BIT ERROR ROBUSTNESS FOR ENERGY-EFFICIENT DNN ACCELERATORS

David Stutz [1] **Nandhini Chandramoorthy** [2] **Matthias Hein** [3] **Bernt Schiele** [1]

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. Our approach generalizes across operating voltages and accelerators, as demonstrated on bit errors from profiled SRAM arrays. We also discuss why weight clipping alone is already a quite effective way to achieve robustness against bit errors. Moreover, we specifically discuss the involved trade-offs regarding accuracy, robustness and precision: Without losing more than 1% in accuracy compared to a normally trained 8-bit DNN, we can reduce energy consumption on CIFAR10 by 20%. 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 an important goal to lower carbon-dioxide emissions of deep neural network (DNN) driven applications and is a critical prerequisite to enable applications in edge computing. *DNN accelerators*, i.e., specialized hardware for inference, are used to reduce and limit energy consumption alongside cost and space compared to mainstream hardware, e.g., 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 (Sze et al., 2017). Reduced precision (Lin et al., 2016) is a widely used measure to reduce energy consumption at the cost of *approximate computing* (Sampson et al., 2011). Recently, DNN accelerators (Reagen et al., 2016; Kim et al., 2018; Chandramoorthy et al., 2019) further lower memory supply voltage to increase energy efficiency since dynamic power varies quadratically with voltage. However, aggressive SRAM supply voltage scaling, causes bit-level failures in SRAM on account of process variation (Ganapathy et al., 2017; Guo et al., 2009) with direct impact on the stored DNN weights. The rate p of these errors increases exponentially with lowered voltage and causes devastating drops in DNN accuracy such that memory reliability becomes the bottleneck in realizing low power DNN accelerators. 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"* (Koppula et al., 2019; Sampson et al., 2011). This is also desirable to improve security against adversarial manipulation of voltage settings (Tang et al., 2017). In general, robustness to bit errors in DNNs is a desirable goal in order to maintain safe operation and should become a standard performance metric in low power DNN design.

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. Voltage (x-axis) and energy (red, right y-axis) are normalized wrt. $V_{\min}$ and the energy per access at $V_{\min}$, respectively. 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 (Murthy et al., 2019; Merolla et al., 2016; Sung et al., 2015) studies robustness *to* quantization, the impact of random bit errors *in* quantized weights has not been considered so far. We find that the choice of quantization scheme has tremendous impact on robustness, even though accuracy is not affected. In particular, we identify a particularly **robust quantization scheme**, RQUANT in Fig. 2 (red). Additionally, independent of the quantization scheme, we propose aggressive **weight clipping** during training. This acts as an explicit regularizer leading to spread out weight distributions, improving robustness significantly, CLIPPING in Fig. 2 (blue). This is in contrast to, e.g., (Zhuang et al.,

[1]Max Planck Institute for Informatics [2]IBM T. J. Watson Research Center [3]University of Tübingen. Correspondence to: David Stutz <david.stutz@mpi-inf.mpg.de>.

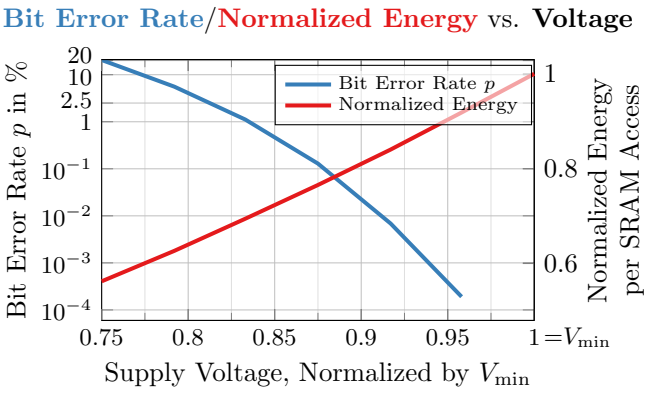

Figure 1: **Energy and Low-Voltage Operation.** Average bit error rate p (blue, left y-axis) from 32 14nm SRAM arrays of size 512×64 from (Chandramoorthy et al., 2019) and energy (red, right y-axis) vs. voltage (x-axis). Voltage is normalized by V_{min}, the minimal measured voltage for error-free operation, as well as the energy per SRAM access at V_{min}. SRAM accesses have significant impact on the DNN accelerator's energy (Chen et al., 2016). Reducing voltage leads to exponentially increasing bit error rates.

