ARTS: an Adaptive Regularization Training Schedule for Activation Sparsity Exploration

Zeqi Zhu, Arash Pourtaherian, Luc Waeijen, Egor Bondarev and Orlando Moreira

November 14, 2022

ARTS: An adaptive regularization training schedule for activation sparsity exploration

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

Abstract*—Brain-inspired event-based processors have attracted considerable attention for edge deployment because of their ability to efficiently process Convolutional Neural Networks (CNNs) by exploiting sparsity. On such processors, one critical feature is that the speed and energy consumption of CNN inference are approximately proportional to the number of non-zero values in the activation maps. Thus, to achieve top performance, an efficient training algorithm is required to largely suppress the activations in CNNs. We propose a novel training method, called Adaptive-Regularization Training Schedule (ARTS), which dramatically decreases the non-zero activations in a model by adaptively altering the regularization coefficient through training. We evaluate our method across an extensive range of computer vision applications, including image classification, object recognition, depth estimation, and semantic segmentation. The results show that our technique can achieve 1.41× to 6.00× more activation suppression on top of ReLU activation across various networks and applications, and outperforms the state-of-the-art methods in terms of training time, activation suppression gains, and accuracy. A case study for a commercially-available event-based processor, Neuronflow, shows that the activation suppression achieved by ARTS effectively reduces CNN inference latency by up to 8.4× and energy consumption by up to 14.1×.

***Index Terms*—deep learning, activation sparsification, efficient training, computation efficiency, energy reduction**

I. Introduction

DEEP learning (DL) systems, dominantly implemented as Convolutional Neural Networks (CNNs), achieve state-of-the-art computer vision performance. CNNs are more accurate and easier to develop than hand-crafted techniques. Thereby, they enable a plethora of applications, opening new markets and opportunities at the edge as well as in the cloud. Novel CNN architectures [1]–[5]) are continuously proposed and utilized in a variety of AI applications to satisfy the ever-growing demands for higher model accuracy. However, the model size (i.e. parameter count) and the computational complexity of CNNs increase along with the accuracy. State-of-the-art CNNs require billions of multiply-accumulate (MAC) operations per inference, using millions of parameters. Frequent parameter accesses and intensive computations can lead to high energy consumption and long latency in inference. This tends to make high frame-rate CNN inference possible only on cloud engines, but not on small and resource-constrained edge devices. However, a considerable part of AI applications is aimed at edge deployment.

Model optimization methods [6]–[8] have been researched to accomplish energy- and latency-efficient CNN processing on the edge platforms. Taking GPU platforms as an example, many researchers have boosted inference efficiency by pruning network filters, channels, or layers [6], [7]. However, the computation reduction gained in a structured manner is not comparable to that achieved in those unstructured ones [9] [10]. Thus, novel computing architectures are proposed to leverage the irregular sparse patterns in deep neural networks. One of the innovative architecture paradigms for CNN inference is event-driven architectures with numerous proposed solutions from academia (e.g. DYNAPs [11]) and industry (e.g. TrueNorth [12], Loihi [13], GrAIcore [14]). This paradigm is inspired by the working mechanism of the human brain to reduce the computational needs of neural networks by extensively exploiting sparsity (i.e. the ratio of zero-valued activations). As shown in Fig. 1, those architectures execute a standard convolution in an event-based manner, which broadcasts source events (activations) via a transposed kernel. If one event is skipped, all its related computations and memory accesses will not be performed. Thus, with a portion of skipped events, the entire compute and memory-access requirements go down proportionally. This demonstrates the need to develop an optimization approach to induce huge quantities of zero in the activation maps for efficient CNN edge processing.

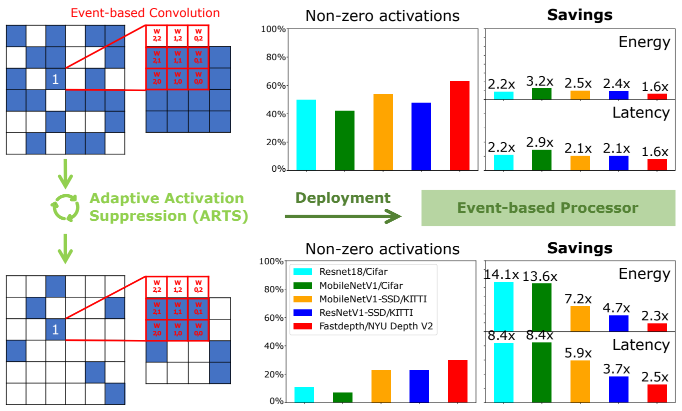

Fig. 1: Our proposed method (ARTS) progressively enforces the number of zeros in the activation maps, leveraged by event-based processors to achieve significant inference energy savings and latency reduction (The blue blocks are non-zero activations, while the white blocks are zero activations).

Most of the state-of-the-art neural networks contain some degree of activation sparsity by implementing the Rectified Linear Unit (ReLU) as an activation function. Nevertheless, this activation sparsity can be actively re-enforced during training to achieve much higher ratios. In this paper, we propose a training methodology that increases the ratio of zeros in the activation maps (called activation sparsity) to dramatically boost the capability of an event-based architecture and to deliver high-accuracy CNN inference at the edge with low latency and low power consumption.

The training process of suppressing non-zero values in neural network activation maps, also referred to as *activation sparsification*, has been studied in prior research. The state-of-the-art techniques mostly incorporate a regularization penalty for activation maps into the loss function [15], [16]. The regularization term is multiplied by a coefficient, allowing for adjustment of the trade-off between activation sparsity and accuracy. Typically, a so-called *one-shot approach* is used, in which a constant coefficient is implemented for each sparsification training, and several training experiments are executed to find the optimal coefficient. This kind of one-shot approach leads to two deficiencies in the optimization: first, aggressively reducing activations with an excessively large regularization coefficient can result in irrecoverable accuracy loss and optimization instability; second, searching for an appropriate coefficient value is computationally expensive since this value varies significantly depending on network capacity, dataset complexity, and application properties. Thus, a very missing part is how to efficiently determine an optimal coefficient, which gives maximized activation sparsity under certain accuracy loss tolerance.

