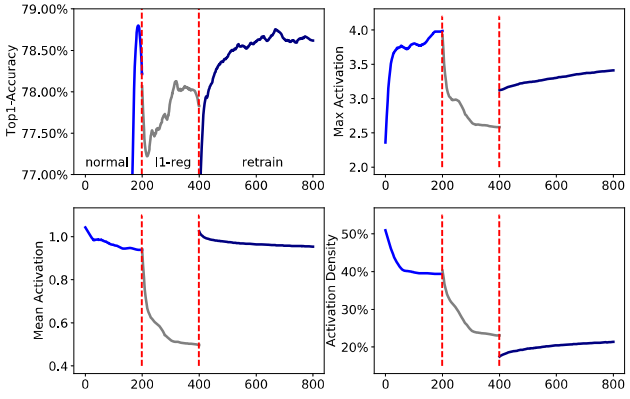

Figure 2. The impact of accuracy recovery on the activation behavior in normal training, regularizer-based training, and retraining on the regularized model.

by merely increasing the penalty. To address this problem, as explained in Fig. 1, we propose to apply a partial-regularization, solely to the near-zero activations. This is because low-magnitude activations are less important for performance [2, 24], which can be forced towards zero and masked out via thresholding. The removal of regularization on the remaining activations, on the other hand, allows them to grow and learn features similar to normal training. To support our statement, we provide a thorough comparison of various sparsity penalties on activation suppression in Fig. 4.

3. Method

In this section, we describe our activation suppression training method, STAR. The objective of STAR is to combine regularizer-based training with thresholding such that the percentage of non-zero activations can be maximally suppressed while maintaining model accuracy. As visualized in Fig. 3, the whole training procedure can be summarized in four stages:

1. ***Standard baseline Training:*** Train an accurate model for the target network and task;

2. ***Regularizer-based Training:*** Apply a layerwise sparse penalty on the activation outputs of the pre-trained model and suppress the activation for a number of epochs, then reduce the learning rate and fine-tune the network until the network's accuracy is recovered;

3. ***Threshold-based Training:*** Add threshold nodes to the regularized model graph and fine-tune the network using partial-regularization on activation until the network's accuracy is recovered;

4. ***Repeat:*** step-3 for n thresholds, where n is typically less than 3, and select the model paired with the threshold giving the optimal trade-off between accuracy and sparsity.

3.1. Naturally-Arising Sparse Activations

We start by examining the natural sparsity of activation maps in DNNs [19, 36]. Prior studies show that activation sparsity is linked with the ReLU non-linearity, defined as $ReLU(x) = max(x, 0)$. It results in an average 62% network-wide activation sparsity through standard network training [32, 35] (see Fig. 3). However, many newly proposed networks [15, 21, 36, 41] apply non-zero activation functions [11, 26, 33, 36] to seek exceptional performance. Unfortunately, these functions are less efficient on edge processors due to issues with post-quantization, sparsity exploration, and complex exponent computations. Therefore, they are commonly replaced by ReLU for hardware efficiency [42]. We adopt the same strategy in our study and introduce thresholding to sparsify the linear activation [36].