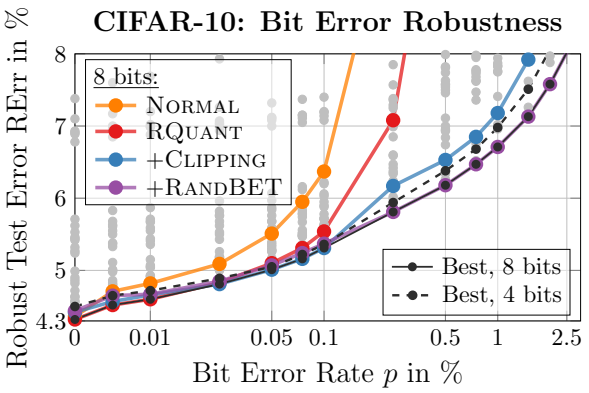

Figure 2: **Robustness to Random Bit Errors.** Robust test error (test error *after* injecting bit errors, RErr, lower is better ↓, y-axis) plotted against bit error rate p (x-axis). Robustness to higher bit error rates allows more energy efficient operation, cf. Fig. 1. For 8 bit, through robust quantization (RQUANT, red), additionally weight clipping (CLIPPING, blue) and finally adding random bit error training (RANDBET, violet) robustness improves significantly. The Pareto optimal frontier is shown for 8 bit (black solid) and 4 bit (dashed) quantization.

2018; Sung et al., 2015) 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 $p = 1\%$, the probability of two or more bit errors in a 64-bit word is 13.5%. Error detection via redundancy (Reagen et al., 2016) or supply voltage boosting (Chandramoorthy et al., 2019) allow error-free low-voltage operation at the cost of additional energy or space. Therefore, (Kim et al., 2018) proposes a co-design approach of training DNNs on *profiled* SRAM bit errors. Similarly, for approximate DRAMs, (Koppula et al., 2019) combines profiled bit error training with a clever weight to DRAM mapping. These approaches work as the spatial bit error patterns can be assumed fixed for a *fixed* accelerator *and* voltage. The bit error pattern is obtained by post-silicon profiling and characterization of memories. The random nature of variation-induced bit errors requires profiling to be carried out for each voltage, memory array and individual chip in order to obtain the corresponding bit error patterns. This makes training DNNs on profiled bit error patterns an expensive process. More importantly, we demonstrate that 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, see Fig. 2 (violet). Thereby, it generalizes across chips *and* voltages, without any profiling, hardware-specific data mapping or other circuit-level mitigation strategies.

Contributions: We combine our **robust fixed-point quantization RQUANT**, i.e., reduced quantization range and robust implementation, with **weight clipping** and **random bit error training (RANDBET)** in order to obtain high robustness against low-voltage induced, random bit errors. We consider fixed-point quantization schemes in terms of robustness *and* accuracy, instead of *solely* focusing on accuracy as related work. Furthermore, we show that aggressive weight clipping, as regularization during training, is an effective strategy to improve robustness through redundancy. In contrast to (Kim et al., 2018; Koppula et al., 2019), the robustness obtained through RANDBET generalizes across chips *and* voltages, as evaluated on profiled SRAM bit error patterns from (Chandramoorthy et al., 2019). Finally, we discuss the involved trade-offs regarding robustness and accuracy and make our code publicly available to facilitate research in this highly applicable area of DNN robustness. Fig. 2 highlights key results on CIFAR10: with 8 bit and an increase in test error of less than 1%, roughly 20% energy savings are possible. Combined with low-precision, e.g., for 4 bit quantization, 30% energy savings are possible at $p = 1\%$ with an increase in error rate of less than 2.5%.

Outline: We review related work in Sec. 2 and provide a detailed description and discussion of the considered low-voltage bit error model in Sec. 3. In Sec. 4, we discuss fixed-point quantization and its influence on bit error robustness and present weight clipping and RANDBET as effective strategies to improve robustness. Finally, Sec. 5 includes our experimental results. We conclude in Sec. 6.

