A. Supplementary Methodology

A.1. Sparsity Regularizers

In a dataflow hardware architecture with an event-driven manner, illustrated in Fig. 7, a convolution is only initiated when there's an arrival event, which makes it efficient for exploiting activation sparsity. In the previous studies [13,17,24,30], various sparsity regularizers are proposed to reduce the activity of the network. Thus, we reproduce them in our experiments and evaluate their capacity of generating sparse activation maps compared to ours.

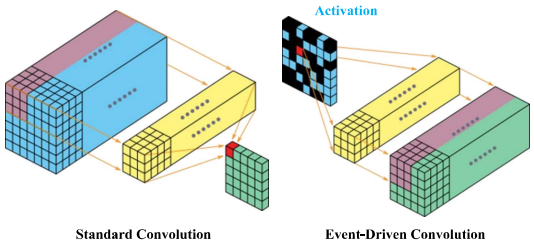

Figure 7. Event-Driven Convolution

A.1.1 L_2 regularization

The L_2 norm, also known as the Frobenius norm, is the square root of the sum of the squares of all the neuron values in the output activation maps. Although the gradients of this norm are effective for high values, they can become smaller as the values get closer to zero. Here's the formula:

$$L_2(x) = \sum_i \sqrt{{x_i}^2} \tag{7}$$

A.1.2 L_1 regularization

The L_1 norm, also referred to as the Manhattan Distance, is the sum of all neuron absolute values in the output activation maps. The gradients of this norm are uniform with respect to x_i, meaning that high and low values are penalized equally. This property makes L_1 norm a strong method to induce sparsity. Thus, we implement it in the first training phase of STAR for an optimal suppression result. It is defined as:

$$L_1(x) = \sum_i |x_i| \tag{8}$$

A.1.3 Hoyer regularization

The square Hoyer regularizer is an approximation of the L_0 norm which has the advantage of being scale invariant as for the L_0 norm and being differentiable almost-everywhere. It is defined as :

$$H(x) = \left(\sum_i |x_i|\right)^2 \left(\sum_i x_i^2\right) \tag{9}$$

A.1.4 SCAD

The SCAD (smoothly clipped absolute deviation) penalty attempts to alleviate the bias issue caused by L_1 and L_2 penalties on large regression values, while also retaining a continuous penalty that encourages sparsity. It is defined as:

$$SCAD(x_i) = \begin{cases} t|x_i| & if \quad x_i \leq t \\ \frac{2\alpha t|x_i| - {x_i}^2 - t^2}{2(\alpha - 1)} & if \quad t \leq |x_i| \leq \alpha t \\ t^2(\alpha + 1)/2 & otherwise \end{cases} \tag{10}$$

A.1.5 Partial-L_1 regularization

During the finetuning phase of STAR, the partial-L_1 technique is proposed as a better alternative to the L_1 regularizer. Unlike L_1 and other regularizers, partial-L_1 only penalizes activations that are close to zero, while allowing large-magnitude activations to grow and learn effective patterns. It's worth noting that the concept of partial-regularization is not limited to the L_1 regularizer, and its advantages can be applied to other types of regularizers as well.

$$partial\ L_1(x) = \sum_i |M(x_i, t) \odot x_i| \tag{11}$$

where

$$M(x_i, t) = \begin{cases} 1 & x_i \in (0, t) \\ 0 & otherwise \end{cases} \tag{12}$$

where

$$t = 2^n, n \in \{-4, -3, -2, ...\} \tag{13}$$

The regularization methods of partial-L_1, Hoyer, and SCAD all aim to prevent high-magnitude activations from shrinking to preserve accuracy. When dealing with large values of x_i where $|x_i| \geq$ a certain threshold, partial-L_1 removes the penalty, Hoyer further increases the activations to higher values, and SCAD applies a constant penalty. While these methods share some similarities, each has unique properties that make them suitable for different scenarios.

A.1.6 Loss function

To impose the activation sparsification, we define our loss function as:

$$L = J(w, x) + \sum_{l} \lambda_l R(x_l) \tag{14}$$

where J is the task loss for standard training, λ_l is the coefficient of sparsity regularization, x_l is the activations of layer l in the neural network, and $R \in \{L_1, L_2, partial - L_1, Hoyer, SCAD\}$ indicates the implemented regularizer.