To overcome these limitations, we propose an efficient training schedule called *Adaptive-Regularization Training Schedule* (ARTS) to gradually increase the ratio of zero values in activation maps. The final sparsified model is achieved in a convenient way by defining a tolerable accuracy loss; enabling a quick model adjustment to the application's requirements in terms of power, performance, and accuracy.

Additionally, activation sparsity is the most easily exploited on an event-based processor, but it is not the only way to reduce computations in a neural network. As the predominant computations in neural networks are computing inner products of activations and weights, making either weights or activations zero can result in a computation reduction. Furthermore, an edge processor's on-chip memory is extremely limited; unstructured pruning methods [9], [10] can help to deeply compress the weight size and allocate it near the processing units. Therefore, we propose an approach of co-integrating unstructured weight pruning with ARTS—emphasizing their mutual compatibility in reducing model redundancy and showing the massive compute-requirement optimization induced by joint sparsification.

In this paper, we make three main contributions:

1) We propose an efficient yet effective adaptive regularization training schedule to boost the sparsity in activation maps, which gradually suppresses non-zero activations while retaining the model.
2) We evaluate our method through 7 CNN networks on 5 different datasets under 4 common computer vision applications. The activation-sparsified models are tested on an industrial event-based architecture, NeuronFlow [14]. We analyze the trade-offs between activation sparsity and latency/power reduction, and we show the actual layerwise performance gains on real-world silicon.
3) We study the compatibility of ARTS with another prominent sparsification approach (unstructured weight pruning) for even higher sparsity ratios in MAC.

The remainder of this paper is structured as follows. Sec. II discusses activation sparsity exploitation in computer architecture. Sec. III gives an overview of the related work on neural network sparsification. Sec. IV describes ARTS in detail. Sec. V presents the experimental setup and results. Sec. VI concludes the paper.

II. Sparsity Exploitation in Compute Platforms

An inherent challenge of exploiting activation sparsity on conventional parallel hardware is the irregularity of sparse patterns and the dynamic locations of non-zero elements at run-time, which leads to complex dynamic control flow. We find that the techniques proposed in earlier studies [15]–[19] targeting these traditional architectures introduce overheads in the form of expensive data transformations (e.g. sparse-to-dense conversion) to benefit from sparsity. Moreover, they require full layer computation, prohibiting the application of advanced fusion techniques [20], hampering latency and increasing memory requirements.

On the other hand, event-driven architectures are inherently capable of exploiting sparsity [13], [14], [21]. In contrast to GPUs, for example, which are statically scheduled and always perform a fixed amount of work independent of the data, event-driven architectures dynamically adapt based on (input) data. In such architectures, events are used to trigger compute, rather than a fixed instruction schedule. Dynamic control flow is a core part of the architectural concept, aligning perfectly with the idea of sparse computation. In the context of sparse neural-network evaluation, a natural fit is to map the output of one or several neurons to an event. If, and only if, a neuron produces an output of significant value, an event is created which triggers updates to the neurons in the next layer(s). In this way, costly computations and memory accesses surrounding insignificant neuron outputs are avoided.

At a high level, the leading neural-network accelerators roughly follow a similar template [13], [14], [21]. They all consist of a grid of data-driven cores that can communicate by means of some interconnect fabric. These data-driven cores are idle, and only become active upon the arrival of events that trigger neuron updates. Depending on the results of these updates, new events may or may not be emitted by the cores.

Although the benefits from ARTS are widely applicable, in this work we focus on sparsity exploitation on event-driven architectures. In particular, all our results are evaluated on an experimental machine based on the successor of NeuronFlow [14], GrAI-VIP, a commercially-available event-driven neural-network accelerator by GrAI Matter Labs. As illustrated in Fig. 2, the experimental machine consists of a 12×12 grid

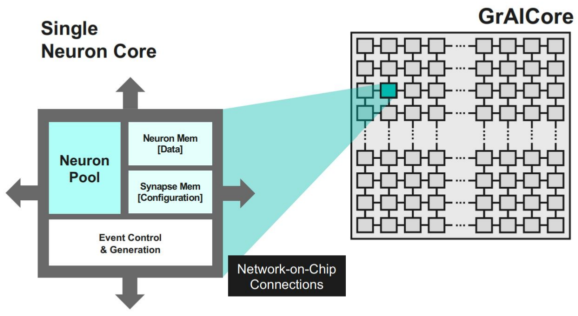
Single Neuron Core
GrAICore
Neuron Pool
Neuron Mem [Data]
Synapse Mem [Configuration]
Event Control & Generation
Network-on-Chip Connections

Fig. 2: Block diagram of the event-driven neural-network accelerator (GrAI VIP). The zoom-in shows the high-level structure of a processing unit.

of event-driven cores. Each core is equipped with 2 Mbit on-chip memory for storing both weights and neuron states in an energy-efficient and performant manner. Furthermore, each event-driven core is equipped with a set of event-queues and vector units to boost performance and energy efficiency. More detailed analyses on this machine are provided in Sec. V-B.

III. Overview of Sparsification Techniques

A. Activation sparsification

Many state-of-the-art network architectures [1], [2] adopt the ReLU activation function, which clamps all negative activation values to zero. It can provide about $50\% \sim 70\%$ inherent activation sparsity in the network. Cnvlutin [22] eliminates those small magnitude activations with a static threshold mask to further enhance the activation sparsity. Yang et al. [23] improve the activation sparsity even further by employing a dynamic dropout approach at run-time. Georgiadis [15] achieves considerable activation sparsity by retraining the network with an L1 regularization of activation in the loss function. However, the method performs a grid-search of the regularization coefficient, which is computationally costly. Kurtz et al. [16] extend the concept by substituting Hoyer regularization for L1 regularization. It increases activation sparsity on classification tasks with faster loss convergence, while still defining the coefficient through grid-search.