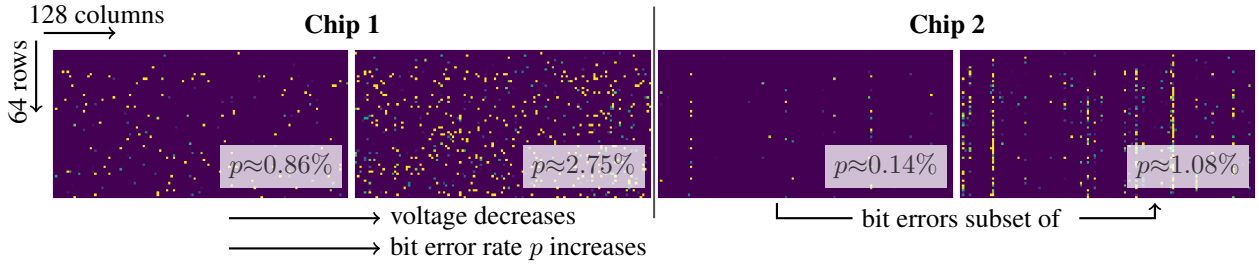

Figure 3: **Exemplary SRAM Bit Error Patterns.** Measured bit errors from two chips with on-chip SRAM (left and right), showing bit flip probability for a segment of size 64×128 bits: yellow indicates a bit flip probability of one, violet indicates zero probability. We show measurements corresponding to two supply voltages. With lower voltage, bit error rate increases. Also, the bit errors for higher voltage (= lower bit error rate) are a subset of those for lower voltage (= higher rate), cf. Sec. 3. Our error model randomly distributes bit errors across space. However, as example, we also show SRAM chip 2 which has a different spatial distribution with bit errors distributed along columns. We aim to obtain robustness across different memory arrays, voltages *and* allowing arbitrary DNN weight to memory mappings.

2 Related Work

We review relevant prior work on quantization, low-voltage induced random bit errors and weight robustness, in general.

Quantization: DNN Quantization (Guo, 2018) is usually motivated by faster DNN inference, e.g., through fixed-point quantization and arithmetic (Shin et al., 2017; Lin et al., 2016; Li et al., 2017), and energy savings. To avoid reduced accuracy, quantization is considered during training (Jacob et al., 2018; Krishnamoorthi, 2018) instead of post-training or with fine-tuning (Goncharenko et al., 2018; Banner et al., 2019; nvt; ner), enabling low-bit quantization such as binary DNNs (Rastegari et al., 2016; Courbariaux et al., 2015). Some works also consider quantizing activations (Rastegari et al., 2016; Choi et al., 2018; Hubara et al., 2017) or gradients (Seide et al., 2014; Alistarh et al., 2016; Zhou et al., 2016). While works such as (Murthy et al., 2019; Merolla et al., 2016; Sung et al., 2015; Alizadeh et al., 2020) study the robustness of DNNs *to* quantization, the robustness of various quantization schemes *against* random bit errors has not been studied. This is in stark contrast to our findings that quantization impacts robustness significantly. Furthermore, works such as (Zhuang et al., 2018; Sung et al., 2015; Park et al., 2018) clip weight outliers to reduce approximation error of inliers, improving accuracy. In contrast, we consider *weight clipping* independent of quantization *as regularization during training* which spreads out the weight distribution and improves robustness to bit errors.

Bit Errors in DNN Accelerators: Recent work (Ganapathy et al., 2017; 2019) demonstrates that bit flips in SRAMs increase exponentially when reducing voltage below V_{min}. The authors of (Chandramoorthy et al., 2019) study the impact of bit flips in different layers of DNNs, showing severe accuracy degradation. Similar observations hold for DRAM (Chang et al., 2017). To prevent accuracy drops at low voltages, (Reagen et al., 2016) combines SRAM fault detection with logic to set faulty data reads to zero. (Chandramoorthy et al., 2019) uses supply voltage boosting for SRAMs to ensure error-free, low-voltage operation, while (Srinivasan et al., 2016) proposes storing critical bits in specifically robust SRAM cells. However, such methods incur power and area overhead. Thus, (Kim et al., 2018) and (Koppula et al., 2019) propose co-design approaches combining training on profiled SRAM/DRAM bit errors with hardware mitigation strategies and clever weight to memory mapping. Besides low-voltage operation for energy efficiency, recent work (Tang et al., 2017) shows that an attacker can reduce voltage maliciously. Similarly, works such as (Kim et al., 2014; Murdock et al., 2020) demonstrate software-based approaches to induce few, but targeted, bit flips in DRAM. In contrast to (Kim et al., 2018; Koppula et al., 2019), our *random bit error training* obtains robustness that generalizes across chips and voltages without expensive chip-specific profiling or hardware mitigation strategies. Furthermore, (Kim et al., 2018; Koppula et al., 2019) do not address the role of quantization and we demonstrate that these approaches can benefit from our weight clipping, as well. We show that energy savings from low-voltage operation and low-precision (Park et al., 2018) can be combined.

