2-in-1 Accelerator: Enabling Random Precision Switch for Winning Both Adversarial Robustness and Efficiency

ABSTRACT

The recent breakthroughs of deep neural networks (DNNs) and the advent of billions of Internet of Things (IoT) devices have excited an explosive demand for intelligent IoT devices equipped with domain-specific DNN accelerators. However, the deployment of DNN accelerator enabled intelligent functionality into real-world IoT devices still remains particularly challenging. First, powerful DNNs often come at a prohibitive complexity, whereas IoT devices often suffer from stringent resource constraints. Second, while DNNs are vulnerable to adversarial attacks especially on IoT devices exposed to complex real-world environments, many IoT applications require strict security. Existing DNN accelerators mostly tackle only one of the two aforementioned challenges (i.e., efficiency or adversarial robustness) while neglecting or even sacrificing the other. To this end, we propose a *2-in-1 Accelerator*, an integrated algorithm-accelerator co-design framework aiming at winning both the adversarial robustness and efficiency of DNN accelerators. Specifically, we first propose a Random Precision Switch (RPS) algorithm that can effectively defend DNNs against adversarial attacks by enabling random DNN quantization as an in-situ model switch during training and inference. Furthermore, we propose a new precision-scalable accelerator featuring (1) a new precision-scalable MAC unit architecture which spatially tiles the temporal MAC units to boost both the achievable efficiency and flexibility and (2) a systematically optimized dataflow that is searched by our generic accelerator optimizer. Extensive experiments and ablation studies validate that our *2-in-1 Accelerator* can not only aggressively boost both the adversarial robustness and efficiency of DNN accelerators under various attacks, but also naturally support instantaneous robustness-efficiency trade-offs adapting to varied resources without the necessity of DNN retraining. We believe our *2-in-1 Accelerator* has opened up an exciting perspective of robust and efficient accelerator design.

1. INTRODUCTION

Deep neural networks' (DNNs) performance breakthroughs and the advent of billions of Internet of Things (IoT) devices have triggered an increased demand for DNN-powered intelligent IoT devices. However, DNNs' deployments into real-world IoT devices still remain challenging. First, powerful DNNs' prohibitive complexity stands at odd with the stringent resource constraints of IoT devices. Second, while DNNs are vulnerable to adversarial attacks, many IoT applications require strict security under dynamic and complex real-world environments. Therefore, techniques boosting both DNNs' efficiency and robustness are highly desired.

To tackle the first challenge above, various domain-specific DNN accelerators [11, 31] have been developed to customize their algorithm-to-mapping methods (i.e., dataflows) and micro-architecture towards the workloads of DNNs to achieve orders-of-magnitude acceleration efficiency improvement over general computing platforms. In parallel, various techniques have been proposed to defend DNNs against adversarial attacks showing promising performance to address the aforementioned robustness challenge. Among them, adversarial training [40, 57, 66, 68], which augments the training set with adversarial samples generated on-the-fly during training, is currently the most effective method. Furthermore, recognizing that both efficiency and robustness are critical to many DNN-powered intelligent applications, pioneering efforts [19, 54, 67] attempt to defend against adversarial attacks within DNN accelerators. Nevertheless, the art of robustness-aware DNN accelerators is still in its infancy, and existing defensive accelerators against adversarial attacks rely on additional detection networks/modules to detect/defend adversarial samples during inference, thus inevitably compromising their accelerator efficiency.

Considering that quantized DNNs are very promising as efficient DNN solutions and also highly desirable in many IoT applications, we first ask an intriguing question: "*Is it possible to leverage quantization to boost DNNs' robustness*?", despite the fact that quantized DNNs have been shown to degrade the models' adversarial robustness unless being equipped with sophisticated regularization schemes [61]. This is inspired by (1) [12, 36, 69] showed that random permutations on the inputs can certifiably defend DNNs against adversarial attacks, and (2) [69] found that weight perturbations are a good complement for input perturbations, because they can narrow the robust generalization gap as weights globally influence the losses of all examples. We thus hypothesize that quantization noise can be leveraged to provide similar effects as permutations to the weights/activations and thus enhance DNNs' robustness, motivating our random precision switch (RPS) algorithm that wins both efficiency and robustness of quantized DNNs. Furthermore, motivated by the bottlenecks of existing precision-scalable accelerators, we further develop a new accelerator to enhance the acceleration efficiency of RPS equipped DNNs. Specifically, we make the following contributions:

- We propose an integrated algorithm-accelerator co-design framework dubbed *2-in-1 Accelerator*, aiming at winning both the adversarial robustness and acceleration

efficiency of DNN accelerators.

- *2-in-1 Accelerator*'s algorithm: We provide a new perspective regarding the role of quantization in DNNs' robustness, and propose a Random Precision Switch (RPS) algorithm that can effectively defend DNNs against adversarial attacks by enabling random DNN quantization as an in-situ model switch during training and inference. RPS equipped DNNs with fixed-point precisions even outperform the full-precision models' robustness.
- *2-in-1 Accelerator*'s architecture: We develop a new precision-scalable accelerator highlighting (1) a new multiply-accumulate (MAC) unit architecture which spatially tiles the temporal MAC units to boost both the achievable efficiency and precision-scalable flexibility and (2) a systematically optimized dataflow searched by our generic accelerator optimizer, largely outperforming existing precision-scalable accelerators.
- We perform a thorough evaluation of *2-in-1 Accelerator* on six DNN models and three datasets under various adversarial attacks, and find that our *2-in-1 Accelerator* achieves up to 7.58× better energy efficiency, 4.59×/36.5× higher throughput over precision-scalable/robustness-aware accelerators, and 24.48% improvement in robust accuracy. We believe that our *2-in-1 Accelerator* framework has not only demonstrated an appealing and effective real-world DNN solution, but also opened up an exciting perspective for winning both robustness and efficiency in DNN accelerators

2. 2-IN-1 ACCELERATOR: ALGORITHM

In this section, we present our RPS algorithm that can simultaneously boost DNNs' robustness and efficiency and thus serve as the algorithmic enabler of our 2-in-1 accelerator.

2.1 Preliminaries of adversarial robustness

[20] finds that DNNs are vulnerable to adversarial attacks, i.e., applying a small permutation δ within a norm ball ($\|\delta\| \leq \epsilon$) to the inputs can mislead DNNs' decisions. For example, the adversarial permutation δ under the ℓ_∞ attack [20] is generated by maximizing the objective:

$$\max_{\|\delta\|_\infty \leq \epsilon} \ell(f_\theta(x+\delta), y) \tag{1}$$

where ℓ is the loss function, θ is the weights of a DNN f, x and y are the input and the corresponding label, respectively.

To boost DNNs' robustness against adversarial attacks, adversarial training optimizing the following minimax problem is currently the strongest defense method [4]:

$$\min_{\theta} \sum_{i} \max_{\|\delta\|_\infty \leq \epsilon} \ell(f_\theta(x_i+\delta), y_i) \tag{2}$$

2.2 Inspirations from previous works

Previous works show that random smoothing or transformations [12, 24, 36, 71] on the inputs help robustify DNNs and [69] shows that weight perturbations are good complements for input perturbations as they globally influence the learning loss of all inputs. Following this spirit, [15,26,69] explicitly introduce randomness and permutations in the models' weights or activations. On the other hand, [39, 64, 66] show that model ensemble can help improve robustness at a cost of efficiency due to the required multiple models. These two aspects inspire us to rethink the connection between quantization's role in the permutations of DNN weights/activations and model robustness and to view a DNN model under different precisions as an in-situ ensemble to boost both robustness and efficiency. As introduced in Sec. 2.4, the proposed RPS algorithm can be seen as an in-situ model switch among different precision choices.

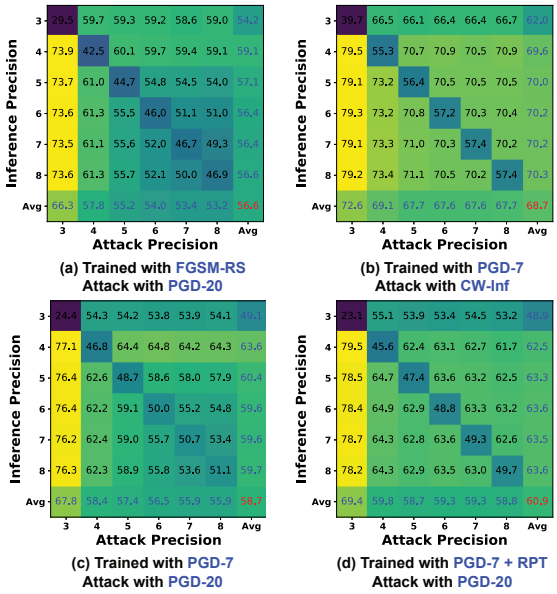

Figure 1: Visualizing the transferability of adversarial attacks between different precisions, where the robust accuracy under different training methods (PGD-7 [40] and FGSM-RS [68]) and attacks (PGD-20 [40] and CW-Inf [8]) is annotated.