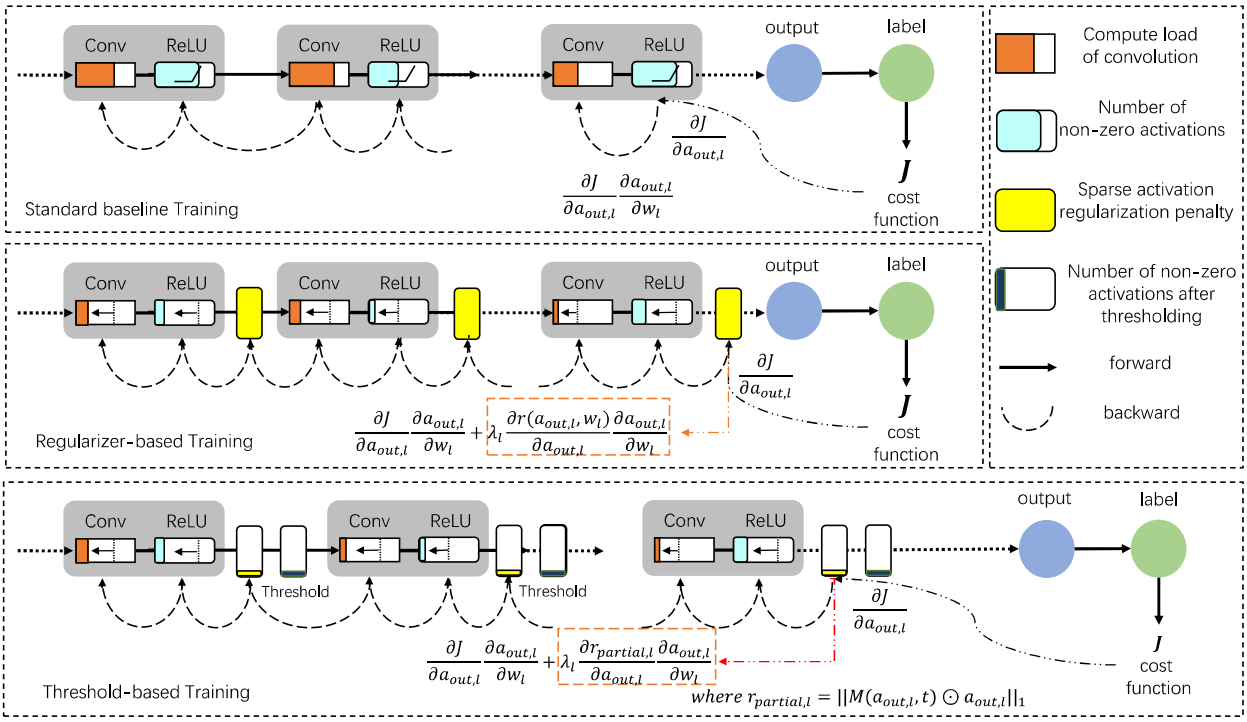

Figure 3. The workflow of the STAR suppression method.

3.2. Regularized Sparse Activations

To induce more activation sparsity, we apply an L_1 regularization to the output of ReLUs during training, as illustrated in the box "Regularizer-based Training" of Fig. 3. Therefore, in the training phase, weight parameters w_l in the l^{th} layer are updated by the gradients g_l, back-propagated from the loss J:

$$g_l = \frac{\partial J}{\partial a_{out,l}} \frac{\partial a_{out,l}}{\partial w_l} + \lambda_l \frac{\partial r_l(w_l, a_{out,l})}{\partial a_{out,l}} \frac{\partial a_{out,l}}{\partial w_l}, \quad (1)$$

where $a_{out,l}$ represents the l^{th} layer activation outputs, r is the activation regularization, and λ_l is the regularization coefficient used to balance sparsity exploration and accuracy recovery in network learning, which can be determined by grid-search [17, 24] or adaptive-learning [48]. As a result, the activation values are compressed towards zero (the black arrow in ReLU), resulting in relative compute reduction in the next convolution layer (the black arrow in Conv). However, as explained in Sec. 2, excessive regularization on the entire activation maps prevents the network from learning effective feature distributions. Thus, Eq. (1) can be further improved with our partial-regularization as follows:

$$g_{partial,l} = \frac{\partial J}{\partial a_{out,l}} \frac{\partial a_{out,l}}{\partial w_l} + \lambda_l \frac{\partial r_{partial,l}}{\partial a_{out,l}} \frac{\partial a_{out,l}}{\partial w_l}, \quad (2)$$

with

$$r_{partial,l} = ||M(a_{out,l}, t) \odot a_{out,l}||_1, \quad (3)$$

where

$$M(a_{out,l}, t) = \begin{cases} 1 & a_{out,l} \in (0, t) \\ 0 & otherwise \end{cases} \quad (4)$$

As shown in Eq. (2) and Eq. (3), both forward and backward passes of the regularization are masked out through Eq. (4), if the activation values are above the threshold t. This partial-regularization $r_{partial,l}$ computes the penalty solely for activations below t, allowing the remaining non-regularized activations to adjust to the desired value.

It is worth mentioning that various sparse regularizers (e.g., L_1, L_2, L_p, SCAD, Hoyer, Kullback-Leibler) have been introduced in previous research [13, 17, 24, 30, 31]. These regularizers are capable of further enhancing the activation sparsity through training while achieving very similar gains in final sparsity as depicted in Fig. 4. Therefore, in this paper, we focus on implementing the simple yet effective L_1 regularizer as the regularization term for Eq. (3). To take advantage of both the suppression ability of L_1 and the activation renaissance phenomenon, we propose a mixed-use of full-L_1 and partial-L_1 regularizers at different stages of the training process. Specifically, we employ a full activation penalty in the initial training stage to maximize the number of zeros and then switch to partial-regularization in the second stage for better accuracy recovery. This approach outperforms all other methods in activation suppression. Due to the page limits, more details of mentioned sparsity regularizers can be found in Appendix A.1.