A.2. Power-of-two Activation Thresholds

As the regularization coefficient λ is increased, the activation regularizer compresses more and more activations so that they are distributed around zero, as depicted in Fig. 8. The compactness of the power-of-two threshold for small values and looseness for large values enables it to perform fine-grained thresholding in the near-zero region.

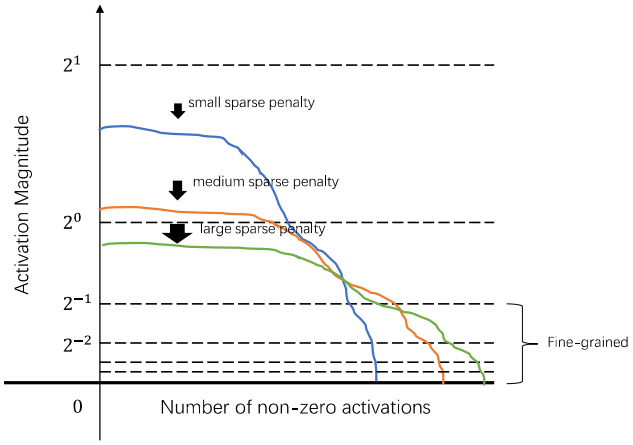

Figure 8. The number and magnitude of activations under different levels of regularization penalty.

A.3. Learnable Activation Thresholds

Naturally-learned activation sparsity varies greatly across network layers, revealing the sensitivity of layers in activation suppression [35]. Thus, performing a more complex threshold setting, i.e., layer- and channel-wise thresholding, should give better results in suppression. However, as mentioned in Sec. 3.3, the procedure of fine-grained threshold tuning is very costly. To reduce the computational cost, we propose to make the network learn its distinct thresholds t through an approximate threshold ReLU: $y = a_{out} \odot \sigma(\beta \odot (a_{out} - t))$, where t is a learnable threshold and $\sigma(.)$ is the sigmoid function. This activation function and its gradient are visualized in Fig. 9. It is differentiable everywhere in training with a fixed hyperparameter β. β is experimentally determined as 100, acting as a proxy to a hard threshold in the inference.

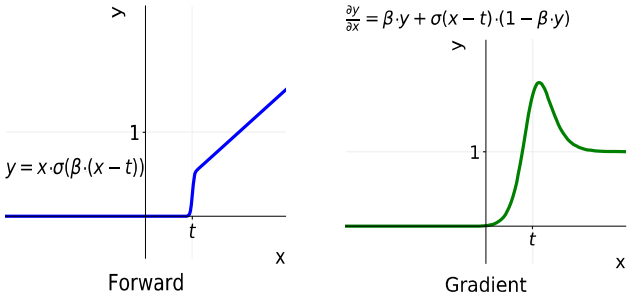

Figure 9. Learnable threshold activation and its gradient

A.4. Collaborative Optimization

To deploy the DNNS on hardware, especially for the edge platforms, weight pruning [47] and weight quantization [40] are highly sought-after since the on-chip memory is extremely constrained on embedded AI accelerators. Applying other optimizations with our activation suppression method (STAR) can achieve the accumulated optimization effect on the final model. The issue that arises when attempting to chain these techniques together is that applying one typically diminishes the gains of the preceding technique. For instance, performing activation suppression or quantization-aware training on a weight-sparsified model can deteriorate the achieved sparsity in model parameters. To solve this problem, we propose to apply several optimization methods simultaneously during the training, so those constraints would continuously affect the model in weight and feature learning.

B. Additional Experimental Results

B.1. Training

Our optimization experiments were conducted on the pre-trained models obtained through standard training. To implement STAR, we divided the training process into two phases: the first phase involved regularization-based training for 20 epochs with a significant learning rate for suppression, while the second phase involved fine-tuning under partial-regularization with thresholding until loss convergence. The hyperparameter settings for each model are presented in Tab. 8. Notably, the regularization coefficient remained constant for both *full*-L_1 and *partial*-L_1 methods, and we applied the thresholds to all activations except the outputs (e.g, softmax and sigmoid). Additionally, we eliminate other regularization techniques such as dropout and weight decay during the optimization training, as STAR has the same regularization effect.