Both state-of-the-art techniques [15], [16] use a constant regularization coefficient through the whole finetuning, known as the one-shot approach. They have mainly focused on the study of sparse regularizers while overlooking the importance of an efficient sparsification training schedule. In contrast, our work introduces an adaptive coefficient scheduler, which enforces more zero values in the activation maps by continuously rewinding the learning rate and raising the regularization coefficient during optimization. Our experimental results show that this kind of gradual approach can yield superior results to existing constant-coefficient approaches.

B. Joint weight and activation sparsification

There are very few studies that examine the integration of weight pruning and activation sparsification. Han et al. [18] find that weight pruning increases the ratio of non-zero activations in the feature maps, not necessarily improving the overall compute sparsity. Yang et al. [23] address this by applying a dynamic pruning mask on activation per inference to minimize the pruning blowback and further boost activation sparsity. Similar to the above work, our approach puts sparse constraints on both weights and activations to avoid the negative impact on the sparsity caused by each method. The difference between our method and Yang et al. [23] is that we do not introduce extra operations (dynamic activation mask) during inference, the lower computational complexity is achieved only through training.

IV. Proposed Training Schedule

A. Fundamental Theory

The neural network learning process is commonly addressed as the optimization of a loss function, defined as:

$$\min_{w \in R^d} \left\{ \frac{1}{N} \sum_{n=1}^{N} f_n(w) + \lambda_w r(w) \right\}, \tag{1}$$

where N is the mini-batch size, n is the index of the training samples, and $w \in \mathbb{R}^d$ indicates model parameters. The term $f_n(w)$ represents the task loss (usually the cross-entropy loss for classification and the l1-norm for regression), and the other term $r(w)$ is a regularization loss implemented on model parameters, typically protecting the model from overfitting.

To impose the activation sparsification, we introduce a sparse regularizer of activation in Eq. (1), simplfied as:

$$\min_{w \in R^d, D \in R^s} \left\{ L(w) + \lambda R_s(x) \right\}, \tag{2}$$

Specifically, the sum loss is composed of two terms: task loss $L(w)$ and sparse penalty $R_s(x)$. The coefficient λ balances the weight of the sparse penalty in the sum loss, and its value largely affects the activation sparsity gains in the final model.

In the training phase, model parameters in the l^{th} layer are updated via the gradients g_l, backpropagated from the loss:

$$\begin{aligned} g_l &= \frac{\partial L(w)}{\partial w_l} + \lambda_l \frac{\partial r(x_{l,n})}{\partial w_l} \\ &= \left[\frac{\partial L(w)}{\partial x_{l+1,n}} \frac{\partial x_{l+1,n}}{\partial x_{l,n}} + \lambda_l \frac{\partial r(x_{l,n})}{\partial x_{l,n}} \right] \frac{\partial x_{l,n}}{\partial w_l}, \end{aligned} \tag{3}$$

Eq. (3) shows that, for the parameters w_l in the l^{th} layer, the updates are determined by two forces: loss gradient from the task and regularization gradient from the sparse penalty. If the regularization coefficient λ_l goes higher, the penalty force will drive the model parameters to produce sparser representations and it will not stop unless the regularization gradient approximates to zero as its loss converges.

Assuming that the learning rate lr remains constant, the increment of weights is mostly driven by the magnitude of the coefficient λ. A strong coefficient λ can enlarge the update of weights, resulting in a gap increase between the pre-trained weights and the activation sparsified weights. The initialization of pre-trained weights becomes meaningless if the coefficient is too large. Thus, merely increasing the coefficient to enhance activation sparsity by a one-shot approach is not a viable strategy, especially for massive sparsity exploration. We're seeking for a cyclic asymptotic strategy that starts with a low regularization coefficient and subsequently increases it by a small amount to improve model sparsity. Such a gradual approach slightly changes model weights on top of the former loop to achieve a sparser representation, preventing the irreversible damage from the dramatic adjustment of weights through a one-shot approach and making the training more robust in an iterative manner.

In the following sections, we first present an updated version of the one-shot approach to activation sparsification, then propose our gradual approach called Adaptive Regularization Training Schedule (ARTS) evolved from this one-shot approach. Additionally, we introduce the sparse regularizer implemented for our study, and finally design a joint optimization method to check the compatibility of ARTS with other popular sparsification methods, such as weight pruning.

B. *One-shot training schedule*

Our proposed one-shot training schedule for activation sparsification is built on the common knowledge of sparse representations learning. As a classic training procedure of CNNs with ReLU, in the early training phase, the large learning rate drives the dramatic improvement in the loss function, combined with heavy drops in activation density; in the latter phases, the learning rate is reduced to stabilize the weights around the loss minima, while the activation density slightly increases [24]. Thus, we propose to sparsify the pretrained model by applying the learning rate schedule from the standard training, known as learning rate rewinding [25], [26]. The optimization training starts by suppressing the activations dramatically to the bottom; then, the lost performance is recovered through fine-tuning.

C. *Adaptive-regularization training schedule*

Defining the optimal coefficient for the one-shot approach requires domain experts to explore the search space intelligently using rule-based heuristics; otherwise, grid-search requires significant time and computation power resources. To balance the trade-off between performance and training time, we design an iterative regularization training scheme that starts with a small regularization coefficient and progressively increases it to reach the optimal. In particular, given a pretrained model w and an expectation E, we add a regularization penalty to the loss function to drive the activations to zero. The regularization coefficient is initialized at a small value and continuously incremented by a small δ: $\lambda_{i+1} = \lambda_i + \delta\lambda$, where i is the index of iteration loops, $\delta\lambda$ is the granularity in which we add up the penalty. The growing condition of λ is that the monitored metric value (e.g. accuracy) of the validation dataset surpasses an expectation value, which is the baseline metric with $\eta\%$ tolerance.

