Bit Error Robustness for Energy-Efficient DNN Accelerators

David Stutz[1] Nandhini Chandramoorthy[2] Matthias Hein[3] Bernt Schiele[1]
[1]Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken
[2]IBM T. J. Watson Research Center, NY [3]University of Tübingen, Tübingen
{dstutz,schiele}@mpi-inf.mpg.de, nandhini.chandramoorthy@ibm.com, matthias.hein@uni-tuebingen.de

Abstract

Deep neural network (DNN) accelerators received considerable attention in past years due to saved energy compared to mainstream hardware. Low-voltage operation of DNN accelerators allows to further reduce energy consumption significantly, however, causes bit-level failures in the memory storing the quantized DNN weights. In this paper, we show that a combination of ***robust fixed-point quantization***, ***weight clipping****, and* ***random bit error training (RandBET) improves robustness against random bit errors in (quantized) DNN weights*** *significantly. This leads to high energy savings from* both *low-voltage operation* as well as *low-precision quantization. Furthermore, our approach generalizes across operating voltages and accelerators, as demonstrated on bit errors from profiled SRAM arrays. Without losing more than* 1% *in accuracy, we can reduce energy consumption on CIFAR10 by* 20% *for a* 8*-bit quantized DNN. Higher energy savings of, e.g.,* 30%*, are possible at the cost of* 2.5% *accuracy, even for* 4*-bit DNNs.*

1. Introduction

Energy-efficiency is important to lower carbon-dioxide emissions of deep neural network (DNN) driven applications and to enable applications in edge computing. *DNN accelerators*, i.e., specialized hardware for inference, reduce energy consumption alongside cost and space compared to mainstream GPUs. These accelerators generally feature on-chip SRAM used as scratchpads, e.g., to store DNN weights. Data access/movement constitutes a dominant component of accelerator energy consumption [16]. Reduced precision [9] is one way to reduce energy consumption at the cost of *approximate computing* [13]. Similarly, recent DNN accelerators [12, 6, 1] lower memory supply voltage to increase energy efficiency since dynamic power varies quadratically with voltage. However, operating at very low voltages causes reliability issues in SRAMs in the form of bit-level failures [3, 4] with direct impact

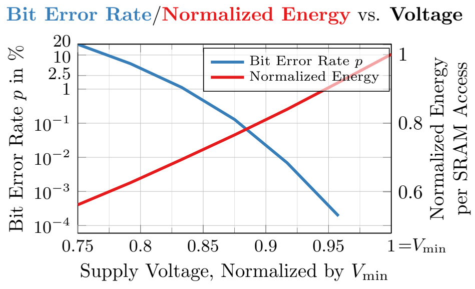
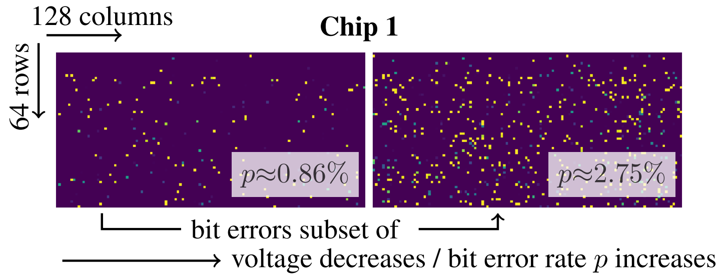


Figure 1: **Energy and Low-Voltage Operation.** Average bit error rate p (blue, left y-axis) from 32 14nm SRAM arrays of size 512×64 from [1] and energy (red, right y-axis) vs voltage (x-axis). Voltage is normalized by $V_{\min}$, the minimal measured voltage for error-free operation. Reducing voltage leads to exponentially increasing bit error rates.

on the stored DNN weights. The rate p of these errors increases exponentially with lowered voltage, causing devastating drops in DNN accuracy. In this paper, we aim to enable very low-voltage operation of DNN accelerators by developing DNNs robust to such bit errors in their weights, allowing DNN inference on *"approximate hardware"* [7].