3.3. Finetuning with Activation Thresholds

As depicted in the box labeled as "Threshold-based Training" in Fig. 3, thresholds are inserted right after the L_1 regularizer to suppress extra non-zero activations. Evidently, simply increasing the threshold values without retraining leads to significant accuracy degradation. Thus, we first apply a large threshold t to filter out a substantial percentage of forward activations and then fine-tune the weights to adjust to this threshold. As observed during activation learning, retraining causes the activation values to grow over the threshold for better accuracy recovery. To maintain the amount of below-threshold activations, it is essential to incorporate partial-regularization and thresholding simultaneously in this finetuning process. The algorithmic description can be found in the appendix.

Besides, the authors of [24] claim that the forward and backward information flow of a layer can be disrupted by a relatively high activation threshold, thereby preventing the network from learning. We mitigate this problem by using the Straight Through Estimator (STE) [4] method for passing the inhibited gradients by threshold onto the related weights as an estimator, described in Eq. (5) and Eq. (6). This threshold function solely filters forward events for high activation sparsity and bypasses the threshold operator backward for effective weight updates.

$$T(a_{out,l}) = \begin{cases} 0 & a_{out,l} \in (-\infty, t) \\ a_{out,l} & a_{out,l} \in [t, +\infty) \end{cases} \tag{5}$$

$$\frac{\partial(w_l, a_{out,l})}{\partial a_{out,l}} = \begin{cases} 0 & a_{out,l} \in (-\infty, 0) \\ 1 & a_{out,l} \in [0, +\infty) \end{cases} \tag{6}$$

Moreover, searching for an optimal threshold is a time-consuming endeavor. For a network with D layers and C channels, considering N threshold candidates, layer-wise thresholding provides $O(D \times N)$ possibilities for threshold combination, whereas channel-wise thresholding increases this complexity to $O(D \times C \times N)$. To simplify this process, we suggest implementing a power-of-two as the threshold (i.e., 2^n, $n \in Z$). As the sparse penalty compels all activations towards zero, a power-of-two is efficient in achieving fine-grained thresholding in the proximity of zero due to its dense value distribution in that area, visualized in Fig. 8. However, through our empirical analysis (see Appendix A.3 and Appendix B.3), we find that model-wise thresholding can yield similar suppression performance as layer-wise thresholding, providing a more hardware-friendly option for our method. Consequently, the overall search effort can be reduced to $O(log_2 N)$, making the compute logic less expensive and freeing up more memory for other purposes.

4. Evaluation

We describe the experimental setup and results to evaluate our proposed method (STAR) for activation suppression. First, we show how the mixed approach capitalizes on the benefits of full-regularization for activation suppression and partial-regularization for accuracy recovery. Following that, we conduct an ablation study to acquire a better knowledge of the impact of each component in STAR. Furthermore, we examine STAR across a wide range of networks to demonstrate its benefits in activation suppression compared to the state-of-the-art. Finally, we evaluate our optimized models on a commercial event-driven edge processor. The performance results demonstrate its efficiency in terms of latency and energy savings for event-driven processing.

4.1. Experimental Setup

Models: The primary network investigated in our study is ResNet [19] on Cifar-100 [23] and ImageNet [10], on which we perform extensive trials for various activation suppression approaches to demonstrate the sustainability and effectiveness of our proposed strategy. We additionally examine lightweight architectures like MobileNetV2 [36] to show the impact of our technique on networks of variable complexity. Nonetheless, such a simple validation on image classification is inadequate to achieve the objectives of sophisticated applications in industrial contexts. The generalization of our method is evidenced by the extension to other application domains, such as object detection (SSD [25], YOLOX [15] on KITTI [16] and VOC2007 [12]) and semantic segmentation (DeepLabV3+ [6] on Cityscapes [7]).

Training: Due to page limitation, the detailed training information can be found in Appendix B.1.

Environment: Our method STAR is implemented in TensorFlow (TF). We build software tools in Python to automate the customized implementation and monitor the optimization training process. The optimization training and model profiling are carried out on a single Nvidia GPU – Quadro RTX 5000. After optimization, the trained models are evaluated on GrAI-VIP [44], a commercially-available event-driven DNN accelerator by GrAI Matter Labs. It is a 12-nm taped-out chip with 144 SIMD-4 cores running at 650MHz. More details can be found in Appendix C.

4.2. Results on activation sparsity exploration

4.2.1 Comparison of various activation regularizers

We have experimented with various activation suppression methods. For the regularizer-based training, we implement the existing regularizers: L_1, L_2, *Hoyer*, and *SCAD*, and our two-stage observation-inspired approach (*Retrain w/o* L_1, described in Sec. 2) for sparsification. For a mixed approach combining regularization and thresholding, we employed the L_1 regularizer in the first stage to suppress activations, followed by fine-tuning with either *Full-*L_1 penalty *(*L_1 *w/ Thresh)* or *Partial-*L_1 penalty *(STAR)* along with thresholding in the second stage. The terms *Full-*L_1 and *Partial-*L_1

indicate the implementation of L_1 penalty on the entire and partial activation maps, respectively.