The iterative scheme consists of a bunch of one-shot approaches. The complete ARTS procedure is shown in Algorithm 1. In any training loop, the training will continue with

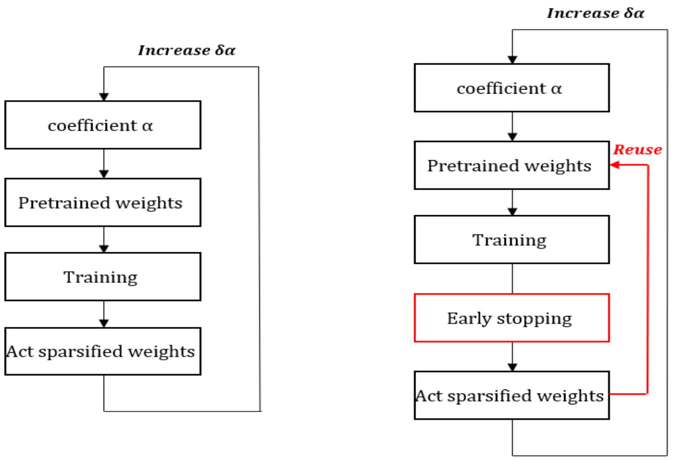

Fig. 3: Grid search (Left) and our method ARTS (Right).

an updated λ if the growing condition is satisfied (this makes each loop have a different time length); otherwise, the training will be terminated after an interval of finetuning. Our proposed gradual approach enables the final model performance to be extremely close to the expected value E due to the fine-grained coefficient adjustment.

The primary difference between our adaptive coefficient method and the grid-search on one-short approach is that in each loop, we reuse the pre-trained weights from the previous training loop and omit some superfluous finetunings, illustrated in Fig. 3. Applying a mild regularization coefficient increase results in a small update on weights between two consecutive training loops, thereby preventing the significant accuracy degradation from aggressive activation suppression. Additionally, our training schedule effectively brings down the entire optimization time by removing those redundant training epochs on two sides of each loop. For instance, in the n^{th} training loop, the sparse regularization term of an optimized model is smaller than the one of the original model. This makes the sparse regularization approach the minimal faster, especially for the final loops with a deep sparse model. Moreover, since the metric of validation can surpass the expectation, the long finetuning process of weight stabilization can be done once in the final loop.

In principle, the ARTS method solves the deficiencies in earlier one-shot approaches, such as the degradation in performance caused by aggressive activation suppression, and improve the training time for the entire optimization.

D. *Sparse regularizer on activation*

In our method, we simply select the hoyer regularizer as the regularization term $r(x_{l,n})$ for activation sparsification in Eq. (2). Hoyer et al. [27] first propose this regularizer as a normalized sparseness measure. It is defined as the ratio of the l1-norm and the l2-norm of a vector: $r(x_l) = \left(\sum_{i=1}^{d} |x_l|\right)^2 / \left(\sum_{i=1}^{d} x_l^2\right)$. Recently, it has been applied for weight and activation sparsification [16], [28] because of its desirable properties for model sparsification training: scale-

Algorithm 1: Adaptive Regularization Training for Activation Sparsification

```
1  INPUT: Pre-trained model $\mathbf{w}$, regularization factor for $l-th$ layer $\lambda_l$, regularization loss $R_t$
2  INPUT: Initial learning rate $lr$, decay $\gamma$, patience interval $K_p$, recovery interval $K_r$, granularity $\delta\lambda$.
3  INPUT: Performance tolerance $\eta\%$, expectation $E$.
4  INIT: Set uniform coefficient $\lambda_l$ for all the layers.
5  INIT: Iteration $i = 0$, regularization stablization steps $sr_i = 0$, weight stablization steps $sw_i = 0$.
6  TRAIN:
7    While ($Acc \geq E$ or $sw_i < K_r$) do
8      If $Acc \geq E$ then
9        Increase the regularization: $\lambda_{l,i+1} \leftarrow \lambda_{l,i} + \delta\lambda$
10       Update the loop index: $i \leftarrow i + 1$
11       Rewind the learning rate: $lr_i \leftarrow lr$
12       Reset the regularizer stablization step: $sr_i \leftarrow 0$
13       Reset the weight stablization step: $sw_i \leftarrow 0$
14     Else do
15       If ($R_{t,i}$ converges and $sr_i \geq K_p$) then
16         Reduce the learning rate: $lr_i \leftarrow lr \times \gamma$
17         Count the stablization step: $sw_i \leftarrow sw_i + 1$
18       Else do
19         Count the stablization step: $sr_i \leftarrow sr_i + 1$
20       End if
21     End if
22     Weight update by gradient descent
23   End While
24 FINETUNE the final iteration weights $w_i$ to recover accuracy ($\sim$ expectation $E$)
25 OUTPUT: Sparsified model $w_i$.
```

invariance and differentiability almost-everywhere. An additional advantage of this regularizer is that it can turn weights of small value to zero, whereas protecting large weights through gradient descent. Such a property can efficiently recover sparse solutions from coherent and redundant representations.

E. Joint optimization for weight and activation sparsity

The fundamental aspect of ARTS is that it can be seamlessly integrated with other compression/acceleration algorithms due to its adaptive coefficient settings during training. Exemplary, we focus on weight pruning in this study, since it enables model compression and sparse computation for efficient inference on the event-based architectures. Two sequential procedures are proposed to compare for the optimal sparsification: one starts with ARTS for activation sparsification, followed by an iterative weight pruning approach; the other works *vice versa*. We quantify the joint optimization performance in terms of MAC sparsity (i.e. the fraction of zeros in MAC), as the event-based processors can skip execution here.