2.3 Poor transferability between precisions

To validate our above hypothesis that a DNN model under different precisions can be seen as an in-situ ensemble, we empirically check the robustness of such an ensemble by evaluating the transferability of adversarial attacks between different precisions. As elaborated below, we find that the adversarial attacks transfer poorly between different precisions of an adversarially trained model, regardless of its adversarial training methods.

Experiment settings. We conduct experiments that adopt adversarial attacks generated under one precision to attack the same adversarially trained model quantized to another precision. In particular, we apply PGD-20 [40] and CW-Inf [8] attacks, to PreActResNet-18 (following [68]) which is adversarially trained using different adversarial training methods [40,68] using an 8-bit linear quantizer [29] under training settings introduced in Sec. 4.1. We annotate the robust accuracy evaluated on adversarial examples in Fig. 1 where the attack precision denotes the precision for generating attacks which are adopted to attack the same model quantized to another inference precision. The diagonal elements denote the robust accuracy with the same attack/inference precision and the non-diagonal elements denote the robust accuracy under transferred attacks from different precisions.

Observations. As observed from Fig. 1 (a)~(c), we can find that (1) training and attacking at the same low precisions indeed notably degrades the robust accuracy, as shown in the diagonals of Fig. 1, aligning with observations in [38]; (2) it's more difficult for adversarial attacks generated under one precision to fool the same adversarially trained model quantized to a different precision, regardless of the relative difference between the two precisions; (3) the poor transferability is consistent across different adversarial training methods of the model and attacks; and (4) the average robust accuracies of all precisions under white-box attacks are consistently higher than the full-precision models trained with the corresponding adversarial training methods, indicating that randomly selecting an inference precision can potentially provide effective defense.The *full-precision* accuracies of PreActResNet-18 trained with PGD-7/FGSM-RS are 51.2%/47.1%, respectively.

Analysis. The key conclusion is that for white-box attacks, adversarial attacks generated at one precision transfer poorly to another precision. We hypothesize that this poor transferability is because adversarial perturbations are shielded by the quantization noise between the two precisions, which can not be effectively learned by gradient-based attacks. In particular, for a k-bit linear quantizer, the quantized activation A_q (the same for weights) can be formulated as $A_q = S_k \lfloor \frac{A}{S_k} \rceil$, where $\lfloor \cdot \rceil$ is the rounding operation and $S_k = \frac{A_{max} - A_{min}}{2^k - 1}$ is the scale factor. For standard quantization, gradient-based attacks can effectively learn the rounding effect via straight-through estimation [74], i.e., $\frac{\partial L}{\partial A} \approx \frac{\partial L}{\partial A_q}$, where L is the loss function. However, the quantization noise $S_m \lfloor \frac{A}{S_m} \rceil - S_n \lfloor \frac{A}{S_n} \rceil$ between two different precisions m-bit and n-bit cannot be effectively learned in a gradient-based manner, thus adversarial perturbations can be shielded by the quantization noise.

2.4 RPS towards robust DNNs

Motivated by the poor transferability between different precisions of an adversarially trained model, we propose an RPS algorithm to boost both model robustness and efficiency via enabling random DNN quantization as an in-situ model switch during training and inference.

RPS training. We propose an RPS training pipeline to (1) maintain a decent natural accuracy when the model is directly quantized to different precisions during inference, and (2) further increase the difficulty of transferring adversarial examples between different precisions. To this end, we adversarially train a model from scratch via (1) randomly selecting a precision from a candidate set in each iteration for generating adversarial examples and updating the model with the selected precision, and (2) equipping the model with switchable batch normalization (SBN) [21, 30] to independently record the statistics of different precisions given their corresponding adversarial examples. In particular, randomly selecting a training precision improves the capability of instant precision switch during inference and SBN enlarges the gap between different inference/attack precisions inspired by [21, 30, 70] which separately handles the statistics of different inputs. As shown in Fig. 1 (d), the same adversarially trained model equipped with RPT shows larger robust gaps between different inference/attack precisions, especially under larger precision, as compared to the corresponding ones in Fig. 1 (c). Note that during inference, the multiplication and addition operations of SBN can be fused into the scale factors of linear quantizers [29] and the model bias, respectively, thus does not require additional modules over existing low precision accelerators.

RPS inference. Given a model adversarially trained via our RPS training method termed RPT, the proposed RPS inference method dubbed RPI randomly selects one precision from an inference precision set to quantize the model's weights and activations during inference. Based on the analysis in Sec. 2.3, randomly selecting an inference precision can greatly degrade the effectiveness of adversarial attacks as long as the attacks are not generated under the same precision, as consistently observed in Figs. 1.

The RPS training and inference algorithms (i.e., RPT and RPI) on top of PGD-7 [40] adversarial training are summarized in Alg. 1, which is similar when applying on top of other adversarial training methods.

2.5 Instant trade-offs between robustness and efficiency

In addition to win both robustness and efficiency, another benefit of our RPS algorithm is the instant trade-off capability between DNNs' robustness and efficiency during run-time to adapt to (1) the safety conditions of the external environments and (2) the remaining resource (e.g., battery power) on the device. Specifically, our RPS achieves this via (1) switching to lower precisions when enabling random precision inference to trade robustness in less dangerous environments for a higher average efficiency, or (2) directly adopting a static low precision training under safe environments to pursue merely high efficiency. This property can be highly desirable in real world applications especially intelligent IoT ones. We will next discuss the proposed accelerator that can not only improve the hardware execution efficiency of DNNs resulting from our RPS algorithm but also set a new record of precision-scalable acceleration.

3. 2-IN-1 ACCELERATOR: ARCHITECTURE

In this section, we introduce our proposed accelerator architecture dedicated for variable-precision DNNs (e.g., RPS equipped DNNs in Sec. 2) to achieve much improved acceleration efficiency. In particular, we first identify and analyze the bottlenecks of existing precision-scalable accelerators in Sec. 3.1, and present a new MAC unit architecture in Sec. 3.2 and an automated accelerator optimizer in Sec. 3.3 that together tackles the aforementioned bottlenecks.

3.1 Bottlenecks of SOTA precision-scalable accelerators

Despite the impressive performance achieved by SOTA precision-scalable accelerators [32, 35, 44, 45, 55, 56, 58, 59], they are still limited in their achievable acceleration performance especially when accelerating more complex variable-precision DNNs, e.g., RPS equipped DNNs in which all the layers may switch their precision to any possible precision in a candidate set during inference. The bottlenecks of SOTA precision-scalable accelerators are described below.

3.1.1 Dilemma between flexibility and performance

Algorithm 1 The RPS Algorithm (i.e., RPT and RPI)

Require: Training dataset D_{train}, model f_θ, precision set Set_Q, total training epochs T, step size α, adversarial dataset D_{adv} generated on f_θ by attackers

=== RPS Training ===
Equip f_θ with SBN
for $epoch \in [1, T]$ **do**
  for $(x, y) \in D_{train}$ **do**
    Randomly select a precision q from Set_Q
    Obtain f_θ^q by quantizing f_θ to q-bit
    $\delta = 0$ or random initialized
    for $t \in [1, 7]$ **do**
      $\delta = clip_\epsilon\{\delta + \alpha \cdot sign(\nabla_\delta \ell(f_\theta^q(x+\delta), y))\}$
    end for
    $\theta = \theta - \nabla_\theta \ell(f_\theta^q(x+\delta), y)$
  end for
end for
=== RPS Inference ===
for $x_{adv} \in D_{adv}$ **do**
  Randomly select a precision q from Set_Q
  Obtain f_θ^q by quantizing f_θ to q-bit
  Evaluate $\hat{y} = f_\theta^q(x_{adv})$
end for
return $\{\hat{y}\}$

Bottleneck. While various-precision DNNs have gained growing interest thanks to their advantages of enabling instantaneous energy-accuracy trade-off which is highly desirable in many DNN applications such as DNN-powered IoT ones, existing precision-scalable accelerators still struggle in the dilemma between their favored flexibility (i.e., support a large set of precisions) and achieved acceleration performance.

Analysis. SOTA precision-scalable accelerators can be categorized into two classes, i.e., *temporal* and *spatial* designs. The temporal designs [32, 58] adopt bit-serial MAC units to execute a part of the bit operations between two operands during each cycle and then accumulate the results temporally via additional shift logic circuits to support variable precision inference; while the spatial architectures [44, 59] first split the execution of high precision multiplications into several 2-bit multipliers, and then exploit combinational logic circuits to dynamically compose and decompose the 2-bit multipliers to construct variable-precision MAC units. Both designs have their advantages and disadvantages:

On the one hand, temporal designs are inferior in their achieved throughput under lower precisions (<8-bit) compared with spatial designs as validated in [7], since the area of their required shifters and accumulators are determined by their supported highest precision and thus can dominate the area cost, limiting their efficiency normalized over area [59]. On the other hand, spatial designs can only support a limited set of predefined precisions, e.g., 2-/4-/8-/16-bit for Bit Fusion [59], if considering an affordable cost for their required configurability logic circuits due to the spatial constraints of their MAC units, while the precision choices in temporal designs are more flexible as higher precisions can naturally be supported by using more temporal cycles. Therefore, there exists a dilemma between the achieved flexibility and efficiency in SOTA precision-scalable accelerators.