Weight Robustness: Only few works consider weight robustness: (Weng et al., 2020) certify the robustness of weights with respect to L_∞ perturbations and (Cheney et al., 2017) study Gaussian noise on weights. (Rakin et al., 2019; He et al., 2020) consider identifying and (adversarially) flipping few vulnerable bits in quantized weights. Fault tolerance, in contrast, describes structural changes such as removed units, and is rooted in early work such as (Neti et al., 1992; Chiu et al., 1994). Finally, (Ji et al., 2018; Dumford & Scheirer, 2018) explicitly manipulate weights in order to integrate backdoors. We study robustness against *random bit errors*, which exhibit a quite special noise pattern compared to L_∞ or Gaussian noise, cf. Fig. 4.

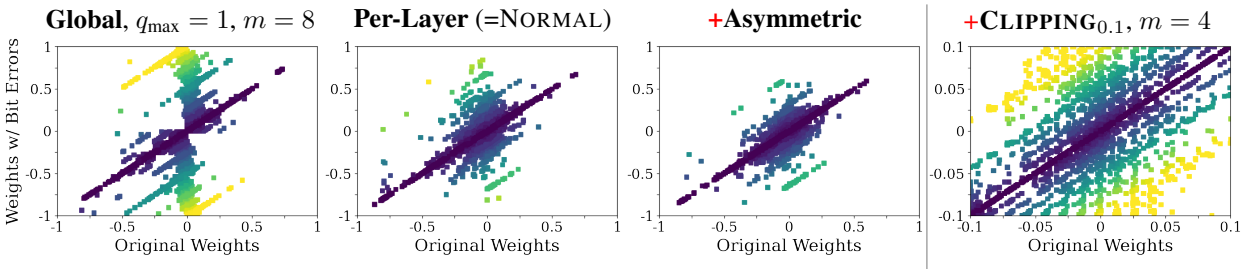

Figure 4: **Quantization and Random Bit Errors.** Original weights (x-axis) plotted against perturbed weights with bit errors (y-axis), for different fixed-point quantization schemes with $m = 8$ bit (left) and $p = 2.5\%$. We also show the $m = 4$ bit case with CLIPPING at $w_{max} = 0.1$, cf. Sec. 4.2. Color indicates absolute error: from zero violet to the maximal possible error yellow of 1 (left) and 0.1 (right). Asymmetric per-layer quantization reduces the impact of bit errors compared to a the symmetric per-layer/global quantization. Clipping reduces absolute error, but the errors *relative* to w_{max} increase.

3 Low-Voltage Induced Random Bit Errors in Quantized DNN Weights

We assume the quantized DNN weights to be stored on multiple memory banks, e.g., SRAM in the case of on-chip scratchpads or DRAM for off-chip memory. As shown in (Ganapathy et al., 2017; Kim et al., 2018; Chandramoorthy et al., 2019), 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, see Fig. 1. This is done intentionally to reduce energy consumption, e.g., (Chandramoorthy et al., 2019; Kim et al., 2018; Koppula et al., 2019), or adversarially by an attacker, e.g., (Tang et al., 2017). Process variation during fabrication causes a variation in the vulnerability of individual bit cells. As shown in Fig. 3 (left), for a specific memory array, bit cell failures are typically approximately random and independent of each other (Ganapathy et al., 2017) even so chips showing patterns with stronger dependencies are possible Fig. 3 (right). Nevertheless, there is generally an "inherited" distribution of bit cell failures across voltages: as described in (Ganapathy et al., 2019), if a bit error occurred at a given voltage, it is likely to occur at lower voltages, as made explicit in Fig. 3. However, across different SRAM arrays in a chip or different chips, the patterns or spatial distribution of bit errors is usually different and can be assumed random (Chandramoorthy et al., 2019). Throughout the paper, 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 realistically captures the nature of low-voltage induced bit errors, from both SRAM and DRAM as confirmed in (Chandramoorthy et al., 2019; Kim et al., 2018; Koppula et al., 2019). However, our approach in Sec. 4 is model-agnostic: the error model can be refined if extensive memory characterization results are available for individual chips. For example, faulty bit cells with 1-to-0 or 0-to-1 flips might not be equally likely. Similarly, as in (Koppula et al., 2019), bit errors might be biased towards alignment along rows or columns of the memory array. The latter case is illustrated in Fig. 3 (right). However, estimating these specifics requires testing infrastructure and detailed characterization of individual chips. More importantly, it introduces the risk of overfitting to few specific memories/chips. Furthermore, we demonstrate that the robustness obtained using our uniform error model generalizes to bit error distributions with strong spatial biases as in Fig. 3 (right).