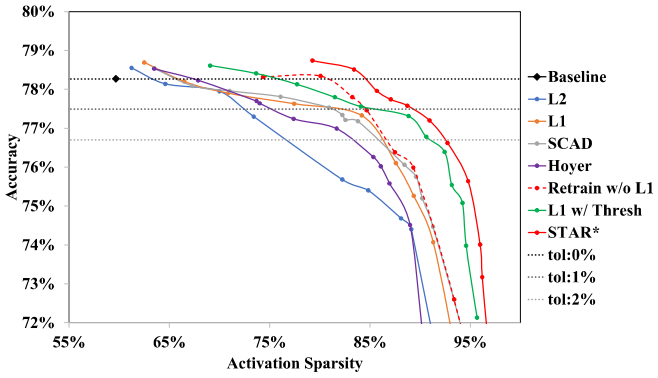

Figure 4. Accuracy-sparsity trade-offs of ResNet18 on Cifar-100 by applying various sparse regularizers w/ and w/o thresholding

As shown in Fig. 4, by comparing regularization-based methods under 0% loss tolerance, *Hoyer* outperforms the other regularizers in terms of sparsity gain because of its higher "diversity" in the activation values [24]. *Retrain w/o* L_1 follows the idea of diversifying the activation distribution by first sparsifying with *full-*L_1 and later removing the L_1 penalty for fine-tuning, which has a better effect on accuracy recovery while maintaining the sparsity gains from L_1. It further boosts the activation sparsity achieved by *Hoyer* from 67.86% to 80.07%.

Moreover, as we give more leeway on the acceptable drop in accuracy (tol: 1%, tol: 2%), both L_1 and *SCAD* surpass *Hoyer* in terms of suppression. *Hoyer* enforces activation x to approach zero if $x < \frac{\sum x^2}{\sum |x|}$, while x moves away from zero otherwise. It can potentially compromise the model's accuracy by distorting the natural distribution of activations when pushing large activations away from their original positions. Similarly, *full-*L_1 compresses large activations towards zero, resulting in reduced accuracy. As a result, the large-magnitude-preserving method *SCAD* outperforms L_1 when the penalty value increases for more sparsity. However, our two-stage approach, *Retrain w/o* L_1, consistently achieves more sparsity compared to any single regularization-based method, particularly when the accuracy drop tolerance is restricted.

4.2.2 Combined regularization and thresholding

The Pareto curves of two combination methods are presented in Fig. 4. The first one integrates the *Full-*L_1 regularization training with thresholding in the finetuning stage, named L_1 *w/ Thresh* while the second combination is the *STAR* method. Simply combining L_1 and thresholding (L_1 *w/ Thresh*) is able to outperform any single regularization-based method in sparsity exploration with a significant margin of 7.16% (the average boosted sparsity under $0 \sim 2\%$ tolerance), nevertheless, it surprisingly suppresses fewer activations than non-threshold method *Retrain w/o* L_1 while the accuracy tolerance is low ($\leq 1\%$). This observation reveals the fact that L_1 *w/ Thresh* emphasizes much on activation suppression while ignoring the importance of preserving large magnitude for accuracy recovery. On the contrary, our method *STAR* inherits the suppression ability from L_1 *w/ Thresh* and improves its deficiencies by allowing the growth of large activations in finetuning. The Pareto curve of *STAR* reveals that our approach can consistently and significantly improve sparsity gains beyond any existing methods, even when accounting for all accuracy drop tolerances.

4.2.3 Ablation studies

In order to estimate the impact of different components of STAR, we conduct complementary experiments: 1) applying L_1 penalty on full activations *(w/ full-*L_1*)*, 2) adding thresholds with or without retraining *(w/ post-train threshold, w/ threshold)*, 3) fine-tuning on the regularized model from experiment 1 without penalty but with thresholding *(w/o* L_1*, w/ threshold)*, 4) combining regularization with thresholding *(w/ full-*L_1*, w/ threshold)*, 5) applying partial regularization with thresholding *(STAR)*.

Tab. 1 shows the results of these experiments with ResNet50 on ImageNet, in terms of top-1 accuracy obtained on the validation set, its relative drop versus the baseline, activation density, activation reduction, MAC density and MAC reduction. The first line of the table reports the numbers for the normal-trained ResNet50 with ReLU sparsity, w.r.t. our baseline. We consider two models to be at the same accuracy level when their $\triangle acc \leq 0.1\%$ and we compute the suppression improvement by $(Act._{base} - Act._{new})/Act._{base}$.