Fig. 1 shows the average bit error rates of SRAM arrays as supply voltage is scaled below $V_{\min}$, i.e., the measured lowest voltage at which there are no bit errors. DNNs robust to a bit error rate (blue, left y-axis) of, e.g., $p = 1\%$ allow to reduce SRAM energy by roughly 30%. To improve DNN robustness to bit errors, we first consider the impact of fixed-point quantization on robustness. While prior work [11, 10, 15] studies robustness *to* quantization, we find that the choice of quantization scheme has tremendous impact on robustness, even though accuracy is not affected. We identify a particularly **robust quantization scheme** (RQuant in Fig. 4, red). Additionally, we propose aggressive **weight clipping** during training as regularization to improve robustness (Clipping in Fig. 4, blue). This is in contrast to, e.g., [19, 15] ignoring weight outliers to reduce quantization range, with sole focus of improving accuracy.

Common error correcting codes (ECCs such as SECDED), cannot correct *multiple* bit errors per word (containing multiple DNN weights). However, for $p = 1\%$,



Model (CIFAR10)	RErr in %, p in %	
***Fixed* Pattern**	p=1	p=2.5
PATTBET$_{0.15}$ p=2.5	8.50	7.41
***Random* Patterns**	p=1	p=2.5
PATTBET$_{0.15}$ p=2.5	12.09	61.59
RANDBET$_{0.15}$ p=1	8.63	64.97
RANDBET$_{0.025}$ p=2.5	7.83	8.63
***Profiled* Chip** (cf. left)	$p\approx$0.86	$p\approx$2.75
RANDBET$_{0.05}$ p=1.5	7.04	9.37

Figure 2: **Left: Exemplary SRAM Bit Error Patterns.** Measured bit errors from on-chip SRAM, showing bit flip probability for a segment of 64×128 bits: yellow indicates a bit flip probability of one, violet indicates zero probability. We show measurements corresponding to two supply voltages. **Right (top): Fixed Pattern Bit Error Training.** RErr for training on a fixed bit error pattern (PATTBET, in combination with RQUANT and CLIPPING) and evaluation on the same pattern and completely random patterns. PATTBET fails to generalize to lower bit error rates (in red), i.e., subsets of bit errors trained on, as well as random bit errors (i.e., other chips). [14] includes further comparison of PATTBET and RANDBET. **Right (bottom): Generalization to Profiled Bit Errors.** RErr for RANDBET on the profiled bit errors shown on the left. RANDBET generalizes well to the profiled bit errors, even though the pattern was not seen during training.

the probability of two or more bit errors in a 64-bit word is 13.5%. Error detection via redundancy [12] or supply voltage boosting [1] allow error-free low-voltage operation at the cost of additional energy or space. Therefore, [6, 7] propose co-design approaches of training DNNs on *profiled* SRAM/DRAM bit errors. These approaches work as the spatial bit error patterns can be assumed fixed for a *fixed* accelerator *and* voltage. However, the random nature of variation-induced bit errors requires profiling to be carried out for each voltage, memory array and individual chip making training DNNs on profiled bit error patterns an expensive process. More importantly, the obtained DNNs do *not* generalize across voltages or to unseen bit error patterns, e.g., from other memory arrays. We propose **random bit error training (RANDBET)** which, in combination with weight clipping and robust quantization, obtains robustness against completely *random* bit error patterns (Fig. 4, violet). Thereby, it generalizes across chips *and* voltages, without profiling, hardware-specific mapping or other circuit-level mitigation strategies.

This paper is a short version of our MLSys'20 work [14]. While it is intended to be self-contained, we refer to [14] for further discussion and results.

2. Low-Voltage Random Bit Errors

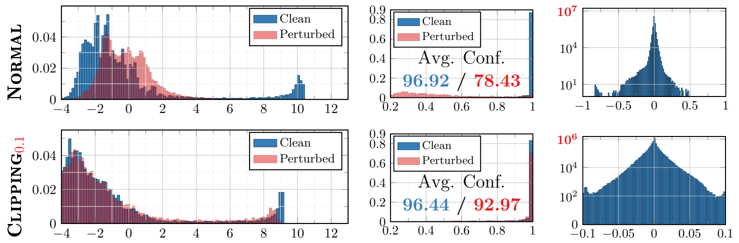
We assume the quantized DNN weights to be stored (linearly) on multiple memory banks, e.g., SRAM or DRAM. Following [3, 6, 1], the probability of memory bit cell failures increases exponentially as operating voltage is scaled below V_{min}, i.e., the minimal voltage required for reliable operation, cf. Fig. 1. This is done intentionally to reduce energy consumption, e.g., [1, 6, 7], or adversarially by an attacker, e.g., [17]. Process variation during fabrication causes a variation in the vulnerability of individual bit cells. For a specific memory array, bit cell failures are typically approximately random and independent of each other [3]. Nevertheless, there is generally an "inherited" distribution of bit cell failures across voltages: as described in [2], if a bit error occurred at a given voltage, it is likely to occur at lower voltages, cf. Fig. 2 (left). However, across different SRAM arrays or different chips, the patterns or spatial distribution of bit errors is usually different and can be assumed random [1]. We use the following bit error model:

Random Bit Error Model: *The probability of a bit error is p (in %) for all weight values and bits. For a fixed memory array, bit errors are persistent across supply voltages, i.e., bit errors at probability $p' \leq p$ also occur at probability p. A bit error flips the currently stored bit. We denote random bit error injection by $BErr_p$.*

This error model captures the nature of low-voltage induced bit errors, from both SRAM and DRAM [1, 6, 7]. However, our approach in Sec. 3 is model-agnostic: the error model can be refined if extensive memory characterization results are available for individual chips. However, estimating these specifics requires testing infrastructure and introduces the risk of overfitting. Furthermore, the robustness obtained using our uniform error model generalizes to profiled bit errors from real chips, cf. Fig. 2 (right).

3. Robustness Against Random Bit Errors

We address robustness against random bit errors in three steps: First, we analyze the impact of fixed-point quantization schemes on bit error robustness. This has been neglected both in prior work on low-voltage DNN accelerators [6, 7] and in work on quantization robustness [11, 10, 15]. This yields our **robust quantization**. On top, we propose aggressive **weight clipping** as regularization during training, enforcing a mor euniformly distributed, i.e., redundant, weight distribution. We argue that the redundancy is a result of limiting weight range while encouraging large logits by



	Model (CIFAR10) $\mathbf{w}_{\max}=\mathbf{0.1}$, p in %	Err in %	RErr in %	
			p=0.5	p=1.5
8bit	NORMAL	4.37	24.76±4.71	87.40±2.47
	RQUANT	**4.32**	11.28±1.47	68.65±9.23
	+CLIPPING	4.82	6.95±0.24	12.22±1.29
	+RANDBET p=1	4.90	**6.36±0.17**	**8.65±0.37**
4bit	CLIPPING	**5.29**	7.71±0.36	15.79±2.54
	+RANDBET p=1	5.39	**7.04±0.21**	**9.77±0.81**

Figure 3: **Left: Effect of Weight Clipping.** Weight clipping constraints the weights (right), thereby implicitly limiting the possible range for logits (left, blue). However, even for $w_{\max} = 0.1$ the DNN is able to produce high confidences (middle, blue), suggesting that more weights are used to obtain these logits. As result, the impact of random bit errors, $p = 1\%$, on the logits/confidences (red) is reduced. **Right: Random Bit Error Training (RANDBET).** Average RErr of RANDBET evaluated at bit error rates $p = 0.5\%$ and $p = 1.5\%$ using $m = 8$ or 4 bits. For low p, weight clipping provides sufficient robustness. For larger p, RANDBET increases robustness significantly.

minimizing the cross-entropy loss. Finally, in addition to robust quantization and weight clipping, we perform **random bit error training (RANDBET)**: in contrast to the fixed bit error patterns in [6, 7], we train on completely *random* bit errors and, generalize across chips and voltages.

3.1. Robust Fixed-Point Quantization

We consider quantization-aware training using a simple fixed-point quantization scheme commonly used in DNN accelerators [1]: However, we focus on the impact of quantization schemes on robustness against random bit errors, mostly neglected so far [11, 10, 15]. Let $f(x; w)$ be a DNN taking an example $x \in [0, 1]^D$, e.g., an image, and weights $w \in \mathbb{R}^W$ as input. Quantization determines how weights are represented in memory, e.g., on SRAM. In a *fixed-point quantization* scheme, m bits allow to represent 2^m distinct values. A weight $w_i \in [-q_{\max}, q_{\max}]$ is represented by a signed m-bit integer $v_i = Q(w_i)$ corresponding to the underlying bits. Here, $[-q_{\max}, q_{\max}]$ is the *symmetric* quantization range and signed integers use two's complement representation. Then, $Q : [-q_{\max}, q_{\max}] \mapsto \{-2^{m-1} - 1, \dots, 2^{m-1} - 1\}$ is defined as

$$Q(w_i) = \left\lfloor \frac{w_i}{\Delta} \right\rfloor,\; Q^{-1}(v_i) = \Delta v_i,\; \Delta = \frac{q_{\max}}{2^{m-1} - 1} \qquad (1)$$