We assume the quantized weights to be mapped linearly to the memory. This is the most direct approach and, in contrast to (Koppula et al., 2019), does not require knowledge of the exact spatial distribution of bit errors. This also means that we do not map particularly vulnerable weights to more reliable memory cells, and therefore no changes to the hardware or the application are required. Thus, in practice, for W weights and m bits per weight value, we sample uniformly $u \sim U(0,1)^{W \times m}$. Then, the j-th bit in the quantized weight $v_i = Q(w_i)$ is flipped iff $u_{ij} \leq p$. Our model assumes that the flipped bits at lower probability $p' \leq p$ are a subset of the flipped bits at probability p and that bit flips to 1 and 0 are equally likely. The noise pattern of random bit errors is illustrated in Fig. 4: for example a single bit flip in the most-significant bit (MSB) of the signed integer v_i can result in a change of roughly half of the quantized range (also cf. Sec. 4.1).

4 Towards 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 acceler-

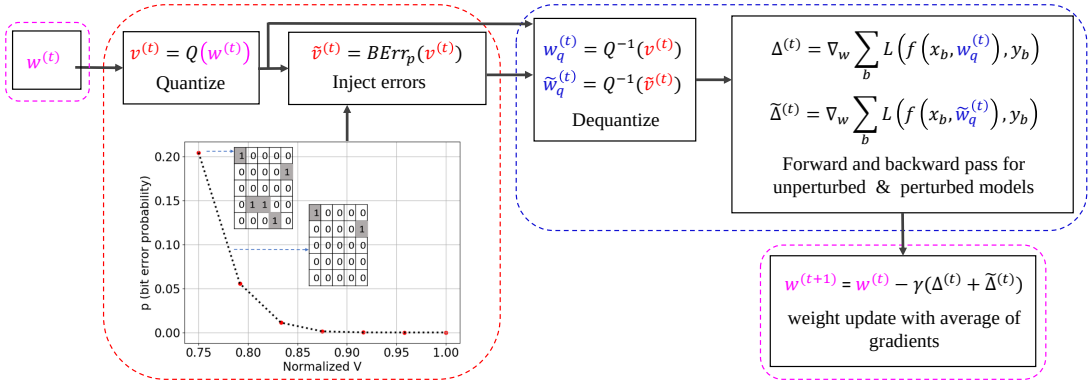

Figure 5: **Random Bit Error Training (RandBET).** We illustrate the data-flow for RandBET as in Alg. 1. Here, BErr_p injects random bit errors in the quantized weights $v^{(t)} = Q(w^{(t)})$, resulting in $\tilde{v}^{(t)}$, while the forward pass is performed on the de-quantized perturbed weights $\tilde{w}_q^{(t)} = Q^{-1}(\tilde{v}^{(t)})$, i.e., fixed-point arithmetic is not emulated. The weight update during training is not affected by bit errors and computed in floating point.

ators (Kim et al., 2018; Koppula et al., 2019) and in work on quantization robustness (Murthy et al., 2019; Merolla et al., 2016; Sung et al., 2015). This yields our **robust quantization** (Sec. 4.1). On top, we propose aggressive **weight clipping** as regularization during training (Sec. 4.2). Weight clipping enforces a more uniformly distributed, i.e., redundant, weight distribution, improving robustness. We show that this is due to minimizing the cross-entropy loss, enforcing large logit differences. Finally, in addition to robust quantization and weight clipping, we perform **random bit error training (RandBET)** (Sec. 4.3): in contrast to the fixed bit error patterns in (Kim et al., 2018; Koppula et al., 2019), we train on completely *random* bit errors and, thus, generalize across chips and voltages. Generalization is measured using *average robust test error (RErr)*, the test error after injecting bit errors, wrt. to our error model from Sec. 3 as well as real, profiled bit error patterns. Robustness against bit error rate p has to induce robustness for $p' \leq p$ (i.e., higher voltage), as well.

4.1 Robust Fixed-Point Quantization

We consider quantization-aware training (Jacob et al., 2018; Krishnamoorthi, 2018) using a generic, deterministic fixed-point quantization scheme commonly used in DNN accelerators (Chandramoorthy et al., 2019). However, we focus on the impact of quantization schemes on robustness against random bit errors, mostly neglected so far (Murthy et al., 2019; Merolla et al., 2016; Sung et al., 2015). We find that quantization affects robustness significantly, even if accuracy is largely unaffected.

Fixed-Point Quantization: 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

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

Flipping the most significant bit (MSB, i.e., sign bit) leads to an absolute error of half the quantization range, i.e., $q_{\max}$ (yellow in Fig. 4). Flipping the least significant bit (LSB) incurs an error of Δ, cf. Eq. (1). Thus, the impact of bit errors "scales with" $q_{\max}$.