Regularization on activation: The line labeled as *"w/ Full-*L_1*"* shows that regularization-based training can further reduce the number of activations and MACs by $1.53\times$ and $1.25\times$, respectively, on top of ReLU sparsity.

Threshold on activation: The line labeled as *"w/ Post-train threshold"* shows a minor reduction of 8.34% in activation density when simply applying a global threshold in post-training. Furthermore, the line labeled as *"w/ threshold"* illustrates that by increasing the threshold and retraining the model, it becomes possible to suppress an additional 14.16% of activations at the same accuracy level. These findings highlight the importance of finetuning with thresholding to achieve optimal performance.

Regularization and threshold on activation: By comparing *"w/o* L_1*, w/ threshold"* and *"w/ Full-*L_1*, w/ threshold"*, we notice that applying L_1 and thresholding simultaneously can preserve a majority of activations below the threshold

Table 1. Results and ablation studies of activation suppression with ResNet50 on ImageNet.

METHOD	VAL TOP-1 ACCURACY(%)	RELATIVE DROP(%) $100 \times \frac{O-B}{B}$	ACTIVATION DENSITY (%)	#REDUCTION $\frac{Act_B}{Act_O}$	MAC DENSITY (%)	#REDUCTION $\frac{MAC_B}{MAC_O}$
RES50-S [19]	76.64	0.00	48.94	1.00X	38.30	1.00X
W/ FULL-L_1	76.17	-0.61	32.00 (-16.94)	1.53X	30.53 (-18.41)	1.25X
W/ POST-TRAIN THRESHOLD	76.10	-0.70	40.60 (-8.34)	1.20X	31.09 (-17.85)	1.23X
W/ THRESHOLD	76.06	-0.76	34.84 (-14.10)	1.40X	26.05 (-22.89)	1.47X
AFTER L_1-REGULARIZED TRAINING						
W/O L_1, W/ THRESHOLD	76.12	-0.68	24.79 (-7.21)	1.97X	24.39 (-6.14)	1.57X
W/ FULL-L_1, W/ THRESHOLD	75.82 (-0.30)	-1.07	19.62 (-12.38)	2.49X	23.19 (-7.34)	1.65X
W/ PARTIAL-L_1, W/ THRESHOLD (**STAR**)	76.12	-0.68	**19.01** (-12.99)	**2.57X**	**22.11** (-8.42)	**1.73X**

as post-training thresholding (otherwise, the activation renaissance happens and the amount of non-zero activations will increase a bit). However, as explained in Sec. 2, this "indiscriminate" penalty probably damages the natural distribution of large activations, resulting in a 0.30% accuracy decrease compared to *"w/o L_1, w/ threshold"*. Therefore, by switching *full-L_1* to *partial-L_1*, STAR recovers the accuracy loss that occurred by *full-L_1* and improves the suppression performance of *"w/o L_1, w/ threshold"* by 23.31%. This verifies our assumption that applying *partial-L_1* can drive low-magnitude activations to stay below the threshold for more sparsity while allowing high-magnitude activations to increase for effective accuracy recovery.

In general, *STAR* results in a substantial 2.57× boost in activation suppression compared to the baseline, while incurring only a 0.68% relative accuracy drop. Additionally, we make an intriguing discovery that regularization and thresholding contribute approximately equally to total activation suppression, with regularization accounting for 16.94% and thresholding for 12.99%. These sparsity gains are in close proximity to the individual gains of 16.94% and 14.10% attained by the two methods, which demonstrates the efficacy of STAR in preserving the benefits of both techniques.

Table 2. Standard deviation of event-driven execution per frame. "-BS" and "-STAR" are the ReLU-sparsified model and the STAR-sparsified model, respectively.

MODELS	EVENT DENSITY [%] MEAN	EVENT DENSITY [%] STD	MAC DENSITY [%] MEAN	MAC DENSITY [%] STD
RES50-BS	49.70	1.68	38.33	0.90
RES50-STAR	21.29	1.18	22.11	0.66
MNV2-BS	63.56	0.88	75.97	0.38
MNV2-STAR	41.78	0.89	69.95	0.35
YOLOX-BS	51.89	0.62	39.51	0.46
YOLOX-STAR	27.07	1.22	19.09	0.65
DEEPLABV3+-BS	49.32	0.53	34.69	0.27
DEEPLABV3+-STAR	19.42	0.66	19.71	0.22

4.2.4 Stability across frames

Furthermore, event-driven architectures are designed to benefit from the sparsity of DNNs. A large diversity in the input frames likely produces significantly varying activation patterns in the network, endangering the stability of event volume and computational loads for per-frame executions. However, Tab. 2 demonstrates that the standard deviation of the recorded activation density and MAC density across the validation frames is under 2% and 1% respectively, which are surprisingly too low to produce any instabilities in inference time. Besides, this stability is not affected by datasets, networks, or activation suppression.