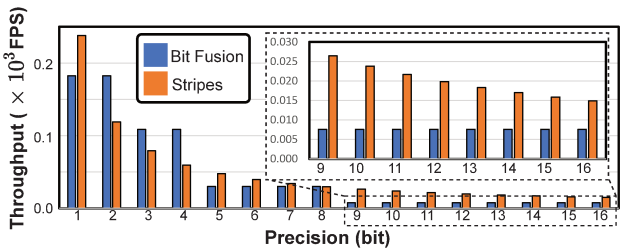

Figure 2: Throughput under different precisions of Bit Fusion and Stripes for accelerating ResNet-50 on ImageNet.

Validation. To validate the above analysis, we show the throughput under different precisions (the same for weights and inputs) of two representative spatial and temporal precision-scalable accelerators (i.e., Bit Fusion [59] and Stripes [32]) in Fig. 2, when accelerating ResNet-50 on ImageNet. The detailed simulation settings can be found in Sec. 4.1. We can observe that the spatial design, Bit Fusion, (1) achieves a higher throughput compared with Stripes under its *supported* precisions (i.e., <8-bit, the most commonly adopted precisions in quantized DNNs [5, 16, 33, 51]; (2) Bit Fusion leads to under-utilization of the hardware resources under its *unsupported* precisions where it has to adopt the closest supported but higher precision; (3) Bit Fusion shows inferior throughput under precisions larger than 8-bit since it has to execute each Bit Bricks four times when the operands' precision is higher than 8-bit. In contrast, while the temporal design, Stripes, is inferior to Bit Fusion under Bit Fusion's supported low precisions, it scales well with the precision, e.g., a consistent improvement in throughput as the execution precision decreases. This set of experiments demonstrates that SOTA precision-scalable accelerators inevitably suffer from the dilemma to trade-off between their achieved flexibility and efficiency, motivating our proposed new accelerator.

3.1.2 *Heavy shift-add overhead for scalable-precision*

Bottleneck. To support variable-precision configurability, existing precision-scalable accelerators require a heavy shift-add overhead, e.g., the shifters in the bit-serial units of the temporal designs [32] and the shifters for composing various 2-bit multipliers in the spatial designs [59] introduce significant or even dominant area and energy costs.

Analysis from related works. The size of the required shifters and accumulators in the *temporal designs* are determined by its highest supported precision and thus can dominate the area cost [59], e.g., the shifter and the accumulator use up around 90% of the total area in a temporal design supporting up to 16-bit, greatly limiting their achievable benefits and leading to inferior normalized efficiency per area. Similar observations have been drawn in [7] that compared with spatial designs, temporal designs have a worse normalized performance, i.e., throughput/area. On the other hand, for *spatial designs*, [7] shows that their MAC unit can require up to 4.4× the area of a standard MAC unit due to the overhead of their scalable units using sub-computation parallelism, and [55] also finds that the shift-add logic circuit in Bit Fusion for supporting precision-scalable configuration occupies a surprisingly large area (67%) and consumes a majority of power consumption (79%). These observations motivate us to explore a new precision-salable accelerator to

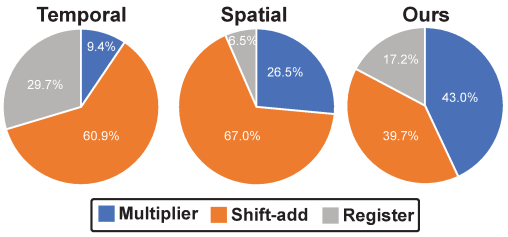

Figure 3: Area breakdown of the MAC units based on SOTA temporal/spatial designs and our proposed design.

reduce the shift-add logic overhead and thus to better allocate the limited area for more MAC units.

Validation. In Fig. 3, we show the area breakdown of the MAC units in Bit Fusion [59], a temporal design reported by Bit Fusion, as well as our proposed architecture introduced in Sec. 3. We can see that the shift-add logic occupies 60.9%/67.0% of the total area in the MAC units of the temporal/spatial designs. In contrast, our design reduces the area of shift-add logic to 39.7% via the techniques proposed in Sec. 3, thus leading to a better performance/area.

3.1.3 Fixed or Limited dataflow optimization

Bottleneck. The dataflow of DNN accelerators can largely impact their acceleration efficiency [41]. For variable-precision DNNs (e.g., RPS equipped ones), each layer might be executed at any precision of the candidate precision set, making it more challenging to find an optimal dataflow for all the cases. For example, a 20-layer DNN with 5 precision choices can require up to a total of 100 different dataflows for achieving the best average efficiency, in contrast to only 20 for a static layer-wise mixed-precision DNN.

Analysis. As analyzed in Eyeriss [11], dataflows can be described as the tiling strategies, including the loop order and tiling factors, across each memory hierarchy. Most of existing precision-scalable accelerators adopt a fixed dataflow within their memory hierarchies or only conduct a limited dataflow optimization. In particular, [32, 55, 58] all use a fixed NoC (Network-on-Chip as defined in [11]) design, i.e., fixing the tiling strategies along both the two dimensions of the MAC array; and Bit Fusion [59] provides a dataflow optimization tool which only considers the loop order optimization for the global buffer and thus lacks flexible dataflow support for other memory hierarchies. Considering that different networks/layers with different precisions might favor a different dataflow, a more comprehensive optimizer is necessary to find the optimal dataflow for further boosting improving the efficiency of precision-scalable accelerators.

3.2 The proposed MAC unit architecture

In this subsection, we introduce the proposed MAC unit architecture. Specifically, we show how a vanilla spatial-temporal MAC unit architecture in Sec. 3.2.1 is evolved into our proposed MAC unit architecture step by step through the optimization methods in Sec. 3.2.2 ~ 3.2.3, and finally present the overall accelerator architecture in Sec. 3.2.4.

3.2.1 A spatial-temporal design

Key idea. As analyzed in Sec. 3.1.1, there exists an inevitable trade-off between bit-level flexibility and acceleration efficiency in temporal and spatial designs. As both

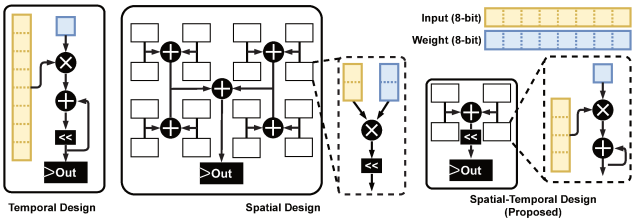

Figure 4: The MAC unit of the temporal design, spatial design, and our spatial-temporal design which spatially tiles the temporal units to marry the advantages of both temporal and spatial designs for variable precision execution. For 8-bit weight and input in this case, it takes 8 cycles, 1 cycles, and 4 cycles for the temporal, spatial, and our design, respectively.

flexibility and efficiency are critical for real applications, we propose a new spatial-temporal MAC unit architecture which spatially tiles the temporal units to combining the advantages of both temporal and spatial designs. In Fig. 4, we show an overview of the MAC unit in the (a) temporal, (b) spatial, and (c) our proposed spatial-temporal designs. We tile the temporal units, i.e., bit-serial units, in the same manner as the Bit Bricks in Bit Fusion [59] so that they can be dynamically composed to support variable precisions, e.g., each of the four bit-serial units takes four cycles to calculate a 2-bit $\times$ 2-bit partial product, the results of which are then fused via shift and accumulation to obtain the final 4-bit $\times$ 4-bit results.

Advantages of the spatial-temporal design. First, our spatial-temporal design maintains a high flexibility in the execution precision choices. Spatial designs [59] can only support limited precision choices (like 2-/4-/8-/16-bit) while our design can flexibly support more commonly used precision, e.g, each of the four bit-serial units can take three cycles to calculate a total of four 3-bit $\times$ 3-bit products, or one 6-bit 6-bit product via dynamic composition. Second, the smaller size (i.e., the supported maximal precision) of the bit-serial units in our spatial-temporal design will help mitigate the area bottleneck caused by the shift-add logic for precision configuration. In particular, one major bottleneck of temporal designs when supporting a high bit-level flexibility is that their shifters and accumulators within each bit-serial module are determined by their highest supported precision, e.g., dominating a 90% of the area in a 16-bit bit-serial unit, as pointed out by [59]. Our spatial-temporal design tackles this bottleneck via spatially composing bit-serial units of smaller sizes, i.e., each bit-serial unit can support up to 4-bit $\times$ 4-bit to constrain the maximal size required by the shifters. More importantly, the number of the required shift-add logic within the bit-serial unit and between different units for dynamic composition can be aggressively reduced with further optimization as introduced in Sec. 3.2.2 and Sec. 3.2.3.