V. Experiments

In this section, we evaluate our proposed training scheduling technique by applying it to a wide range of models. These experiments have three main purposes: (1) examine ARTS on various CNN models/applications and compare the sparsification results with the state-of-the-art methods, (2) demonstrate the energy/latency gains of activation sparsity on a real event-based silicon, *GrAIcore* [29], and (3) manifest the massive MAC sparsity gains in a joint optimization scope.

TABLE I: Comparisons between Grid-Search and our method (ARTS) on overall training time. Coeff. indicates the search space of the regularization coefficient.

Model	Method	Coeff.	Epochs	Speed-up
ResNet18 Cifar-10	Grid Search	1.0e-8 $\sim$ 1.0e-6	5400	
	ARTS	1.0e-8 $\sim$ 7.1e-7	850	**$\sim$3$\times$**
ResNet-SSD KITTI	Grid Search	1.0e-7 $\sim$ 1.0e-5	6000	
	ARTS	1.0e-7 $\sim$ 1.8e-5	1900	**$\sim$3$\times$**
Fastdepth NYU	Grid Search	1.0e-8 $\sim$ 1.0e-7	1000	
	ARTS	1.0e-8 $\sim$ 6.0e-8	80	**$\sim$12$\times$**
SkipNet Cityscapes	Grid Search	1.0e-8 $\sim$ 5.0e-8	500	
	ARTS	1.0e-8 $\sim$ 4.0e-8	150	**$\sim$3$\times$**

We pick several state-of-the-art CNN architectures and carry out the experiments on well-known datasets: ResNet [1] and MobileNet [2] for image classification on Cifar-10/Cifar-100 [30], SSD [4] for object detection on KITTI-2D [31], Fastdepth [3] for depth estimation on NYU Depth v2 [32], and SkipNet [5] for semantic segmentation on Cityscapes [33]. These experiments include a wide range of CNNs, with varying complexity. Networks like MobileNet, SSD, Fastdepth and SkipNet have already been designed with low computing complexity in mind, posing a barrier for gaining easy savings.

Experimental setup: Our gradual sparsification method ARTS is implemented in TensorFlow. The baseline models are achieved through training from scratch. The pre-trained models are used as the starting points for the sparsification experiments. We experiment on a commercial-grade high-level simulator written for *GrAIcore* [29]. The simulator is event-accurate and can report the inference latency and energy/power consumption.

A. Activation sparsity exploration

In this subsection, we quantify the activation sparsity achieved through ARTS. Therefore, we study the average activation sparsity for three cases: 1) the baseline models, 2) the constant regularized models (one-shot approach), 3) the ARTS-sparsified models (gradual approach).

a) Effects of ARTS on training time: we use the identical hyperparameter settings for grid-search and ARTS to provide a fair comparison, including the initial coefficient and learning-rate schedule. The search space of ARTS does not need to be defined manually; a specified loss tolerance will suffice. The search space of grid-search is roughly defined based on the coefficient range achieved by ARTS, with a coefficient increment $\lfloor \log_{10} \lambda \rfloor$. Tab. I shows that ARTS reduces the entire training time by at least 3$\times$ versus grid-search, and its ending coefficient is relatively fine-grained compared to grid-search.

b) Effects of ARTS on activation sparsity: The ARTS experimental results are displayed in Tab. III. One salient trend is that those more redundant models allow for greater exploration of zeros in activations. For instance, by implementing ARTS, the ResNet-based models achieve lower non-zero activations than the MobileNet-based ones. Nonetheless, we also note that ARTS can still largely boost the activation sparsity for those light-weight models by up to 7.14$\times$ on *Cifar-10* and 3.85$\times$ on *KITTI*, bringing additional efficiency improvements

TABLE II: Activation sparsification through our method ARTS on various models and applications, compared with the state-of-the-art constant coefficient approaches.

Application	Dataset	Model	Method	Acc.↑	Act.↓
Image Classification	Cifar-10	MobileNetV1	L1	+0.21 %	30 %
			Hoyer	−0.38 %	32 %
			Ours	+0.01 %	**36 %**
		ResNet18	L1	−1.10 %	22 %
			Hoyer	+0.03 %	30 %
			Ours	+0.01 %	**33 %**
	Cifar-100	ResNet18	L1	+0.00 %	16 %
			Hoyer	+0.26 %	17 %
			Ours	+0.01 %	**25 %**
	ImageNet	ResNet50	L1	−0.30 %	1 %
			Hoyer	−0.30 %	8 %
			Ours	−0.40 %	**14 %**
Object Detection	KITTI-2D	MobileNet-SSD	L1	−0.40 %	11 %
			Hoyer	−0.16 %	6 %
			Ours	+0.41 %	**28 %**
		ResNet-SSD	L1	+0.14 %	24 %
			Hoyer	−0.33 %	11 %
			Ours	+1.05 %	**24 %**
Depth Estimation	NYU Depth v2	Fastdepth	L1	+0.46 %	30 %
			Hoyer	−0.28 %	33 %
			Ours	+0.14 %	**33 %**
Semantic Segmentation	Cityscapes	SkipNet+	L1	+1.02 %	21 %
			Hoyer	−0.19 %	21 %
			Ours	−0.31%	**24 %**

on the edge deployment. The second noticeable trend is that dataset complexities also affect the final activation sparsity gains. The same network *ResNet18* explores more zeros on a simple dataset *Cifar-10* than on a complex one *Cifar-100*. A third observation is that more challenging computer vision tasks, such as object detection, depth estimation, and semantic segmentation, require much richer activation representations than the classification tasks to retain high performance. For instance, segmentation networks always have the shape of a bottleneck. To extract the features from inputs and reconstruct them at outputs, the network necessitates the transfer of vast amounts of information from left to right, which makes the middle layers extremely crowded (see Fig. 4). In general, our method is especially effective in the context of redundant models, where activation sparsity can be thoroughly explored, without accuracy loss.