Table 3. The speedups and energy-savings of DNN inference at batch size 1 on GrAI-VIP related to the activation suppression.

MODEL	ACTIVATION SUPPRESSION	LATENCY REDUCTION	ENERGY REDUCTION
RES50-STAR	2.50X	2.78X	2.09X
MNV2-STAR	1.92X	1.28X	1.35X
YOLOX-STAR	2.24X	2.12X	2.08X

4.2.5 Comparison to SOTA

In Fig. 5, STAR (red) achieves an average of 80% in total activation sparsity for a heavy-weight model and 65% for a light-weight model with a small impact on accuracy ($\leq$ 1%). The comparison of L_1 and STAR shows that our approach results in a significant increase in activation suppression (sup) up to an average number of 35% relative to an optimized version of the SOTA approach [17] on a wide range of sophisticated networks. Notice that the above improvement is under the condition that the accuracy loss relative to the Tensorflow baseline is $\leq$ 0.5%. With additional tolerance (tol) on performance loss (i.e, 1 $\sim$ 2%), the improvement can be further enlarged to an average of 45%, which can be transferred to roughly 1.8× more latency and energy savings in event-driven processing.

4.3. Results on event-driven processor

4.3.1 End-to-end inference performance

Let us describe how well the explored activation sparsity translates to actual latency, FPS, and energy improvement in end-to-end inference on an event-driven hardware GrAI-VIP. Tab. 3 presents that our suppression method can lead to an average of 2.06× latency reduction and 1.84× energy savings in end-to-end inference on top of ReLU spar-

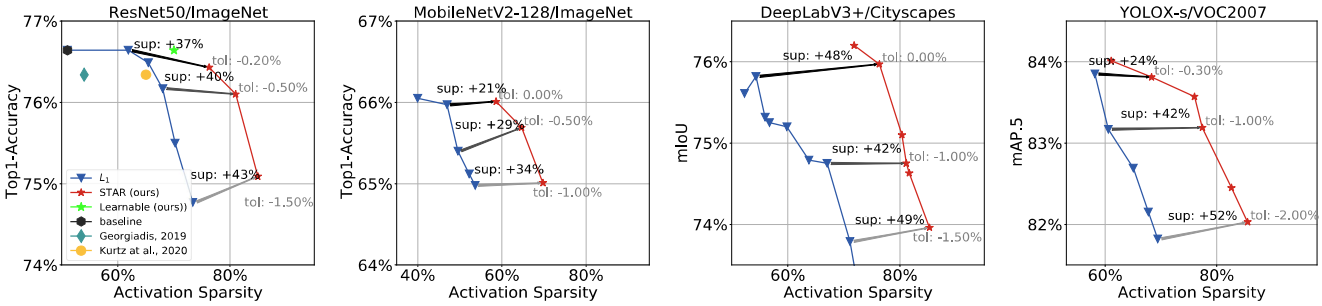

Figure 5. Comparison on various networks, on validation accuracy v.s. activation sparsity with the state-of-the-art (SOTA) under different loss tolerances (**tol**). Label **up** represents a relative improvement of activation suppression between SOTA and STAR.

sity across various networks. Most importantly, both reduction factors are roughly proportional to the suppression factor, confirming that STAR can consistently improve the latency and power efficiency of DNNs inference in event-driven processing.

4.3.2 Comparison with edge GPU device

As shown in Fig. 6, we utilize Jetson's performance as a benchmark and find that GrAI-VIP exhibits slightly better performance than Jetson Nano in most of the standard trained networks when operating at a lower frequency of 650 MHz. However, with the help of STAR, GrAI-VIP showcases a significant FPS boost of 2.86× on average over Nano by leveraging activation sparsity in our sparsified models. Notably, the application of STAR improves the FPS of ResNet50 from lower than Nano to twice that of Nano while simultaneously reducing its energy consumption by half. These findings highlight the effectiveness of STAR and its critical role in boosting the performance of event-driven processors.