Global and Per-Layer Quantization: $q_{\max}$ can be chosen to accommodate all weights, i.e., $q_{\max} = \max_i |w_i|$. This is called *global* quantization. However, it has become standard to apply quantization *per-layer* allowing to adapt $q_{\max}$ to each layer. As in PyTorch (Paszke et al., 2017), we consider weights and biases of each layer separately. By reducing the quantization range for each layer individually, the errors incurred by bit flips are automatically minimized, cf. Fig. 4. The **per-layer, symmetric quantization is our default reference**, referred to as Normal. However, it turns out that it is further beneficial to consider arbitrary quantization ranges $[q_{\min}, q_{\max}]$ (allowing $q_{\min} > 0$). In practice, we first map $[q_{\min}, q_{\max}]$ to $[-1, 1]$ and then quantize $[-1, 1]$ using Eq. (1). Overall, per-layer asymmetric quantization has the finest granularity, i.e., lowest Δ and approximation error. Nevertheless it is not the most robust quantization.

Robust Quantization: Quantization as in Eq. (1) does *not* provide optimal robustness against bit errors. First, the floor operation $\lfloor w_i/\Delta \rfloor$ is commonly implemented as float-to-integer conversion. Using proper rounding $\lceil w_i/\Delta \rfloor$ instead has negligible impact on accuracy, even though approximation error improves slightly. In stark contrast, bit

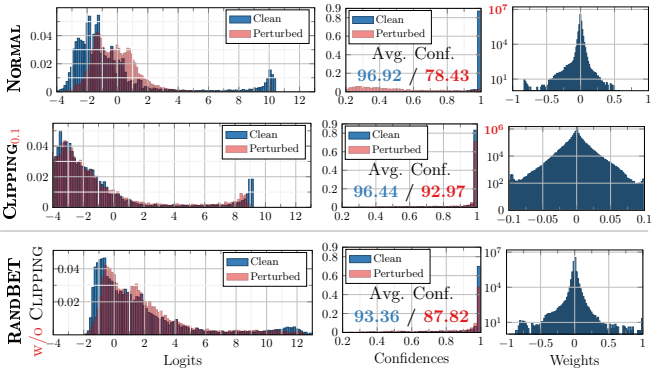

Figure 6: **Effect of Weight Clipping.** On CIFAR10, weight clipping constraints the weights (right), thereby implicitly limiting the possible range for logits (left, blue). However, even for $w_{\max} = 0.1$ the DNN is able to produce high confidences (middle, blue), suggesting that more weights are used to obtain these logits. Furthermore, the impact of random bit errors, $p = 1\%$, on the logits/confidences (red) is reduced significantly. RandBET (trained with $p = 1\%$, w/o weight clipping), increases the range of weights and is less effective at preserving logit/confidence distribution.

Algorithm 1 Random Bit Error Training (RandBET). The forward passes are performed using de-quantized weights (blue). Perturbed weights are obtained by injecting bit errors in the quantized weights (in red). The update, averaging gradients from both forward passes, is performed in floating-point (magenta). Also see Fig. 5.

procedure RandBET(p)
  initialize $w^{(0)}$
  for $t = 0, \dots, T-1$ **do**
    sample batch $\{(x_b, y_b)\}_{b=1}^{B}$
    {element-wise clipping:}
    $w^{(t)} = \min(w_{\max}, \max(-w_{\max}, w^{(t)}))$
    {quantization:}
    $v^{(t)} = Q(w^{(t)})$
    $w_q^{(t)} = Q^{-1}(v^{(t)})$
    {clean forward and backward pass:}
    $\Delta^{(t)} = \nabla_w \sum_{b=1}^{B} \mathcal{L}(f(x_b; w_q^{(t)}), y_b)$
    {*perturbed* forward and backward pass:}
    $\tilde{w}_q^{(t)} = Q^{-1}(\text{BErr}_p(v^{(t)}))$ {inject random bit errors}
    $\tilde{\Delta}^{(t)} = \nabla_w \sum_{b=1}^{B} \mathcal{L}(f(x_b; \tilde{w}_q^{(t)}), y_b)$
    {average gradients and weight update:}
    $w^{(t+1)} = w^{(t)} - \gamma(\Delta^{(t)} + \tilde{\Delta}^{(t)})$
  return $w_q^{(T)} = Q^{-1}(Q(w^{(T)}))$