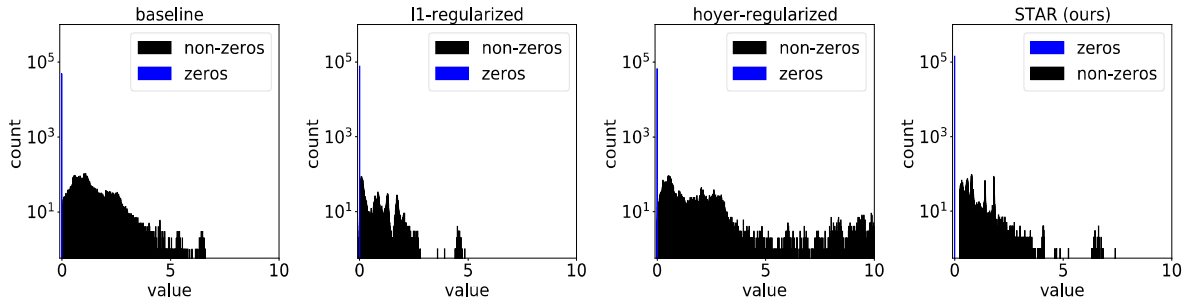

Figure 10. llustration of the output distribution at layer-ReLU 5 of ResNet/ImageNet with various regularizers. The Y axis is log-scale.

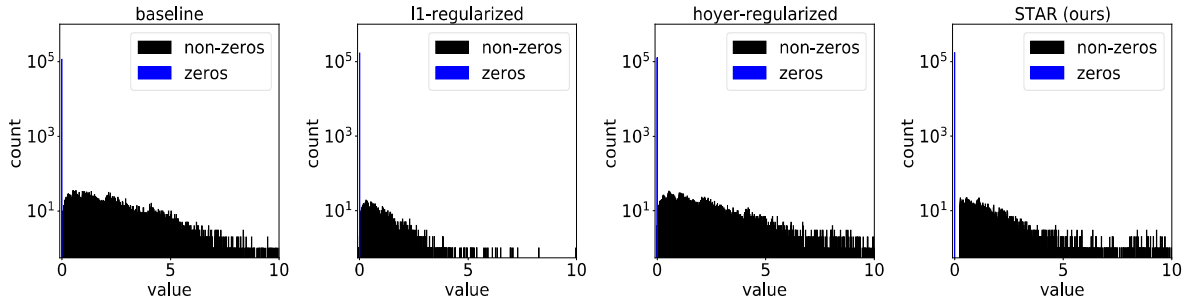

Figure 11. llustration of the output distribution at layer-ReLU 30 of ResNet/ImageNet with various regularizers. The Y axis is log-scale.

B.2. Impact on Activation Learning Behaviour

We conduct the suppression experiments with different activation regularization strategies, including L_1, *Hoyer* and our *partial*-L_1. As shown in Fig. 10 and Fig. 11, we compare the output activation distribution with those regularizers at the shallow activation layer (ReLU-5) and the deep activation layer (ReLU-30). We have the following observations: (1) The L_1-regularized model has more compression on the activation values, and induces more zeros than the baseline. (2) The *Hoyer*-regularized model largely spreads the distribution of activation values while providing a similar number of zeros as the L_1-regularized model. (3) Our *STAR*-sparsified model features the distribution of non-zero activations substantially similar to the baseline, which explains its superior accuracy recovery relative to other regularized models. On the other hand, the sub-figures *(STAR)* reveal that our partial-regularization maintains a large number of zero and near-zero events in the accuracy recovery stage, which are suppressed in the later thresholding to increase sparsity. This makes *STAR* significantly outperform the others in activation suppression.

It is worth noting that [24] claims the "diversity" of event distribution assists the *Hoyer*-regularized model to outperform the L_1-regularized model in accuracy recovery. However, *Hoyer*-regularizer forces the activation x to move towards 0 if $x < \frac{\sum x^2}{\sum |x|}$, otherwise x moves away from 0. The effect of pushing away those large activations from their original positions can potentially hinder the model accuracy by distorting the natural distribution of activations.

B.3. Layer-wise Learnable vs Model-wise Hard Thresholding

In Tab. 7, we demonstrate that layerwise learnable thresholding achieves slightly less suppression at the same accuracy level as the hard thresholding method. Further analysis of the layerwise activation density in Fig. 12 reveals that while most of the layers with learned thresholds have fewer activations than those with hard thresholds, the learned thresholds in the first few layers are explicitly lower than in the latter layers, resulting in a higher percentage of non-zero activations. Surprisingly, the first threshold is learned to be negative, enabling practically full activations to pass. However, this can lead to increased computation in the next-layer convolution and higher memory consumption for thresholds and activation maps. Therefore, for similar performance, model-wise hard thresholding is more friendly for hardware deployment.