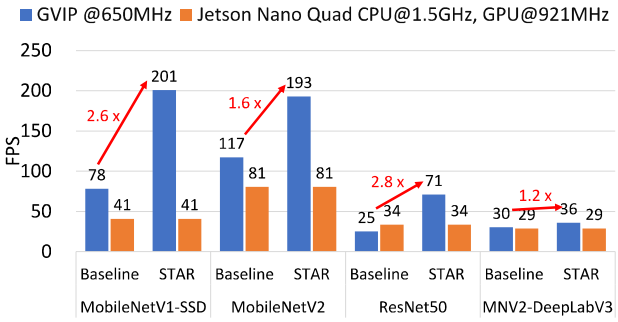

Figure 6. FPS comparisons between Jetson Nano and GrAI-VIP

5. Related Work

Modern network architectures [19, 36] widely apply ReLU as the activation function, inducing $\sim 50\%$ inherent activation sparsity [35]. This sparsity can be leveraged by either sparse matrix-dense matrix multiplication (SpMM) on the conventional hardware [22, 24, 37] or data-flow combined event-driven convolution on those newly emerged neuromorphic systems [9, 29]. Especially for the later ones, the inference latency and energy improve considerably as sparsity increases.

Therefore, to further explore activation sparsity, earlier works [2, 34] eliminate low-magnitude activations via static thresholds in inference. Nevertheless, the sparsity gains are limited due to accuracy concerns. Recently, regularization methods have been applied to network training [17, 24, 30] to increase sparsity in the activation maps. Interestingly, the difference between these regularizers tends to be modest in activation suppression. Some researchers [38] even claim that it is challenging to attain over 70% activation sparsity while maintaining accuracy loss within 2%. Instead of exploring new regularizers, [48] develops an adaptive training schedule to efficiently alter the regularization coefficient for the optimal. Their sparsity improvement is consistent across various applications but the increment remains small. In contrast, STAR smoothly combines regularization and thresholding in training and preserves the suppression effects of both techniques. The accumulated sparsity allows STAR to outperform all the existing methods.

6. Conclusions

In this paper, we present STAR, a novel activation suppression training method designed to fulfill the booming needs for sparsity exploration in event-driven processing. We discuss the limits of the existing approaches and propose STAR based on an in-depth analysis of activation behaviour in network learning. STAR combines partial-regularization with thresholding to maximally suppress non-zero activations in DNNs while maintaining accuracy. Consequently, STAR outperforms the SOTA on activation suppression by a significant margin of 35% with negligible accuracy loss, demonstrating its strong ability to induce activation sparsity. Moreover, the achieved activation sparsity can be effectively translated into latency reduction by up to $2.06\times$ and energy savings by up to $1.84\times$ on an event-driven processor. These results confirm the effectiveness of STAR in optimizing DNNs for efficient event-driven processing.

References

[1] Filipp Akopyan, Jun Sawada, Andrew S. Cassidy, Rodrigo Alvarez-Icaza, John V. Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael P. Beakes, Bernard Brezzo, Jente Benedict Kuang, Rajit Manohar, William P. Risk, Bryan L. Jackson, and Dharmendra S. Modha. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34:1537–1557, 2015. 1

[2] Jorge Albericio, Patrick Judd, Tayler Hetherington, Tor Aamodt, Natalie Enright Jerger, and Andreas Moshovos. Cnvlutin: Ineffectual-neuron-free deep neural network computing. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 1–13, 2016. 2, 3, 8

[3] Lennart Bamberg, Arash Pourtaherian, Luc Waeijen, Anupam Chahar, and Orlando Moreira. Synapse compression for event-based convolutional-neural-network accelerators. *CoRR*, abs/2112.07019, 2021. 14

[4] Y. Bengio. Estimating or propagating gradients through stochastic neurons. 05 2013. 5

[5] Simone Bianco, Rémi Cadène, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018. 1

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. ECCV*, 2018. 2, 5

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. IEEE CVPR*, 2016. 5

[8] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. 1

[9] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021. 1, 8

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: a large-scale hierarchical image database. In *Proc. IEEE CVPR*, pages 248–255, Jun. 2009. 5

[11] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *CoRR*, abs/1702.03118, 2017. 3

[12] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015. 5

[13] Jianqing Fan and Runze Li. Variable selection via penalized likelihood. 10 1999. 4, 11

[14] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 02 2019. 2

[15] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021. 2, 3, 5

[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 5

[17] Georgios Georgiadis. Accelerating convolutional neural networks via activation map compression. In *Proc. IEEE CVPR*, pages 7078–7088, Jun. 2019. 2, 4, 7, 8, 11

[18] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010. 1

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE CVPR*, pages 770–778, 2016. 2, 3, 5, 7, 8, 15