error robustness is improved considerably. During training, DNNs can compensate the differences in approximation errors, even for small precision $m < 8$. However, at test time, rounding decreases the impact of bit errors considerably. Second, Eq. (1) uses signed integers for symmetric quantization. For asymmetric quantization, with arbitrary $[q_{\min}, q_{\max}]$, we found quantization into *unsigned* integers to improve robustness, i.e., $Q : [q_{\min}, q_{\max}] \mapsto \{0, \dots, 2^m - 1\}$. This is implemented using an additive term of $2^{m-1} - 1$ in Eq. (1). While accuracy is not affected, the effect of bit errors in the sign bit changes: in symmetric quantization, the sign bit mirrors the sign of the weight value. For asymmetric quantization, an unsigned integer representation is more meaningful. Overall, our **robust fixed-point quantization (RQuant)** uses per-layer, asymmetric quantization into unsigned integers with rounding. These seemingly *small differences* have little to no impact on accuracy, while having tremendous impact on robustness against bit errors, see Sec. 5.1 and App. D.

4.2 Training with Weight Clipping as Regularization

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 (cf. Sec. 4.1), i.e., $q_{\max} \leq w_{\max}$. It might seem that weight clipping with small $w_{\max}$ automatically improves robustness against bit errors as the absolute errors are reduced. However, the *relative* errors are not influenced by rescaling. As the DNN's decision is usually invariant to rescaling, reducing the scale of the weights does not impact robustness. In fact, the mean relative error of the weights in Fig. 4 (right) increased with clipping at $w_{\max} = 0.1$. Thus, weight clipping does *not* "trivially" improve robustness by reducing the scale of weights. Nevertheless, we found that weight clipping actually improves robustness considerably on top of our robust quantization.

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. This is illustrated in Fig. 6 (right): using $w_{\max} = 0.1$, the weights are (depending on the layer) up to 5 times smaller. Considering deep NNs, the "effective" scale factor for the logits is significantly larger, scaling exponentially with the number of layers. Thus, using $w_{\max} = 0.1$ is a significant constraint on the DNNs ability to produce large logits. As result, weight clipping produces a much more uniform weight distribution. Fig. 6 (left and middle) shows that a DNN constrained at $w_{\max} = 0.1$ can produce similar logit and confidence distributions (in blue) as the unclipped DNN. At the same time, random bit errors, have a significantly smaller impact on the logits and confidences (in red). Fig. 6 (right column) also shows the induced redundancy in the weight distribution. Weight clipping leads to more weights being utilized, i.e., less weights are zero (note log-scale, marked in red, on the y-axis). Also,

more weights reach large values, relative to the maximum absolute weight. Overall, we found weight clipping to be an easy-to-use but effective measure to improve weight robustness. We use CLIPPING$_{w_{\max}=0.1}$ to refer to, e.g., weight clipping with $w_{\max} = 0.1$. For more evidence supporting our argumentation, see Tab. 2. For example, we show that DNNs loose robustness when using label smoothing, i.e., not enforcing high confidences/logits during training.

4.3 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 following learning problem, which we optimize as illustrated in Fig. 5:

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

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. This is desirable to avoid an increase in (clean) test error and stabilizes training compared to training only on bit errors in the weights. Note that bit error rate p implies, in expectation, pmW bit errors. Following Alg. 1, 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. In spirit, this is similar to data augmentation, however, the perturbation is applied on the weights instead of the inputs. As we found that introducing bit errors right from the start may prevent the DNN from converging, we apply bit errors as soon as the (clean) cross-entropy loss is below 1.75. Interestingly, weight clipping and RANDBET have somewhat orthogonal effects, which allows to combine them easily in practice: While weight clipping encourages redundancy in weights by constraining them to $[-w_{\max}, w_{\max}]$, RANDBET (w/o weight clipping) causes the DNN to have larger tails in the weight distribution, as shown in Fig. 6 (bottom). However, considering logits and confidences, especially with random bit errors (in red), RANDBET alone performs slightly worse than CLIPPING$_{0.1}$. Thus, RANDBET becomes particularly effective when combined with weight clipping, as we make explicit using the notation RANDBET$_{w_{\max}}$ and in Alg. 1.

5 EXPERIMENTS

We present experiments on MNIST (LeCun et al., 1998) and CIFAR (Krizhevsky, 2009). We first analyze the impact of fixed-point quantization schemes on robustness (Sec. 5.1). Subsequently, we discuss weight clipping (CLIPPING, Sec. 5.2), showing that improved robustness originates from increased redundancy in the weight distribution. Then, we focus on random bit error training (RANDBET, Sec. 5.3). We show that related work (Kim et al., 2018; Koppula et al., 2019) does not generalize, while RANDBET generalizes across chips and voltages, as demonstrated on profiled bit error patterns from different chips. Sec. 5.4 summarizes our results for various precisions m.