Note that Bit Fusion also adopts a temporal-spatial manner for 16-bit inference by temporally executing 8-bit inference with their spatial unit for four cycles to compose a 16-bit result to avoid more complex logic for precision configurability, e.g., shifters of larger sizes. However, their temporal execution of the spatial units cannot benefit the bit-level flexibility like our design which spatially tiles the temporal units.

Spatial-temporal scheduling for different precisions. In our design, each bit-serial unit supports up to 4-bit $\times$ 4-bit calculation and each MAC unit adopts up to four bit-serial units

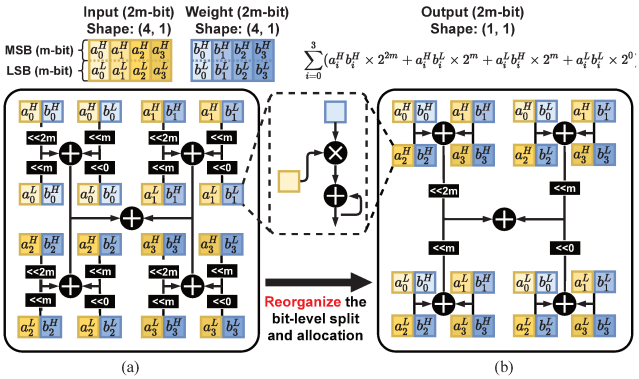

Figure 5: Reorganizing the bit-level split and allocation reduces the number of shifters by $1/n$ (n=4 in this case, denoting the of partial sums) when handling the inputs and weights of 2m-bit. Here a_i^L/b_i^L is the first m-bit LSB of inputs/weights and a_i^H/b_i^H is the remaining MSBs of the i-th partial sum.

for calculating one partial sum, i.e., supporting up to 8-bit × 8-bit calculation. For dealing with the precision higher than 8-bit, we follow Bit Fusion to temporally execute the whole MAC unit and then accumulate their results, considering that (1) the cost of more complex precision configurability under higher precisions will be higher and (2) 8-bit or lower precisions are sufficient for most DNN inference without accuracy degradation [5, 16, 33, 51].

Next, we introduce the detailed schedule of our MAC unit that is conducted spatially across the bit-serial units and temporally across cycles under each precision. Specifically, for operands with precisions no more than 4-bit, each bit-serial unit will independently calculate one partial sum of the final output; For operands with up to 6-bit × 6-bit / 8-bit × 8-bit, each of the four bit-serial units calculates a partial product with up to 3-bit × 3-bit / 4-bit × 4-bit, and then all the partial products will be composed to the final result via shift and accumulation; For operands with more irregular precisions like 5-bit × 5-bit, we split it into (3-bit+2-bit)×(3-bit+2-bit), i.e., four bit-serial units will take the computation of 3-bit × 3-bit, 2-bit × 2-bit, and two 3-bit × 2-bit, respectively, and similarly, operands with 7-bit can be split into (4-bit+3-bit); and for operands higher than 8-bit, the calculation will be split to no more than 8-bit and temporally executed by the whole MAC unit as mentioned above, e.g., 12-bit × 12-bit can be split into four 6-bit × 6-bit, each of which will be sequentially executed by the MAC unit and then accumulated. The above analysis also works for asymmetrical precisions, e.g., 4-bit × 2-bit which takes only two cycles for each bit-serial units to complete the execution.

3.2.2 *Opt-1: Reorganize bit-level split/allocation*

Motivation. It's important to improve bit-level split and allocation of the inputs/weights for the MAC units in precision-scalable accelerators, considering that the overhead of the shifters and accumulators for precision configurability is coupled with the workload patterns [7]. For example, if each bit-serial unit in a MAC unit processes one bit-level partial product of different outputs, their outputs need to be accumulated by different accumulators, thus requiring a large area overhead. Therefore, we aim to reorganize the workloads, more specifically, the bit-level split and allocation strategy to reduce the required shifters and accumulators in a MAC unit.

Calculating multiple partial sums in one MAC unit. We increase the number of bit-serial units in each MAC unit to simultaneously calculate multiple partial sums of the same output pixel as shown in Fig. 5 (a), which implies that the weights come from different kernel rows (R) and columns (S) while the inputs come from different input channels (C) for calculating the partial sums. Therefore, all the partial sums can be directly accumulated in one accumulator regardless of the execution precision. From a tiling strategy perspective, we explicitly tile the R, S, or C dimension in the MAC unit for further improving the area/energy efficiency while freeing up the used dataflow in the NoC (i.e., MAC array) and global buffer levels for layerwise optimization as introduced in Sec. 3.3. Such a flexibility is necessary for dataflow optimization towards reducing the data movement cost of each layer. More importantly, simultaneously calculating multiple partial sums also bring out another opportunity to aggressively reduce the required shifters as introduced below.

Reorganize the bit-level split and allocation. The number of shifters for the dynamic composition of bit-serial units can be reduced via reorganizing the bit-level split and allocation strategy. Suppose that calculating the i-th partial sum of an operand a_i can be formulated as $a_i = a_i^H \times 2^m + a_i^L \times 2^0$ where a_i^L is the first m-bit LSB and a_i^H is the remaining MSBs, then the final result of one MAC unit can be formulated as the sum of the totally n partial sums:

$$\sum_{i=0}^{n-1}(a_i^H \times 2^m + a_i^L \times 2^0)(b_i^H \times 2^m + b_i^L \times 2^0) \tag{3}$$

$$= \sum_{i=0}^{n-1}(a_i^H b_i^H \times 2^{2m} + a_i^H b_i^L \times 2^m + a_i^L b_i^H \times 2^m + a_i^L b_i^L \times 2^0) \tag{4}$$

$$= \sum_{i=0}^{n-1}(a_i^H b_i^H) \times 2^{2m} + \sum_{i=0}^{n-1}(a_i^H b_i^L) \times 2^m + \sum_{i=0}^{n-1}(a_i^L b_i^H) \times 2^m + \sum_{i=0}^{n-1}(a_i^L b_i^L) \times 2^0 \tag{5}$$

The original design in Fig. 5 (a) corresponds to Eq. 4 where totally $4n$ shifters are required for combining the outputs from different temporal units, whereas the reformulation in Eq. 5 only requires 4 shifters. Inspired by this, instead of accumulating different partial sums, we adopt a *first-reduce-then-shift* strategy for the partial products of the same magnitude (i.e., requiring the same number of shifts) from different partial sums are organized as a group which is mapped into a set of bit-serial units as shown in Fig. 5 (b). In this way, the outputs of the bit-serial units in a group can be directly summed together without any shifters and the final result of one MAC unit is the combination of the outputs from different groups via a group-wise shift-add logic. This is equivalent to accumulate different partial sums in Fig. 5 (a) as formulated in Eq. 4 but the number of shifters **cross the bit-serial units** for precision configurability is reduced by $1/n$ as shown in Eq. 5.

3.2.3 *Opt-2: Fuse the shift-add logic of bit-serial units in one group*

As introduced in Sec. 3.2.2, the outputs from each group of the bit-serial units can be directly accumulated without any shifter between the bit-serial units. This property brings another significant benefit in that all the shift-add logic of the bit-serial units in one group can be fused into one shift-add logic, named group shift-add, as shown in Fig. 6 (the leftmost zoom-in of one group). In particular, since the total number

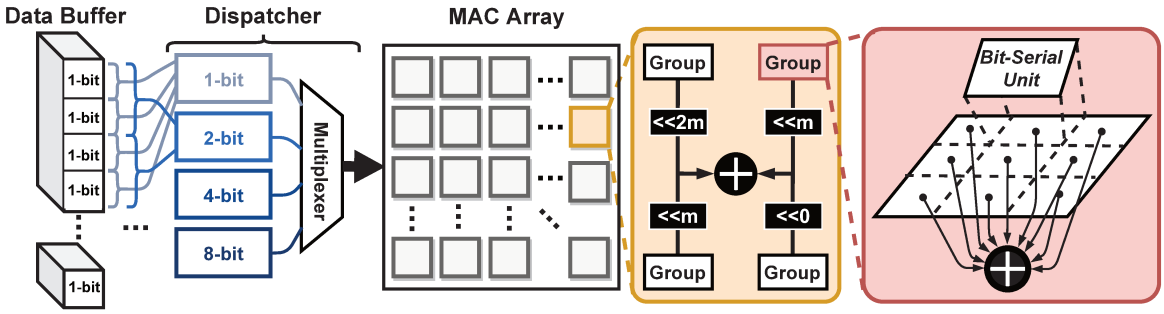

Figure 6: The overall architecture of the proposed *2-in-1 Accelerator*.

of shifts is the same for all the bit-serial units in one group, in each cycle the partial products of all the bit-serial units in one group can be directly summed together and the fed into the group shift-add module. Such an optimization reduces the required number of shifters **within the bit-serial units** by $1/n$. The synthesized results show that our final MAC unit design in Fig. 6 achieves 2.3× and 4.88× improvement in throughput/area and energy-efficiency/operation, respectively, compared with Bit Fusion under 8-bit×8-bit.