This quantization is symmetric around zero and zero is represented exactly. Note that we consider quantizing weights only. In *global* quantization, $q_{\max}$ is chosen to accommodate all weights, i.e., $q_{\max} = \max_i |w_i|$. However, it has become standard to apply quantization *per-layer* allowing to adapt $q_{\max}$ to each layer. **Per-layer, symmetric quantization is our default**, referred to as NORMAL.

To further reduce quantization error, we also consider arbitrary quantization ranges $[q_{\min}, q_{\max}]$ (allowing $q_{\min} > 0$): we map $[q_{\min}, q_{\max}]$ to $[-1, 1]$ and quantize $[-1, 1]$ as above. The resulting per-layer *asymmetric* quantization has the finest granularity (i.e., lowest Δ), however, is not the most robust. Therefore, we further replace the floor operation $\lfloor {}^{w_i}/_{\Delta} \rfloor$ with proper rounding $\lceil {}^{w_i}/_{\Delta} \rfloor$. Similarly, for asymmetric quantization, we use quantization into *unsigned* integers, i.e., $Q : [q_{\min}, q_{\max}] \mapsto \{0, \dots, 2^m - 1\}$, instead. DNNs are able to "learn around" these implementation details (i.e., the crude floor operation or asymmetric quantization into *signed* integers) when optimizing accuracy. However, regarding bit erro robustness, these details make a difference, see [14] for a detailed discussion. This means that these differences have little to no impact on accuracy, while having tremendous impact on robustness against bit errors.

3.2. Weight Clipping

Weight clipping refers to constraining the weights to $[-w_{\max}, w_{\max}]$ *during training*, where $w_{\max}$ is a hyperparameter. Generally, $w_{\max}$ is independent of the quantization range(s) which always adapt(s) to the weight range(s) at hand. However, weight clipping limits the maximum possible quantization range, i.e., $q_{\max} \leq w_{\max}$. Note that the *relative* errors induced by bit errors do *not* change through weight clipping. As the DNN's decision is usually invariant to rescaling, reducing the scale of the weights does not impact robustness. In fact, we found the mean relative error of the weights to increase with clipping, e.g., at $w_{\max} = 0.1$. Thus, weight clipping does *not* "trivially" improve robustness by reducing the scale of weights. Instead, we found that the interplay of weight clipping and minimizing the the cross-entropy loss during training is the key. High confidences can only be achieved by large differences in the logits. Because the weights are limited to $[-w_{\max}, w_{\max}]$, large logits can only be achieved using more weights in each layer to produce larger outputs. As a result, weight clipping leads to more weights being utilized, i.e., more redundancy in the weights, making them less susceptible to (bit) errors, as illustrated in Fig. 3 (left). Also, weight clipping is easy to implement, adds negligible training cost and does not interfer with other regularizers such as weight decay.

3.3. Random Bit Error Training (RANDBET)

In *addition to* weight clipping and robust quantization, we inject random bit errors with probability p during training to further improve robustness. This results in the fol-