Table 4. Results of collaborative optimization on ResNet50

STAR	MP	QAT	VAL ACC	ACTIVATION SUPPRESSION	SIZE REDUCT	MAC REDUCT
			76.64%	2.01×	1.00×	2.61×
✓			76.12%	5.26×	1.00×	4.52×
✓	✓		75.16%	5.09×	3.92×	9.37×
✓	✓	✓	75.22%	5.05×	15.31×	9.16×

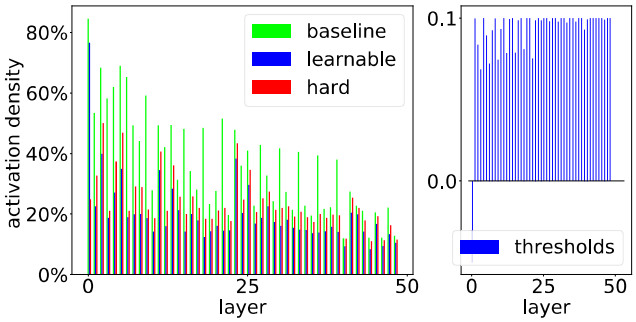

Figure 12. Layerwise comparisons of activation density and threshold value between hard thresholding and learnable thresholding.

B.4. Integration with Other Optimization Methods

GrAI-VIP is an event-driven processor designed for edge AI applications. Unlike the common Jetson Nano edge-GPU platform, which has a large DRAM memory, GrAI-VIP is equipped with only 36 MB of on-chip memory to ensure efficient data transfer. To fit ResNet50 and evaluate its performance on GrAI-VIP, we conducted collaborative optimization, as described in Appendix A.4. The results of ResNet50 on ImageNet at each optimization stage are shown in Tab. 4. Firstly, applying STAR resulted in $5.26\times$ activation suppression. Next, conducting magnitude-based pruning (MP) with STAR simultaneously removed 75% of the network parameters while maintaining activation sparsity. Finally, performing quantization-aware training (QAT) together with STAR and pruning led to a total reduction of $15.31\times$ in model size. Overall, the final optimized model achieved a $9.16\times$ reduction in MACs and a $15.31\times$ reduction in size with a 1.42% accuracy loss compared to the dense model. This makes ResNet50 able to fit in on-chip memory for both weights and activations. In addition, Tab. 5 shows that applying 80% weight pruning on ResNet50 results in a smaller variation in accuracy drop for STAR (from -0.52% to -0.68%) compared to L_1 regularizer (from -0.47% to -0.74%), indicating that STAR is more robust in collaborating with other optimizations compared to L_1 regularizer.

Table 5. Effect of pruning on activation suppression (ResNet50-ImageNet)

Method	Accuracy (%)	Activation Sparsity	Weight Sparsity
Baseline	76.64	51.06	0.00
L_1	76.17 (-0.47)	68.00	0.00
STAR	76.12 (-0.52)	81.00	0.00
Pruned model	75.84	51.00	79.04
L_1	75.10 (-0.74)	68.26	79.04
STAR	75.16 (-0.68)	80.17	79.04

Table 6. Resources and Features of GrAI-VIP

Resources/Features	GrAI-VIP
Process	TSMC 12FFC
Silicon Area	$7.6 \times 7.6\, mm^2$
Transistors	$4.5\, G$
Max # Neuron Cores	144
Max # Neuron	18 Millions
Max # Synapses	48 Millions
On-chip Memory	36 MB
Information Coding	Graded spike events (up to 16-bit payload)
Processing type	16-bits floating points
Synapses type	2/4/8/16-bits Floating points
Frequency	$650\, MHz$

C. Details of Event-Driven Processor GrAI-VIP

Here, we provide a brief overview of the event-driven processor GrAI-VIP, which is used in the experiments to generate the hardware performance of our sparsified models. GrAI-VIP is a commercially-available event-driven neural-network accelerator, based on the successor of Neuron-Flow [3, 29], from GrAI Matter Labs. As illustrated in Fig. 13, it is a 12-nm taped-out SoC with a 12×12 grid of SIMD-4 event-driven cores. Each core is equipped with $2Mbits$ on-chip memory for the storage of both weights and neuron states in an energy-efficient and performant manner. Besides, each event-driven core is equipped with a set of event-queues and vector units to boost performance and energy efficiency.