Note that (1) this optimization is specific to our design that organizes the bit-serial units into groups without the necessity of having unit-wise shifters and such opportunities do not exist in previous temporal/spatial designs; and (2) although the group shift-add can be potentially further combined with the group-wise shift-add, this will also increase the critical path and limit the system frequency. Thus, we keep them as two separate parts in our design.

3.2.4 *Overall architecture*

The overall *2-in-1 Accelerator* architecture is shown in Fig. 6, where the data is packed by a dispatcher which is implemented by a multiplexer to enable different granularities (i.e., 1/2/4/8-bit) for accessing the data buffer, and then passed to the MAC array as described in Sec. 3.2.1~ 3.2.3 for further processing. To this end, our *2-in-1 Accelerator* can (1) fully achieve the "win-win" in robustness and efficiency on top of the proposed RPS algorithm, and (2) support instantaneous robustness-efficiency trade-offs as validated in Sec. 4.4.

3.3 The proposed automated optimizer

It is well-known that both the dataflow and micro-architecture of a DNN accelerator is critical to its achievable efficiency. For example, [41] shows that different dataflows can result in a 10× difference in the accelerators' efficiency. Meanwhile, the number of all possible dataflows and micro-architectures for an accelerator can easily explode [72], which can be time consuming and might not even be practical to manually identify, and it can be greatly useful to have a generic accelerator optimizer that can automatically search for both the optimal dataflow and micro-architecture given the target acceleration efficiency and hardware resource (e.g., area). To this end, we propose an automated optimizer with two modes, i.e., (1) search for merely dataflows and (2) search for both the dataflows and micro-architectures given an area budget.

Searching for merely dataflows. For this mode, we adopt an evolutionary algorithm [43]. Specifically, the searchable factors include the tiling factors for each data dimension and the loop order for each memory hierarchy. Note that the optimal refresh location, which is the one occupying the most memory size without causing overflow, can be automatically derived since all the memory sizes are fixed in this mode. If all possible refresh locations cause overflows, the corresponding design is invalid and discarded. As shown in Alg. 2, we start from a population of randomly initialized for-loop descriptions and in each cycle, select the top 30% designs in terms of efficiency as a new population, and then conduct (1) crossover (i.e., generate a new design via randomly selecting two designs from the population and inserting one design's loop order in one memory hierarchy or tiling factors of one data dimension to the other design) and (2) mutation (i.e., generate a new design from a randomly selected dataflow via randomly permuting its loop order in one memory hierarchy or tiling factors of one data dimension to another choice). After enlarging the pool to the original population size, we will start a new cycle and iterate this process until reaching a predefined maximal cycle number. Note that in both modes, we adopt an open-sourced generic performance predictor of DNN accelerators [75] to obtain the efficiency for a given dataflow and micro-architecture pair.

Searching for both dataflows and micro-architectures. The search engine under this mode can be built on top of that for the above mode. Specifically, we predefine a design space with a set of available choices for the MAC array size and memory sizes in each memory hierarchy which are then synthesized to acquire the unit energy and area; and then adopt another evolutionary algorithm similar to Alg. 2 to explore the design space, where the efficiency of an micro-architecture is measured by calculating its average energy/throughput under different precisions after optimizing the dataflow via Alg. 2.

Note that for a fair comparison with the baselines, in this work we only optimize the dataflow of each workload and adopt the same memory/MAC array area with our baselines.

4. EXPERIMENT RESULTS

4.1 Experiment Setup

4.1.1 *Algorithm Setup*

Networks & datasets. We evaluate our RPS algorithm on three networks and three datasets which are the most commonly used ones in the robustness literature [40, 57, 68], i.e., PreActResNet-18 and WideResNet-32 on CIFAR-10/100 and ResNet-50 on ImageNet. We use a linear quantizer [29] for quantizing weights/activations to the same precision.

Training settings. We adopt four SOTA adversarial training methods, including FGSM [20], FGSM-RS [68], PGD-7 [40], and Free [57] and apply our RPS algorithm on top of

Algorithm 2 Evolutionary Search for Dataflows

Require: Architecture $arch$, Workload (layer information and execution precision), Total cycle number $Total\_Cycle$, Population size $Psize$

Initialize a $population$ of $dataflow$ with different loop orders and tiling factors according to the workload
for $cycle \in [1, Total\_Cycle]$ **do**
    Select the top 30% $dataflow$ from the $population$ based on the predicted efficiency of the workload
    while $size(population) < Psize$ **do**
        Randomly select two $dataflow$, do crossover, append to $population$ if valid
        Randomly select one $dataflow$, do mutation, append to $population$ if valid
    end while
end for
return The best $dataflow$ in the $population$

them. We follow their original papers for the adversarial training hyper-parameter settings and follow the model training settings in [40] and [57] for CIFAF-10/100 and ImageNet.

Attack settings. We consider the strong attacks including three white-box attacks PGD [40], AutoAttack [13], CW [8] and one gradient-free attack Bandits [28], with different numbers of iterations/restarts and permutation strengths $\epsilon = 8, 12, 16$. We assume the adversary adopts random precision from the same inference precision set as our RPS without losing generality since (1) any attack precision out of RPS's inference precision set will merely increase RPS's robust accuracy according to Fig. 1, and (2) while the adversary may select precisions with better attacking success rates, our RPS can also increase the probability of sampling more robust precisions for stronger defense, here we assume both the adversary and RPS adopt random precision for simplicity.

4.1.2 Accelerator Setup

Accelerator development and synthesis. In order to evaluate our proposed accelerator, we implement a custom cycle-accurate simulator, aiming to model the behavior of the synthesized circuits. The design parameters in the simulator are obtained from gate-level netlist and SRAM which are generated based on a commercial 28nm technology using the Synopsys Design Compiler and Memory compiler provided by the foundry. Specifically, proper activity factor are set at the input ports of the memory/computation units, and the energy is calculated using PrimeTime [65].

Baselines. We benchmark with two SOTA precision-scalable accelerators Bit Fusion [59] and Stripes [32], and one robustness-aware accelerator DNNGuard [67]. For a fair comparison, we adopt the same memory area and MAC array area with Bit Fusion, and we modify the unit energy of Bit Fusion's official simulator to scale it from 45nm to 28nm following the rule in [1]. For Stripes, thanks to the clear description of the design in the paper and the easy representation, we build a cycle-accurate simulator for it with the same memory/MAC array area with Bit Fusion and our design, and optimize the dataflow for its workloads with our automated optimizer.

Workloads. We adopt six networks (WideResNet-32 and ResNet-18 on CIFAR-10 with 32×32 inputs and AlexNet, VGG-16, ResNet-18/50 on ImageNet with 224×224 inputs) under 1∼16-bit as our workloads.

Table 1: Evaluating RPS on two networks and three adversarial training methods FGSM [20], FGSM-RS [68], and PGD-7 [40] on CIFAR-10 under different PGD attacks.

	PreActResNet-18			WideResNet-32		
Adversarial Training Method	Natural (%)	PGD-20 (%)	PGD-100 (%)	Natural (%)	PGD-20 (%)	PGD-100 (%)
FGSM	67.04	41.48	41.37	66.76	40.78	40.55
FGSM + **RPS**	80.58	**64.08**	**63.56**	64.09	**50.70**	**48.72**
FGSM-RS	86.08	41.76	41.13	89.95	45.33	44.77
FGSM-RS + **RPS**	82.11	**59.33**	**59.32**	87.87	**60.07**	**59.12**
PGD-7	82.02	51.17	50.93	85.25	54.61	54.36
PGD-7 + **RPS**	82.16	**65.15**	**64.88**	81.52	**66.75**	**66.28**

Table 2: Evaluating RPS on two networks trained with FGSM-RS [68] and PGD-7 [40] on CIFAR-100.

	PreActResNet-18			WideResNet-32		
Adversarial Training Method	Natural (%)	PGD-20 (%)	PGD-100 (%)	Natural Acc (%)	PGD-20 (%)	PGD-100 (%)
FGSM-RS	57.6	26.14	25.88	67.29	25.35	24.78
FGSM-RS + **RPS**	51.09	**36.75**	**37.18**	64.95	**39.18**	**38.36**
PGD-7	56.31	27.97	27.77	60.36	31.06	30.86
PGD-7 + **RPS**	56.2	**41.74**	**42.1**	58.41	**40.45**	**40.5**

4.2 Evaluate 2-in-1 Accelerator's algorithm

We evaluate the improvement in robustness via applying the proposed RPS on top of SOTA adversarial training methods. Note that all the baselines are SOTA adversarial training methods with full precision, i.e., no quantization is applied. Our RPS adopts a precision set of 4∼16-bit if not specifically stated and we provide an ablation study of different choices of precision sets in Sec. 4.2.4.