Additionally, Tab. III shows that our method can provide more activation sparsity in the final optimized model by sacrificing approximately 1% of its baseline accuracy. This shows that ARTS provides a more automated workflow on activation sparsification than the constant coefficient approaches by pre-defining a loss tolerance. Users can easily trade-off accuracy for increased sparsity under various application scenarios.

c) Comparisons with state-of-the-art approaches: Tab. II presents the comparison between our approach and the state-of-the-art methods. In general, our method outperforms the two SOTA approaches by boosting the accuracy on average by 0.44% and suppressing more non-zero activations on average by 8% across all the experiments. We infer that the superior results of our method can be related to two factors: first, our approach continuously increases the coefficient by a small value throughout the training, allowing for a fine-grained final coefficient for sparsification; secondly, progressively growing the coefficient can achieve more zeros in the activation maps than by a one-shot approach while maintaining accuracy. Another observation from Tab. II is that the constant Hoyer regularizer does not always outperform the l1-norm one in various scenarios. Considering our training schedule can serve different regularizers, we can easily replace the default regularizer for a better optimized model.

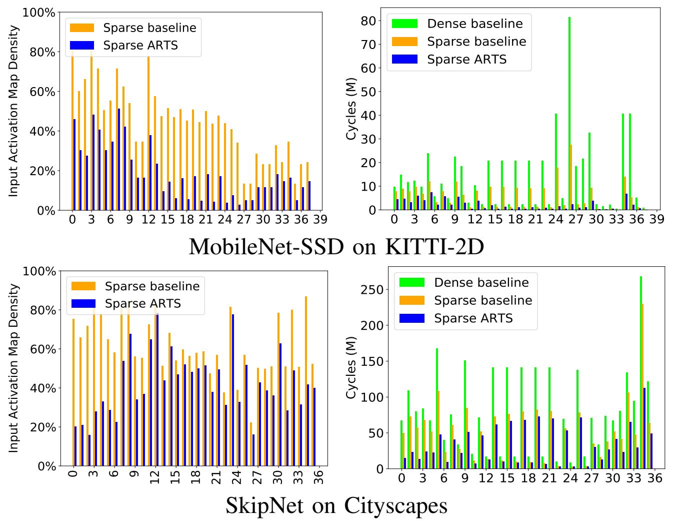

Fig. 4: Layerwise analysis of activation density (Left) and executed cycles on the event-based processor (Right)

B. Evaluation on event-based processors

a) Layer-wise Analysis: We run the models with half-precision float (FP16) on an event-based simulator, written for a 12-nm taped-out chip with 144 SIMD-4 cores running at 800MHz. Fig. 4 shows that ARTS can consistently boost the activation sparsity across all the layers, significantly beyond the sparse baseline. Additionally, the explored zeros in the activation maps can be effectively transferred to the reduction of cycle-counts for layers executed on the event-based processor (even for depthwise layers, which are not efficient on most of the hardware). This illustrates that the event-based processor can benefit substantially from sparsification. As a result, the reduced cycle-counts result in a significant improvement in achievable frame-throughput (FPS), latency, and energy consumption. Another noticeable finding is that ARTS exceedingly optimizes the computationally-heavy layers in the network, largely improving the throughput bottlenecks for pipelines with massive-multicore event-based processing.

b) End-to-end inference performance: The event-based simulation results on different levels of activation sparsification are shown in Fig. 5. Frame latency and energy consumption reduce as the non-zero activations declines in the sparsified models, and both latency and energy consumption are approximately linear to the percentage of non-zero activations. Additionally, we notice that the slopes of the energy-activation and latency-activation curves vary amongst models. This is

TABLE III: Results of activation sparsification through ReLU and ARTS. Acts. indicates the percentage of non-zero activations. Acc. denotes $Top1Acc.$, $mAP.5$, $delta1$, and $mIoU$ for image classification, object detection, depth estimation, and semantic segmentation, respectively. Speed-up 1 is the non-zero activation ratio of the ReLU sparsified baseline and the ARTS sparsified model. Speed-up 2 is the ratio of the ARTS sparsified model's activation to that of the dense baseline.

Application	**Dataset**	**Model**	**Variant**	**Acc.**	**Act.**	**Speed-up 1**	**Speed-up 2**
Image Classification	Cifar-10 [30]	MobileNetV1 [2]	Baseline	93.59 %	50 %	1.00 ×	2.00 ×
			ARTS tol-0 %	93.60 % (+0.01 %)	14 %	3.57 ×	7.14 ×
			ARTS tol-1 %	92.80 % (–0.79 %)	11 %	4.55 ×	9.10 ×
		ResNet18 [1]	Baseline	94.00 %	42 %	1.00 ×	2.38 ×
			ARTS tol-0 %	94.01 % (+0.01 %)	9 %	4.67 ×	11.11 ×
			ARTS tol-1 %	93.04 % (–0.96 %)	7 %	6.00 ×	14.29 ×
	Cifar-100 [30]	ResNet18 [1]	Baseline	76.14 %	43 %	1.00 ×	2.33 ×
			ARTS tol-0 %	76.15 % (+0.01 %)	18 %	2.37 ×	5.56 ×
			ARTS tol-1 %	75.33 % (–0.81 %)	14 %	3.06 ×	7.14 ×
Object Detection	KITTI-2D [31]	MobileNet-SSD [4]	Baseline	75.80 %	54 %	1.00 ×	1.85 ×
			ARTS tol-0 %	76.21 % (+0.41 %)	26 %	2.08 ×	3.85 ×
			ARTS tol-1 %	75.15 % (–0.65 %)	23 %	2.35 ×	4.35 ×
		ResNet-SSD [4]	Baseline	75.73 %	48 %	1.00 ×	2.08 ×
			ARTS tol-0 %	76.78 % (+1.05 %)	24 %	2.00 ×	4.17 ×
			ARTS tol-1 %	75.25 % (–0.48 %)	23 %	2.09 ×	4.35 ×
Depth Estimation	NYU Depth v2 [32]	Fastdepth [3]	Baseline	77.28 %	63 %	1.00 ×	1.59 ×
			ARTS tol-0 %	77.42 % (+0.14 %)	30 %	2.10 ×	3.33 ×
			ARTS tol-1 %	76.93 % (–0.35 %)	30 %	2.10 ×	3.33 ×
Semantic Segmentation	Cityscapes [33]	SkipNet [5]	Baseline	61.47 %	62 %	1.00 ×	1.62 ×
			ARTS tol-0 %	61.76 % (+0.26 %)	38 %	1.64 ×	2.65 ×
			ARTS tol-1 %	59.90 % (–1.57 %)	34 %	1.81 ×	2.93 ×