Additionally, Tab. 6 provides a comprehensive description of GrAI-VIP features and its hardware development kits (HDK) are shown in Fig. 14 and Fig. 15. The host board in Fig. 14, equipped with Intel Atom Quad Core, can support the operating system Linux and the deep learning framework Tensorflow to run DNNs.

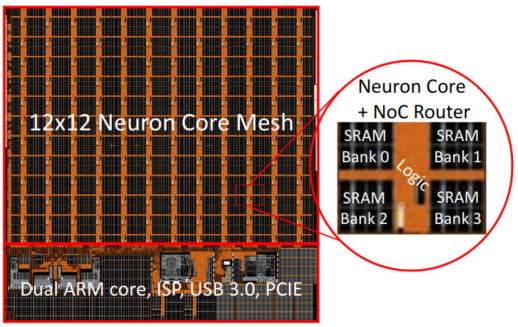

Figure 13. Block diagram of the event-driven neural-network accelerator (GrAI VIP). The zoom-in shows the high-level structure of a neuron core.

Table 7. Comparison between layer-wise learnable thresholds and model-wise hard thresholds with ResNet50 on ImageNet.

METHOD	VAL TOP-1 ACCURACY(%)	RELATIVE DROP(%) $100 \times \frac{O-B}{B}$	ACTIVATION DENSITY (%)	#REDUCTION $\frac{Act_B}{Act_O}$
RES50-S [19]	76.64	0.00	48.94	1.00X
W/O L_1, W/ HARD THRESHOLD	76.67	+0.04	**29.75**	**1.65X**
W/O L_1, W/ TRAINABLE THRESHOLD	76.64	-0.00	30.63	1.60X
W/ PARTIAL-L_1, W/ HARD THRESHOLD (**STAR**)	76.12	-0.68	**19.01**	**2.57X**
W/ PARTIAL-L_1, W/ TRAINABLE THRESHOLD	76.06	-0.76	22.26	2.20X

Table 8. L_1 coefficient and global threshold settings for various networks during STAR activation suppression

NETWORK	DATASET	METHOD	L_1 COEFFICIENT	THRESHOLD	ACCURACY (%)	ACTIVATION SPARSITY (%)
RESNET-18	CIFAR-100	BASELINE			78.27	59.67
		STAR	3E-7	2^{-2}	78.51	83.40
RESNET-50	IMAGENET	BASELINE			76.64	51.05
		STAR	3E-7	2^{-2}	76.12	80.99
MOBILENETV2	IMAGENET	BASELINE			72.03	39.38
		STAR	5E-7	2^{-3}	71.17	70.88
MOBILENETV1-SSD	KITTI	BASELINE			75.51	52.00
		STAR	1E-6	2^{-7}	75.27	81.45
YOLOX-RELU	VOC2007	BASELINE			84.21	50.12
		STAR	4E-7	2^{-5}	83.85	68.30
MOBILENETV2-DEEPLABV3	VOC2007	BASELINE			65.60	38.82
		STAR	2E-7	2^{-1}	65.08	63.33
RESNET50-DEEPLABV3+	CITYSCAPE	BASELINE			75.61	52.27
		STAR	1E-8	2^{-2}	75.97	76.31

Figure 14. Linux PC with GrAI VIP M.2 slot

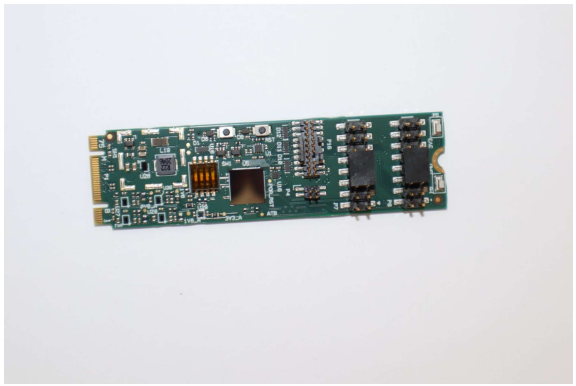

Figure 15. GVIP 80mm M.2 board