4.2.1 Benchmark on CIFAR-10/100/ImageNet

Benchmark on CIFAR-10. As summarized in Tab. 1, we can observe that (1) RPS consistently enhances the robust accuracy under PGD attacks, largely outperforming SOTA adversarial training methods with full precision. In particular, RPS achieves a 13.98%/12.14% higher robust accuracy under PGD-20 attacks on PreActResNet-18 and WideResNet-32, respectively, while notably improving the efficiency due to the low precision execution as evaluated in Sec. 4.3; (2) RPS also enhances the robust accuracy by 13.57%∼22.60% under PGD-20 attacks on top of FGSM/FGSM-RS. It is noteworthy that although FGSM adversarial training can be easily ineffective against iteration-based attacks [34], our RPS can still significantly improve its robust accuracy by 22.6%. In addition, we also benchmark with SOTA methods for improving the robustness of quantized networks [38] and find that under PGD-20 attack on CIFAR-10, our RPS achieves a 14.6% and 22.5% higher robust accuracy for $\epsilon = 8$ and 16, respectively, on the same PGD-7 trained network as compared to the best reported robust accuracy among all the settings in [38].

Benchmark on CIFAR-100. The observations on CIFAR-100 are consistent with CIFAR-10. In particular, RPS achieves 10.61%/13.77% and 13.83%/9.39% higher robust accuracy on top of FGSM-RS/PGD-7 training under PGD-20 attacks on PreActResNet-18 and WideResNet-32, respectively.

Benchmark on ImageNet. We further evaluate RPS on a larger scale dataset, i.e, ImageNet, as shown in Tab. 3. We can observe that RPS achieves a **triple-win** in terms of the natural accuracy, robust accuracy, and model efficiency on top of both adversarial training methods. In particular, RPS achieves a 7.65%/10.11% higher robust accuracy over

Table 3: Evaluating RPS on top of two adversarial training methods (FGSM-RS [68] and Free [57]) on ResNet-50 under PGD-10 and PGD-50 attacks with $\epsilon = 4$ on ImageNet.

Adversarial Training Method	Natural (%)	PGD-10 (%)	PGD-50 (%)
FGSM-RS	55.45	30.28	30.18
FGSM-RS + **RPS**	**63.21**	**37.93**	**37.12**
Free	60.21	32.77	31.88
Free + **RPS**	**64.58**	**42.88**	**42.72**

Table 4: Evaluating RPS under larger permutations on three networks with trained by PGD-7 on CIFAR-10.

Network	Adversarial Training Method	epsilon=12 Natural (%)	epsilon=12 PGD-20 (%)	epsilon=12 PGD-100 (%)	epsilon=16 Natural Acc (%)	epsilon=16 PGD-20 (%)	epsilon=16 PGD-100 (%)
PreAct	PGD-7	77.49	37.84	36.77	75.39	27.28	24.24
-ResNet18	PGD-7 + **RPS**	77.45	**56.73**	**56.62**	75.02	**50.54**	**50.16**
Wide	PGD-7	81.8	39.73	38.49	78.91	28.92	25.82
-ResNet32	PGD-7 + **RPS**	78.26	**53.74**	**52.42**	75.34	**46.82**	**44.85**

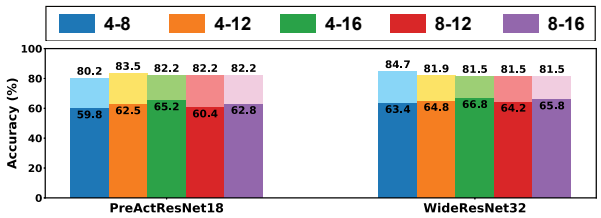

Figure 7: The natural and robust accuracy by PGD-7 training equipped with RPS under different precision sets. Deep and light colors denote robust and natural accuracy, respectively.

FGSM-RS [68] and Free [57], respectively, under the PGD-10 attack, indicating our RPS's scalability and applicability on large-scale and complex datasets.

4.2.2 *Benchmark under larger permutations*

We further evaluate RPS's scalability under larger permutations with PGD-7 training on CIFAR-10 as listed in Tab. 4. Interestingly, RPS even achieves larger robustness improvements. Tab. 4 shows that RPS leads to a 14.01%~18.89% and 17.90%~23.26% higher robust accuracy under PGD-20 attacks with $\epsilon = 12$ and 16, respectively. Larger improvements under stronger adversarial attacks validate RPS's applicability to more challenging environments.

4.2.3 *Benchmark under more attacks*

Considering many defense methods are found to be ineffective under stronger attack, we evaluate RPS against more attack types with different permutation strength. As observed from Tab. 5, RPS consistently improves the robust accuracy across different attacks/models/distortions, e.g., a higher robust accuracy of 6.88%~9.12% under Auto-Attack, which is one of the current strongest adaptive attack and more surprisingly, 9.97%~18.87% under CW-Inf attack, where we find the poor transferability between different attack/inference precisions is more notable. In addition, RPS achieves a 5.01%~24.48% higher robustness accuracy under Bandits attacks which is a gradient-free attack, indicating RPS does not suffer from the obfuscated gradient problem [4]. In fact, we find RPS does not show any characteristic behavior for obfuscated gradient discussed in [4].

4.2.4 *Influence of precision choices*

Table 5: Evaluating RPS on two networks trained by PGD-7 under more strong attacks with ϵ=8 and 12 on CIFAR-10.

	PreActResNet-18		WideResNet-32	
Attack Type	PGD-7	PGD-7 + **RPS**	PGD-7	PGD-7 + **RPS**
AutoAttack (ϵ=8)	47.18	**54.56**	51.66	**58.54**
AutoAttack (ϵ=12)	27.59	**35.83**	30.71	**39.83**
CW-Inf (ϵ=8)	57.88	**71.44**	62.13	**72.10**
CW-Inf (ϵ=12)	46.70	**65.57**	50.14	**66.99**
Bandits (ϵ=8)	59.75	**71.75**	63.49	**68.50**
Bandits (ϵ=12)	46.04	**70.52**	49.77	**67.01**

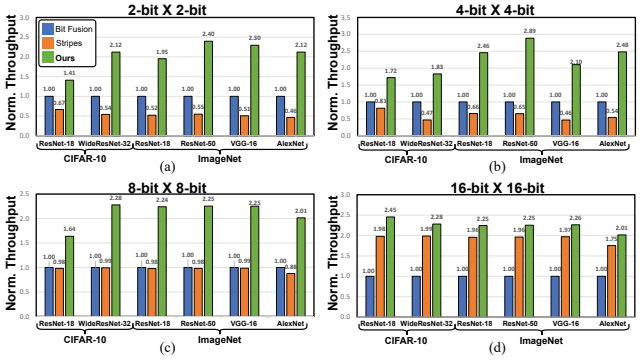

Figure 8: Normalized throughput comparison of Bit Fusion, Stripes, and our *2-in-1 Accelerator* on top of six networks and four execution precisions.

Fig. 7 shows the natural and robust accuracy (under PGD-20 attacks) of two networks with different precision sets, on top of PGD-7 training on CIFAR-10. We can see that (1) RPS consistently achieves a higher robust accuracy over SOTA PGD-7 training under different settings; and (2) a larger precision range for reducing the probability of hitting the attackers' precision is desired for improving robustness.

4.3 Evaluate 2-in-1 Accelerator's architecture

4.3.1 *Benchmark with Bit Fusion and Stripes*

Throughput comparison. We compare the throughput of Bit Fusion, Stripes, and our *2-in-1 Accelerator* on top of six networks and four execution precisions in Fig. 8. All the throughput results are normalized to the ones of Bit Fusion. We can observe that our design outperforms the baselines across all the networks and precisions with 1.41× ~ 2.88× and 1.15× ~ 4.59× higher throughput over Bit Fusion and Stripes, respectively. Such improvement mainly comes from (1) the high throughput/area of our proposed MAC unit architecture, and (2) the effectiveness of our automated optimizer in reducing the memory stalls to fully utilize the capability of our MAC unit. For example, when using ResNet-50 on ImageNet with 4x4-bit, our MAC unit design boosts throughput by 2.25× over Bit-Fusion, and the automated optimizer further improves the throughput by 1.28× via reducing memory stalls. In addition, we can observe that Bit Fusion shows better throughput over Stripes under execution precisions lower than 8-bit while showing inferior throughput at 16-bit, which is consistent with the analysis in Sec. 3.1.1 that Bit Fusion requires to execute each MAC unit four times for execution precisions higher than 8-bit. Althought our accelerator adopts a similar manner to deal with 16-bit, it can still achieve 1.15× higher throughput over Stripes, validating the superiority of

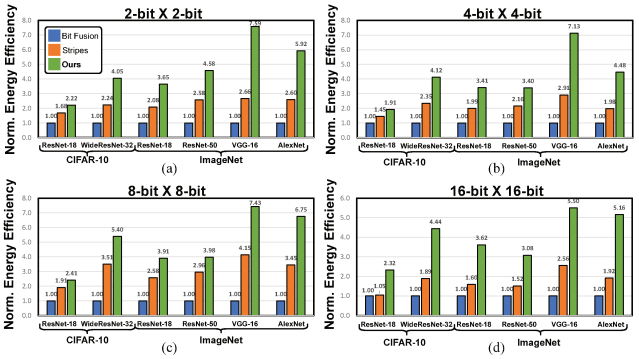

Figure 9: Normalized energy efficiency comparison of Bit Fusion, Stripes, and our *2-in-1 Accelerator* on top of six networks and four execution precisions.