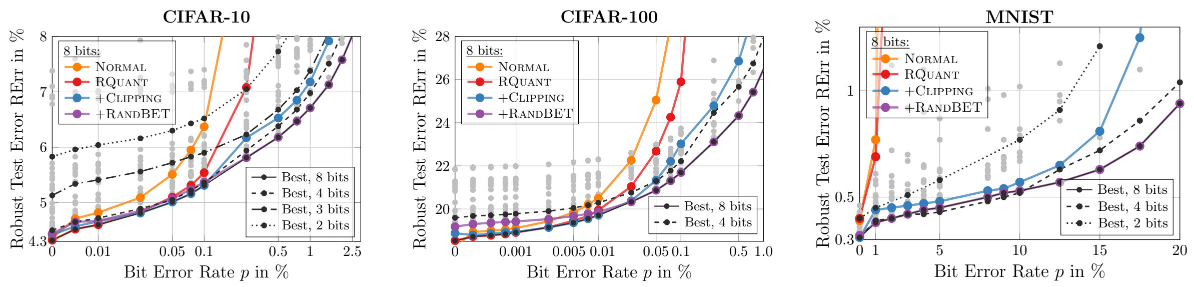


Figure 4: **Robustness to Random Bit Errors.** Robust test error (test error *after* injecting bit errors in the quantized weights, RErr, ↓, y-axis) plotted against bit error rate p (x-axis) for our robust fixed-point quantization (RQuant, orange), weight clipping (Clipping, blue) and random bit error training (RandBET, violet). In each case, we highlight the best model for each bit error rate. Additionally, we report the overall best model per bit error rate for various bit error rates, e.g., $m = 8, 4, 3$ and 2 on CIFAR10. For 8 bit and low bit error rates, Clipping is often sufficient. However, for 4 bit or higher bit error rates, RandBET is crucial to keep RErr low.

lowing learning problem:

$$\begin{aligned} &\min_w \mathbb{E}[\mathcal{L}(f(x;\tilde{w}),y) + \mathcal{L}(f(x;w),y)] \\ &\text{s.t.} \quad v = Q(w),\ \tilde{v} = \text{BErr}_p(v),\ \tilde{w} = Q^{-1}(\tilde{v}). \end{aligned} \tag{2}$$

where (x, y) are labeled examples, $\mathcal{L}$ is the cross-entropy loss and $v = Q(w)$ denotes the (element-wise) quantized weights w which are to be learned. $\text{BErr}_p(v)$ injects random bit errors with rate p in v. Note that we consider both the loss on clean weights and weights with bit errors to avoid an increase in (clean) test error and stabilizes training. Note that bit error rate p implies, in expectation, pmW bit errors. We use stochastic gradient descent to optimize Eq. (2), by performing the gradient computation using the perturbed weights $\tilde{w} = Q^{-1}(\tilde{v})$ with $\tilde{v} = \text{BErr}_p(v)$, while applying the gradient update on the (floating-point) clean weights w.

4. Experiments

We conduct experiments on MNIST and CIFAR [8] and report (clean) test error Err (lower is better, ↓), corresponding to *clean* weights, and **robust test error RErr** (↓), i.e., the **test error *after* injecting bit errors into the weights**. We report *average* RErr across 50 samples of random bit errors for a specific rate p. We use SimpleNet [5] on CIFAR10 and Wide ResNet (WRN) [18] on CIFAR100. Normal training with the standard and our robust quantization are denoted Normal and RQuant, respectively. Weight clipping with w_{max} is referred to as $\text{Clipping}_{w_{\text{max}}}$ or together with RandBET as $\text{RandBET}_{w_{\text{max}}}$.

Fig. 3 (right) presents RErr on CIFAR10 for $m = 8$ and $m = 4$ bit, showing that our combination of RQuant, Clipping and RandBET (trained with $p = 1\%$ bit error rate) improves robustness to random bit errors significantly, especially for high bit error rates. For smaller bit error rates, e.g., $p = 0.5\%$, Clipping with $w_{\text{max}} = 0.1$ might be sufficient for robust operation, achieving 6.95%RErr, while RandBET is necessary at higher bit error rates, e.g., $p = 1.5\%$. For lower precision, i.e., $m = 4$ bits, the benefit of RandBET is pronounced even further, reducing RErr significantly from 15.79% to 9.77% for $p = 1.5\%$. We also emphasize that RandBET generalize to lower bit errors than trained on. This is in contrast to related work [6, 7], training on fixed bit error patterns (e.g., profiled) as demonstrated in Fig. 2 (right). RandBET also generalizes to bit errors profiled from real chips, see Fig. 2 (right).

Our experiments are summarized in Fig. 4 (right), plotting RErr against bit error rate p for various Clipping and RandBET models corresponding to different w_{max}/p (indicated in ● gray) in comparison to Normal and RQuant. RQuant (red) clearly outperforms Normal (orange), however, RErr increases quickly even for low bit error rates. Clipping (blue) generally reduces RErr, but only the combination with RandBET (violet) can keep RErr around 6% for a bit error rate of $p \approx 0.5\%$ on CIFAR10. This corresponds to roughly 25% energy savings in Fig. 1 (left). The best model for each bit error rate p and different precisions m is shown in black (e.g., solid for $m = 8$ or dashed for $m = 4$). Even for $m = 4$ bits precision, RandBET ensures low RErr. This enables energy savings from *both* low-voltage operation *and* low precision quantization.