[20] Sebastian Höppner, Yexin Yan, Andreas Dixius, Stefan Scholze, Johannes Partzsch, Marco Stolba, Florian Kelber, Bernhard Vogginger, Felix Neumärker, Georg Ellguth, Stephan Hartmann, Stefan Schiefer, Thomas Hocker, Dennis Walter, Gengting Liu, Jim D. Garside, Stephen B. Furber, and Christian Mayr. The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing. *ArXiv*, abs/2103.08392, 2021. 1

[21] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019. 3

[22] Yu Ji, Ling Liang, Lei Deng, Youyang Zhang, Youhui Zhang, and Yuan Xie. Tetris: Tile-matching the tremendous irregular sparsity. pages 4119–4129, 2018. 8

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[24] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *Proc. ICML*, pages 5533–5543, Jul. 2020. 2, 3, 4, 5, 6, 8, 11, 13

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. Ssd: Single shot multibox detector. volume 9905, pages 21–37, 10 2016. 5

[26] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013. 3

[27] Alireza Makhzani and Brendan Frey. Winner-take-all autoencoders. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume*

2, NIPS'15, page 2791–2799, Cambridge, MA, USA, 2015. MIT Press. 2

[28] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *IEEE Trans. Biomed. Circuits Syst.*, pages 106–122, Aug. 2017. 1

[29] Orlando Moreira, Amirreza Yousefzadeh, Fabian Chersi, Gokturk Cinserin, Rik-Jan Zwartenkot, Ajay Kapoor, Peng Qiao, Peter Kievits, Mina A. Khoei, Louis Rouillard, Aimee Ferouge, Jonathan C. Tapson, and Ashoka Visweswara. Neuronflow: a neuromorphic processor architecture for live AI applications. In *Proc. DATE*, pages 840–845, 2020. 1, 8, 14

[30] Simon Narduzzi, Siavash A. Bigdeli, Shih-Chii Liu, and L. Andrea Dunbar. Optimizing the consumption of spiking neural networks with activity regularization. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–65, 2022. 2, 4, 8, 11

[31] Andrew Ng. Cs294a lecture notes. 2011. 4

[32] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen Keckler, and William Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH Computer Architecture News*, 45, May 2017. 3

[33] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2018. 3

[34] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. IEEE Press, 2016. 2, 8

[35] Minsoo Rhu, Mike O'Connor, Niladrish Chatterjee, Jeff Pool, Youngeun Kwon, and Stephen Keckler. Compressing dma engine: Leveraging activation sparsity for training deep neural networks. pages 78–91, 02 2018. 3, 8, 12

[36] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. 2, 3, 5, 8

[37] Gil Shomron and Uri C. Weiser. Spatial correlation and value prediction in convolutional neural networks. *IEEE Computer Architecture Letters*, 18:10–13, 2019. 8

[38] Yousefzadeh A Corradi F. Stuijt J, Sifalakis M. brain: An event-driven and fully synthesizable architecture for spiking neural networks. *Front Neurosci.*, pages 106–122, May. 2021. 8

[39] Thierry Tambe, En-Yu Yang, Zishen Wan, Yuntian Deng, Vijay Janapa Reddi, Alexander Rush, David Brooks, and Gu-Yeon Wei. Adaptivfloat: A floating-point based data type for resilient deep learning inference, 09 2019. 2

[40] Thierry Tambe, En-Yu Yang, Zishen Wan, Yuntian Deng, Vijay Janapa Reddi, Alexander Rush, David Brooks, and Gu-Yeon Wei. Adaptivfloat: A floating-point based data type for resilient deep learning inference, 09 2019. 12

[41] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. 3

[42] Mingxing Tan and Quoc V. Le. 04 2020. 3

[43] Anup Vanarse, Adam Osseiran, Alexander Rassau, and Peter van der Made. A hardware-deployable neuromorphic solution for encoding and classification of electronic nose data. *Sensors*, 19:4831, 11 2019. 1

[44] Sally Ward-Foxton. GrAI Matter research gives rise to AI processor for the edge, Jan. 2020. 5

[45] Dalin Zhang, Kaixuan Chen, Yan Zhao, B. Yang, Li-Ping Yao, and Christian S. Jensen. Design automation for fast, lightweight, and effective deep learning models: A survey. *ArXiv*, abs/2208.10498, 2022. 1

[46] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. 10 2017. 2

[47] Michael H. Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression, 2018. 12

[48] Zeqi Zhu, Arash Pourtaherian, Luc J.W. Waeijen, Egor Bondarau, and Orlando Moreira. Arts: An adaptive regularization training schedule for activation sparsity exploration. 2022. DSD2022: Euromicro Conference on Digital Systems Design 2022. 1, 2, 4, 8