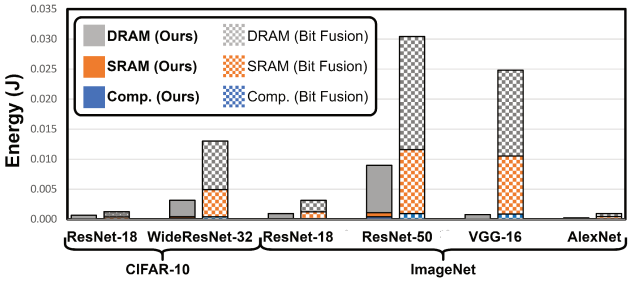

Figure 10: Energy breakdown of our *2-in-1 Accelerator* and Bit Fusion on six networks executed with 4-bit×4-bit.

the proposed spatial-temporal design.

Energy efficiency comparison. We compare the energy efficiency of Bit Fusion, Stripes, and our *2-in-1 Accelerator* on top of six networks and four execution precisions in Fig. 9. All the energy efficiency results are normalized to the ones of Bit Fusion. We can observe that our proposed architecture consistently achieves the best energy efficiency across all the networks and precisions with 1.91× ~ 7.58× and 1.25× ~ 2.85× energy efficiency over Bit Fusion and Stripes, respectively. Here we fully optimize the dataflow of Stripes so that it also outperforms Bit Fusion in terms of energy efficiency.

We also compare the energy breakdown between our design and Bit Fusion in Fig. 10. We can observe that although DRAM access still dominates the total energy, the energy for both computation in MAC and data movement (i.e., access DRAM and SRAM) are all reduced over Bit Fusion. The former is due to the higher energy efficiency/operation of our MAC unit and the latter is due to (1) the new opportunities in better mapping strategies brought by the proposed MAC unit with better throughput/area and output reuse, and (2) the effectiveness of our automated optimizer on a more flexible dataflow search space.

Throughput evolution with the execution precision. To further validate the scalability along different execution precisions of our *2-in-1 Accelerator* over spatial/temporal designs, we show the throughput under different precisions (the same weight/input precision) of Bit Fusion, Stripes, and our design when accelerating WideResNet-32 on CIFAR-10 and ResNet-50 on ImageNet. As observed in Fig. 11, our *2-in-1 Accelerator* shows both superior efficiency and flexibility as it (1) consistently outperforms both the baselines under

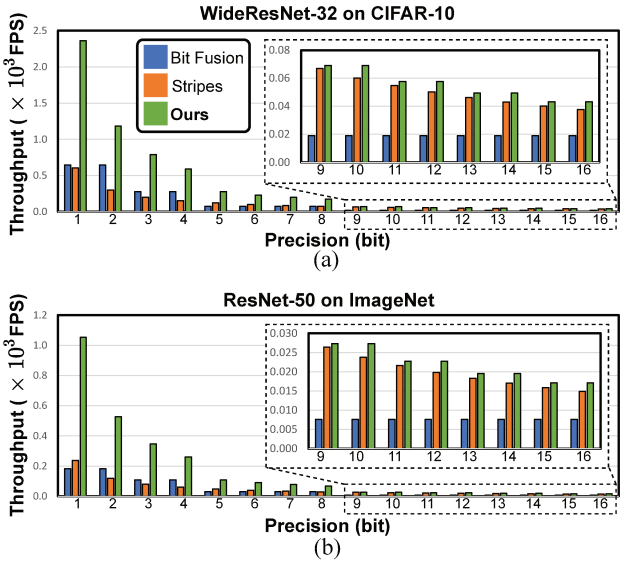

Figure 11: Throughput under different precisions of Bit Fusion, Stripes, and our *2-in-1 Accelerator* for accelerating WideResNet-32 on CIFAR-10 and ResNet-50 on ImageNet.

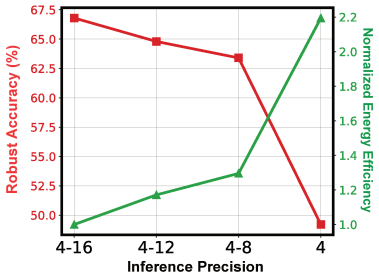

Figure 12: *2-in-1 Accelerator*'s instant robustness-efficiency trade-off on top of WideResNet-32 and CIFAR-10.

all the precisions by up to 4.42× higher throughput, and (2) achieves a consistent improvement in the throughput as precision decreases. In addition, under execution precisions lower than 8-bit which are the common choices of recent quantization works [5, 16, 33, 51], our design shows more than 1.82× higher throughput compared with the best baseline; under execution precisions higher than 8-bit which are inferior choices for spatial designs as analyzed in Sec. 3.1.1, our design still achieves higher throughput over Stripes, which benefits from the spatial-temporal design of our MAC unit.

4.3.2 Benchmark with robustness-aware accelerators

Boosting both robustness and efficiency in one accelerator is a significant feature and benefit of our *2-in-1 Accelerator*. We further benchmark with a SOTA robustness-aware accelerator DNNGuard [67] to show the superiority of our framework. In particular, we compare the throughput/area of our *2-in-1 Accelerator* and that of DNNGuard on AlexNet, VGG-16, and ResNet-50 which are reported by [67]. We find that our design achieves 36.5×/17.9×, 19.3×/9.5×, 12.8×/6.4× higher throughput compared with DNNGuard when adopting 4~8-bit/4~16-bit for accelerating AlexNet, VGG-16, ResNet-50, respectively. This indicates the superiority and practicality of deploying our *2-in-1 Accelerator* in real-world IoT applications where both security and efficiency matters.

4.4 Instant robustness-efficiency trade-offs of the 2-in-1 Accelerator

As analyzed in Sec. 2.5, our *2-in-1 Accelerator* also features the capability to enable instant robustness-efficiency trade-offs at run-time to adapt to both the safety conditions of the environments and the remaining power on the device. We show an example of executing WideResNet-32 with CIFAR-10 inputs on our *2-in-1 Accelerator* with different execution precisions (RPS with 4~16-bit, 4~12-bit ,4~8-bit, static 4-bit) and record the robust accuracy and the (average) energy efficiency. As shown in Fig. 12, our *2-in-1 Accelerator* can instantly switch between high precision sets, low precision sets, and static low precision to balance robustness and efficiency with a comparable natural accuracy (within 81.5%~84.7%).

5. RELATED WORKS

Adversarial attacks and defenses. [20] shows that small permutations onto the inputs can mislead DNNs' decisions, which is known as adversarial attacks. Later, stronger attacks, including both white-box [8, 13, 40, 46, 50] and black-box ones [3, 10, 23, 27, 28], are proposed to aggressively degrade the accuracy of the target DNN models. To defend DNNs against adversarial attacks, adversarial training [40, 57, 66, 68], which augments the training set with adversarial samples generated during training, is currently the most effective method. In parallel, other defense methods [6, 17, 24, 36, 37, 42, 63, 69, 73] have also been proposed. There has been a continuous competition between adversaries and defenders, and the readers are referred to [2, 9] for more attack/defense methods.

Robustness of quantized models. As both robustness and efficiency are critical for most DNN applications, pioneering works have strived to design robust quantized DNNs. In particular, [18, 48] propose robust binary neural networks (BNNs) and [53] adopts tanh-based quantization to increase robustness, while these works have been observed to suffer from the obfuscated gradient problem [4, 49], which is a false sense of security. Later, [38] finds that quantized DNNs are actually more vulnerable to adversarial attacks due to the error amplification effect, i.e., the magnitude of adversarial perturbation is amplified when passing through the DNN layers. To tackle this effect, [38, 60] propose robustness-aware regularization methods for DNN training, and [61] retrains the network via feedback learning [62]. In addition, [47] searches for layerwise precision and [22] constructs a unified formulation to balance and enforce the models' robustness and compactness, respectively. In contrast, our RPS algorithm *leverages quantization* to aggressively enhance robustness, which even largely surpasses the full-precision models.

Precision-scalable accelerators. To support variable precisions for different DNN models/layers, various precision-scalable accelerators have been proposed to dynamically and flexibly handle the varied workloads, which can be categorized into two classes, i.e., *temporal* and *spatial* designs. For temporal designs, pioneering works, such as Stripes [32], LOOM [58], and Tartan [14], adopt bit-serial MAC units to provide precision configurability, which can flexibly handle any prevision yet suffer from inferior efficiency per area over their spatial counterparts [7, 59], and more recently UNPU [35] fabricates a bit-serial DNN accelerator to support variable weight precisions while the activations use full precision. For spatial designs, Bit Fusion [59] proposes to use combinational logic to dynamically compose and decompose 2-bit multipliers to construct variable-precision MAC units; Later, BitBlade [55] improves Bit Fusion via pulling out the shifting logic of each MAC unit and sharing it across the multipliers to reduce the area overhead; DVAFS [44, 45] propose to turn off parts of the multipliers at low precision to increase the energy efficiency at a constant throughput; and DeepRecon [56] skips parts of the pipeline stages of a floating-point-multiplier to support either one 16-bit, two 12-bit, or four 8-bit multiplications. Detailed benchmarks for different precision-scalable MAC unit architectures can be found in [7]. Our proposed MAC unit architecture marries the best of both temporal and spatial designs and is integrated to construct a new precision-scalable accelerator, which consistently outperforms SOTA designs under various settings.