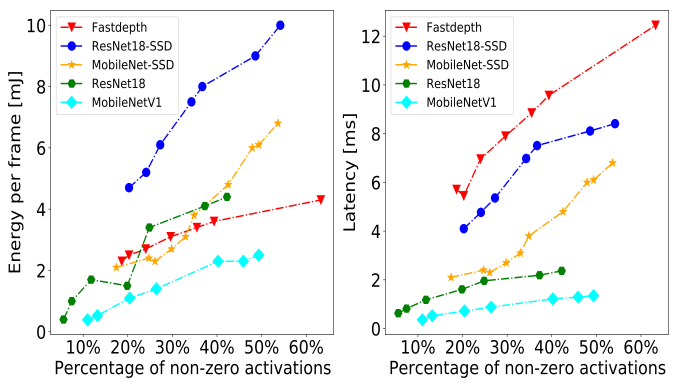

Fig. 5: Energy consumption (Left) and Latency (Right) of model inference (batch=1) with different levels of activation density.

because the slope value is related to the number of operations triggered per event (i.e. non-zero activations), which varies with the kernel size and channel depth of the layer. However, the relative improvements in energy consumption and latency are roughly proportional to the activation sparsity gains.

C. Joint optimization of ARTS and weight pruning

a) Sparsification interaction: Fig. 6 shows how the two sparsification techniques (ARTS and weight pruning) interact during joint optimization. In most circumstances, the sparsification space for pruning is constrained after activation sparsification; meanwhile, weight pruning also increases the non-zero activations while raising the prune ratio, particularly for pre-sparsified models. This reveals the fact that the sparsification executed first has a greater effect on the joint optimization, while the other has a smaller effect. As a result, it is critical to prioritize the implementation for various applications.

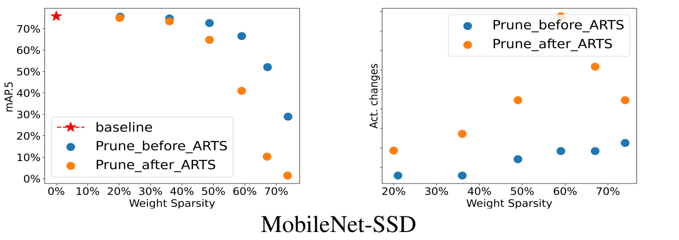

Fig. 6: Comparisons of pruned and non-pruned models on accuracy (Left) and non-zero activation increment (Right).

b) Results on joint optimization: Tab. IV shows the MAC reductions when using pruning and ARTS techniques consequently. As shown in the *Act.* columns, ARTS can eliminate the pruning blowback on the activations and further boost the activation sparsity. Furthermore, (*MAC.*) columns reveal the fact that both joint strategies are superior to any individual sparsification on MAC suppression. In addition, if we look at the 10^{th} and 16^{th} (*MAC.*) columns, we notice that for classification applications (e.g., image classification, object detection), the network can achieve more MAC reduction by executing the joint optimization in the order: baseline $\rightarrow$ ARTS $\rightarrow$ pruning, since many non-zero activations can be eliminated to draw the network attention to the important features only, especially in the deep layers (see Fig. 4). For applications with element-wise predictions (e.g., depth estimation), the network is more sensitive to the suppression of non-zero activations than to weight pruning. Thus, Strategy 2 (order: baseline $\rightarrow$ pruning $\rightarrow$ ARTS), dominated by weight pruning, achieves higher MAC reductions in those encoder-decoder architectures.

TABLE IV: Multiply-Accumulate Compute (MAC) reduction with two joint optimization strategies. MAC. is the percentage of non-zeros in MAC. P_m is the prune ratio on model parameters.

Model	Baseline			Strategy 1						Strategy 2					
	Original Training			ARTS		→	Prune			Prune		→	ARTS		
	Acc.	Act.	MAC.	Acc.	Act.	MAC.	Acc.	P_m	MAC.	Acc.	P_m	MAC.	Acc.	Act.	MAC.
MobileNetV1	93.59%	50%	47%	93.60%	14%	13%	93.57%	82%	**5%**	93.56%	49%	30%	93.62%	14%	12%
ResNet18	94.00%	42%	36%	94.01%	9%	11%	93.97%	97%	**1%**	93.86%	62%	15%	94.00%	12%	6%
MobileNet-SSD	75.80%	54%	37%	76.21%	26%	12%	75.40%	15%	**10%**	76.01%	30%	28%	76.01%	33%	13%
ResNet-SSD	75.73%	48%	36%	76.78%	18%	18%	76.30%	36%	**14%**	75.54%	36%	25%	75.57%	27%	17%
Fastdepth	77.28%	63%	61%	77.42%	30%	44%	77.16%	51%	26%	77.37%	67%	26%	77.31%	39%	**22%**