5. Conclusion

Overall, the proposed combination of **robust quantization**, **weight clipping** and **random bit error training (RandBET)** enables robust low-voltage operation *without* requiring expensive error correcting codes (ECCs) or other circuit techniques [12, 1]. Furthermore, our analysis applies both to DRAM, commonly off-chip, and SRAM, usually used as scratchpads on-chip of DNN accelerators. Compared to co-design approaches [6, 7], we do not require expert knowledge or expensive profiling infrastructure. Moreover, RandBET improves over these approaches by generalizing across chips *and* voltages. We also show that robust fixed-point quantization *only with* weight clipping can provide reasonable robustness. Finally, to further reduce energy consumption, our approach also enables low-voltage operation at low precisions, e.g., 4 bits or lower.

References

[1] Nandhini Chandramoorthy, Karthik Swaminathan, Martin Cochet, Arun Paidimarri, Schuyler Eldridge, Rajiv V. Joshi, Matthew M. Ziegler, Alper Buyuktosunoglu, and Pradip Bose. Resilient low voltage accelerators for high energy efficiency. In *HPCA*, 2019. 1, 2, 3, 4

[2] Shrikanth Ganapathy, John Kalamatianos, Bradford M. Beckmann, Steven Raasch, and Lukasz G. Szafaryn. Killi: Runtime fault classification to deploy low voltage caches without MBIST. In *HPCA*, 2019. 2

[3] Shrikanth Ganapathy, John Kalamatianos, Keith Kasprak, and Steven Raasch. On characterizing near-threshold SRAM failures in FinFET technology. In *DAC*, 2017. 1, 2

[4] Zheng Guo, Andrew Carlson, Liang-Teck Pang, Kenneth Duong, Tsu-Jae King Liu, and Borivoje Nikolic. Large-scale SRAM variability characterization in 45 nm CMOS. *JSSC*, 44(11), 2009. 1

[5] Seyyed Hossein HasanPour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv.org*, abs/1608.06037, 2016. 4

[6] Sung Kim, Patrick Howe, Thierry Moreau, Armin Alaghi, Luis Ceze, and Visvesh Sathe. MATIC: learning around errors for efficient low-voltage neural network accelerators. In *DATE*, 2018. 1, 2, 3, 4

[7] Skanda Koppula, Lois Orosa, Abdullah Giray Yaglikçi, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu. EDEN: enabling energy-efficient, high-performance deep neural network inference using approximate DRAM. In *MICRO*, pages 166–181, 2019. 1, 2, 3, 4

[8] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 4

[9] Darryl Dexu Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *ICML*, 2016. 1

[10] Paul Merolla, Rathinakumar Appuswamy, John V. Arthur, Steven K. Esser, and Dharmendra S. Modha. Deep neural networks are robust to weight binarization and other nonlinear distortions. *arXiv.org*, abs/1606.01981, 2016. 1, 2, 3

[11] Abhishek Murthy, Himel Das, and Md. Ariful Islam. Robustness of neural networks to parameter quantization. *arXiv.org*, abs/1903.10672, 2019. 1, 2, 3

[12] Brandon Reagen, Paul N. Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David M. Brooks. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *ISCA*, 2016. 1, 2, 4

[13] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. Enerj: Approximate data types for safe and general low-power computation. *SIGPLAN Not.*, 46(6), 2011. 1

[14] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Bit error robustness for energy-efficient dnn accelerators. In *MLSys*, 2021. 2, 3

[15] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *arXiv.org*, abs/1511.06488, 2015. 1, 2, 3

[16] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *IEEE*, 105(12), 2017. 1

[17] Adrian Tang, Simha Sethumadhavan, and Salvatore J. Stolfo. CLKSCREW: exposing the perils of security-oblivious energy management. In *USENIX*, 2017. 2

[18] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 4

[19] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian D. Reid. Towards effective low-bitwidth convolutional neural networks. In *CVPR*, 2018. 1