Robustness-aware DNN accelerators. Despite their importance for real-world applications, the art robustness-aware DNN accelerators is still in its infancy. Pioneering works [19, 54, 67] aim to defend against adversarial attacks within DNN accelerators at a cost of additional detection networks/modules. In particular, [54] proposes an end-to-end framework based on the voting results of multiple detectors , in parallel with the execution of the target DNN to detect malicious inputs during inference; [67] proposes an elastic heterogeneous DNN accelerator architecture to orchestrate the simultaneous execution of the target DNN and the detection network for detecting adversarial samples via an elastic management of the on-chip buffer and PE computing resources; and [19] builds an algorithm-architecture co-designed system to detect adversarial attacks during inference via a random forest module applied on top of the extracted features from the run-time activations. In addition, [52] builds a robustness-aware accelerator based on BNNs which, however, suffers from the obfuscated gradient problem [4] and the work in [25] strives to speed up the attack generation instead of the defense. Nevertheless, all the existing defensive accelerators rely on additional detection networks/modules to detect adversarial samples at inference time, and thus inevitably introduce additional energy/throughput/area overheads that compromise efficiency. In contrast, our work exploits the potential robustness within a DNN model via the proposed RPS algorithm to win both robustness and efficiency within one accelerator without introducing any extra modules.

6. CONCLUSION

Existing DNN accelerators mostly tackle only either efficiency or adversarial robustness while neglecting or even sacrificing the other. In this work, we propose a *2-in-1 Accelerator*, aiming at winning both the adversarial robustness and efficiency of DNN accelerators. *2-in-1 Accelerator* integrates a Random Precision Switch (RPS) algorithm that can effectively defend DNNs against adversarial attacks and a new precision-scalable accelerator featuring a spatial-temporal MAC unit architecture to boost both the achievable efficiency and flexibility and (2) a systematically optimized dataflow generated by our generic accelerator optimizer. Extensive experiments and ablation studies validate our *2-in-1 Accelerator*'s effectiveness and advantages.

REFERENCES

[1] "The scaling of mosfets, moore's law, and itrs." [Online]. Available: http://userweb.eng.gla.ac.uk/fikru.adamu-lema/Chapter_02.pdf

[2] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.

[3] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," in *European Conference on Computer Vision*. Springer, 2020, pp. 484–501.

[4] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning*. PMLR, 2018, pp. 274–283.

[5] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, "Lsq+: Improving low-bit quantization through learnable offsets and better initialization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 696–697.

[6] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018.

[7] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 697–711, 2019.

[8] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[9] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[10] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.

[11] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 367–379, 2016.

[12] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

[13] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2206–2216.

[14] A. Delmas, S. Sharify, P. Judd, and A. Moshovos, "Tartan: Accelerating fully-connected and convolutional layers in deep learning networks by exploiting numerical precision variability," *arXiv preprint arXiv:1707.09068*, 2017.

[15] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *arXiv preprint arXiv:1803.01442*, 2018.

[16] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.

[17] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[18] A. Galloway, G. W. Taylor, and M. Moussa, "Attacking binarized neural networks," *arXiv preprint arXiv:1711.00449*, 2017.

[19] Y. Gan, Y. Qiu, J. Leng, M. Guo, and Y. Zhu, "Ptolemy: Architecture support for robust deep learning," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 241–255.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[21] L. Guerra, B. Zhuang, I. Reid, and T. Drummond, "Switchable precision neural networks," *arXiv preprint arXiv:2002.02815*, 2020.

[22] S. Gui, H. Wang, C. Yu, H. Yang, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," *arXiv preprint arXiv:1902.03538*, 2019.

[23] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2484–2493.

[24] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[25] H. Guo, L. Peng, J. Zhang, F. Qi, and L. Duan, "Hardware accelerator for adversarial attacks on deep learning neural networks," in *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*. IEEE, 2019, pp. 1–8.

[26] Z. He, A. S. Rakin, and D. Fan, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 588–597.

[27] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2137–2146.

[28] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," *arXiv preprint arXiv:1807.07978*, 2018.

[29] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[30] Q. Jin, L. Yang, and Z. Liao, "Adabits: Neural network quantization with adaptive bit-widths," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2146–2156.

[31] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[32] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–12.

[33] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4350–4359.

[34] A. Kurakin, I. Goodfellow, S. Bengio *et al.*, "Adversarial examples in the physical world," 2016.

[35] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "Unpu: A 50.6 tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 218–220.

[36] B. Li, C. Chen, W. Wang, and L. Carin, "Certified adversarial robustness with additive noise," *arXiv preprint arXiv:1809.03113*, 2018.

[37] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1778–1787.

[38] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," *arXiv preprint arXiv:1904.08444*, 2019.

[39] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 369–385.

[40] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[41] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC*

Conference on Computer and Communications Security, 2017, pp. 135–147.

[42] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[43] B. L. Miller, D. E. Goldberg *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[44] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Dvafs: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 488–493.

[45] B. Moons and M. Verhelst, "A 0.3–2.6 tops/w precision-scalable processor for real-time large-scale convnets," in *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*. IEEE, 2016, pp. 1–2.

[46] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[47] P. Panda, "Quanos: adversarial noise sensitivity driven hybrid quantization of neural networks," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, pp. 187–192.

[48] P. Panda, I. Chakraborty, and K. Roy, "Discretization based solutions for secure machine learning against adversarial attacks," *IEEE Access*, vol. 7, pp. 70 157–70 168, 2019.

[49] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[50] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[51] E. Park and S. Yoo, "Profit: A novel training method for sub-4-bit mobilenet models," *arXiv preprint arXiv:2008.04693*, 2020.

[52] Y.-F. Qin, R. Kuang, X.-D. Huang, Y. Li, J. Chen, and X.-S. Miao, "Design of high robustness bnn inference accelerator based on binary memristors," *IEEE Transactions on Electron Devices*, vol. 67, no. 8, pp. 3435–3441, 2020.

[53] A. S. Rakin, J. Yi, B. Gong, and D. Fan, "Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions," *arXiv preprint arXiv:1807.06714*, 2018.

[54] B. D. Rouhani, M. Samragh, M. Javaheripi, T. Javidi, and F. Koushanfar, "Deepfense: Online accelerated defense against adversarial deep learning," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[55] S. Ryu, H. Kim, W. Yi, and J.-J. Kim, "Bitblade: Area and energy-efficient precision-scalable neural network accelerator with bitwise summation," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[56] T. Rzayev, S. Moradi, D. H. Albonesi, and R. Manchar, "Deeprecon: Dynamically reconfigurable architecture for accelerating deep neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 116–124.

[57] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *arXiv preprint arXiv:1904.12843*, 2019.

[58] S. Sharify, A. D. Lascorz, K. Siu, P. Judd, and A. Moshovos, "Loom: Exploiting weight and activation precisions to accelerate convolutional neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[59] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 764–775.

[60] M. Shkolnik, B. Chmiel, R. Banner, G. Shomron, Y. Nahshan, A. Bronstein, and U. Weiser, "Robust quantization: One model to rule them all," *arXiv preprint arXiv:2002.07686*, 2020.

[61] C. Song, E. Fallon, and H. Li, "Improving adversarial robustness in weight-quantized neural networks," *arXiv preprint arXiv:2012.14965*, 2020.

[62] C. Song, Z. Wang, and H. Li, "Feedback learning for improving the robustness of neural networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 686–693.

[63] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[64] T. Strauss, M. Hanselmann, A. Junginger, and H. Ulmer, "Ensemble methods as a defense to adversarial perturbations against deep neural networks," *arXiv preprint arXiv:1709.03423*, 2017.

[65] Synopsys, "PrimeTime PX: Signoff power analysis," https://www.synopsys.com/support/training/signoff/primetimepx-fcd.html, accessed 2019-08-06.

[66] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[67] X. Wang, R. Hou, B. Zhao, F. Yuan, J. Zhang, D. Meng, and X. Qian, "Dnnguard: An elastic heterogeneous dnn accelerator architecture against adversarial attacks," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 19–34.

[68] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Learning Representations*, 2019.

[69] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[70] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 819–828.

[71] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[72] P. Xu, X. Zhang, C. Hao, Y. Zhao, Y. Zhang, Y. Wang, C. Li, Z. Guan, D. Chen, and Y. Lin, "Autodnnchip: An automated dnn chip predictor and builder for both fpgas and asics," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '20, 2020, p. 40–50.

[73] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[74] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," *arXiv preprint arXiv:1903.05662*, 2019.

[75] Y. Zhao, C. Li, Y. Wang, P. Xu, Y. Zhang, and Y. Lin, "Dnn-chip predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures," 2020.