VI. Conclusions

In this paper, we presented an efficient training schedule called ARTS that gradually sparsifies activation maps in CNNs. ARTS achieves significantly higher activation-sparsity rates and shorter training times compared to the state-of-the-art. The technique has high relevance for event-based architectures, which can translate activation sparsity into proportional improvements in throughput, latency, and energy consumption at inference time. An analysis conducted for Neuronflow, a commercial event-based architecture, shows the effectiveness of ARTS, which can improve the latency/throughput by up to 2.5–8.4× and the energy consumption by 2.3–14.1× for various CNN architectures and AI applications. Furthermore, we show ARTS integration with standard weight pruning, resulting in an overall reduction in the compute requirements at inference by up to 4.5–88.0×, compared to a full execution of the neural network.

VII. Acknowledgement

This publication is supported by the ANDANTE project, which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876925. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Belgium, Germany, Netherlands, Portugal, Spain, Switzerland.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[2] A. G. Howard, M. Zhu, B. Chen *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[3] W.Diana and M.Fangchang and Y.Tien-Ju and K.Sertac and S.Vivienne, "FastDepth: Fast Monocular Depth Estimation on Embedded Systems," in *ICRA*, 2019.

[4] L, Wei and A, Dragomir and E, Dumitru and S, Christian and R, Scott E. and F, Cheng-Yang and B, Alexander C., "SSD: Single Shot MultiBox Detector." in *Proc. ECCV*, 2016, pp. 21–37.

[5] M. Siam, M. Badawy, M. Abdelrazek *et al.*, "A comparative study of real-time semantic segmentation for autonomous driving," in *Proc. CVPRW*, Jun. 2018.

[6] Z. Liu, J. Li, Z. Shen *et al.*, "Learning efficient convolutional networks through network slimming," in *Proc. ICCV*, Oct. 2017, pp. 2755–63.

[7] H. Wang, C. Qin, Y. Zhang, and Y. Fu, "Neural pruning via growing regularization," in *Proc. ICLR*, 2021.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *Proc. IEEE CVPR*, 2016.

[9] M. H. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2018.

[10] S. P. Singh and D. Alistarh, "Woodfisher: Efficient second-order approximations for model compression," *CoRR*, vol. abs/2004.14340, 2020.

[11] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, pp. 106–122, Aug. 2017.

[12] F. Akopyan, J. Sawada, A. Cassidy *et al.*, "Truenorth: Design and tool flow of a 65 mW 1 M neuron programmable neurosynaptic chip," *IEEE Trans. CAD*, vol. 34, no. 10, pp. 1537–1557, 2015.

[13] M. Davies, N. Srinivasa, T.-H. Lin *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[14] O. Moreira, A. Yousefzadeh, F. Chersi *et al.*, "Neuronflow: a neuromorphic processor architecture for live AI applications," in *Proc. DATE*, 2020, pp. 840–845.

[15] G. Georgiadis, "Accelerating convolutional neural networks via activation map compression," in *Proc. IEEE CVPR*, Jun. 2019, pp. 7078–7088.

[16] M. Kurtz, J. Kopinsky, R. Gelashvili *et al.*, "Inducing and exploiting activation sparsity for fast inference on deep neural networks," in *Proc. ICML*, Jul. 2020, pp. 5533–5543.

[17] S. Shi and X. Chu, "Speeding up CNNs by exploiting the sparsity of rectifier units," *CoRR*, vol. abs/1704.07724, 2017.

[18] S. Han, X. Liu, H. Mao *et al.*, "EIE: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Comput. Archit. News*, vol. 44, Feb. 2016.

[19] Z. Gong, H. Ji, C. W. Fletcher, C. J. Hughes, and J. Torrellas, "SparseTrain: Leveraging dynamic sparsity in software for training DNNs on general-purpose SIMD processors," in *ACM PACT*, 2020, p. 279–292.

[20] K. Goetschalckx and M. Verhelst, "Breaking high-resolution cnn bandwidth barriers with enhanced depth-first execution," *IEEE J. Emerg. Sel. Top. Circuits Syst*, vol. 9, no. 2, pp. 323–331, 2019.

[21] BrainChip Ltd. (2021, Feb.) Akida neural processor system-on-chip.

[22] J. Albericio, P. Judd, T. Hetherington *et al.*, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *ACM/IEEE Proc. ISCA*, 2016, pp. 1–13.

[23] Q. Yang, J. Mao, Z. Wang, and H. Li, "DASNet: Dynamic activation sparsity for neural network efficiency improvement," *CoRR*, vol. abs/1909.06964, 2019.

[24] M. Rhu, M. O'Connor, N. Chatterjee *et al.*, "Compressing DMA engine: Leveraging activation sparsity for training deep NNs," 2018, pp. 78–91.

[25] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," in *Proc. ICLR*, 2020.

[26] D. H. Le and B.-S. Hua, "Network pruning that matters: A case study on retraining variants," in *Proc. ICLR*, 2021.

[27] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *CoRR*, vol. cs.LG/0408058, 2004.

[28] H. Yang, W. Wen, and H. Li, "DeepHoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures," *CoRR*, vol. abs/1908.09979, 2019.

[29] S. Ward-Foxton. (2020, Jan.) GrAI Matter research gives rise to AI processor for the edge. EETimes. [Online]. Available: www.eetimes.com/grai-matter-raises-14m-for-sparsity-driven-ai-soc/

[30] A. Krizhevsky, "Learning multiple layers of features from tiny images."

[31] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE CVPR*, 2012.

[32] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proc. ECCV*, 2012.

[33] M. Cordts, M. Omran, S. Ramos *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE CVPR*, 2016.