Table 1: **Quantization Robustness.** RErr for random bit errors at $p = 0.05\%$ and $p = 0.5\%$ for normal training with different quantization schemes discussed in Sec. 4.1. Minor differences can have large impact on RErr while clean test error is largely unaffected. For 8 bit the second row shows NORMAL quantization (symmetric/per-layer) whereas the last row is our RQUANT. For 4 bits we show CLIPPING$_{0.1}$ +RQUANT with and without rounding.

	Quantization Schemes (CIFAR10)	Err in %	RErr in % p=0.05	RErr in % p=0.5
8 bit	Eq. (1), global	4.63	86.01 ±3.65	90.71 ±0.49
	Eq. (1), per-layer	4.36	5.51 ±0.19	24.76 ±4.71
	+asymmetric	4.36	6.47 ±0.22	40.78 ±7.56
	+unsigned	4.42	6.97 ±0.28	17.00 ±2.77
	+rounding (=RQUANT)	4.32	**5.10** ±0.13	**11.28** ±1.47
4 bit	w/o rounding*	5.81	90.40 ±0.21	90.36 ±0.2
	w/ rounding*	**5.29**	**5.75** ±0.06	**7.71** ±0.36

Metrics: We report (clean) test error Err (lower is better, ↓), corresponding to *clean* weights, and **robust test error RErr** (↓) which is the **test error after injecting bit errors into the weights**. As the bit errors are random we report the *average* RErr and its standard deviation for 50 samples of random bit errors with rate p as detailed in Sec. 3.

Architecture: We use SimpleNet (HasanPour et al., 2016), providing comparable performance to ResNets (He et al., 2016) with only W=5.5Mio weights on CIFAR10. On MNIST, we halve all channel widths, resulting in roughly 1Mio weights. On CIFAR100, we use a Wide ResNet (WRN) (Zagoruyko & Komodakis, 2016) with group normalization (GN) (Wu & He, 2018). Batch normalization (BN) (Ioffe & Szegedy, 2015) works as well but models using BN yield consistently worse robustness against bit errors, see App. G.1 for a discussion.

Training: We use stochastic gradient descent with an initial learning rate of 0.05, multiplied by 0.1 after $^2/_5$, $^3/_5$ and $^4/_5$ of 100/250 epochs on MNIST/CIFAR. On CIFAR, we whiten the input images and use AutoAugment (Cubuk et al., 2018) with Cutout (Devries & Taylor, 2017). For RANDBET, random bit error injection starts when the loss is below 1.75 on MNIST/CIFAR10 or 3.5 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 CLIPPING$_{w_{\max}}$ or together with RANDBET as RANDBET$_{w_{\max}}$. For RQUANT,

Table 2: **Weight Clipping Robustness.** Clean Err and RErr as well as clean confidence and confidence at p=1% bit errors (in %, higher is better, ↑) for CLIPPING and CLIPPING with label smoothing (+LS). Err increases for $w_{\max} = 0.025$ where the DNN is not able to produce large (clean) confidences. LS consistently reduces robustness, indicating that robustness is due to enforcing high confidence during training *and* weight clipping.

Model (CIFAR10)	Err in %	Conf in %	Conf p=1	RErr in % p=0.1	RErr in % p=1
RQUANT	**4.32**	**97.42**	78.43	5.54	32.05
CLIPPING$_{0.15}$	4.42	96.90	88.41	**5.31**	13.08
CLIPPING$_{0.1}$	4.82	96.66	92.97	5.58	8.93
CLIPPING$_{0.05}$	5.44	95.90	**94.73**	5.90	**7.18**
CLIPPING$_{0.025}$	7.10	84.69	83.28	7.40	8.18
CLIPPING$_{0.15}$ +LS	4.67	88.22	47.55	5.83	29.40
CLIPPING$_{0.1}$ +LS	4.82	87.90	78.89	6.10	10.59
CLIPPING$_{0.05}$ +LS	5.30	87.41	85.04	6.43	7.30

$m = 8$, we obtain 4.3% on CIFAR10 and 18.5% Err on CIFAR100. On MNIST, 0.47% are possible even for $m = 2$.

Our **appendix** includes implementation details (App. D), more information on our experimental setup (App. F), and complementary experiments (App. G). Among others, we discuss the robustness of BN (App. G.1), other architectures such as ResNet-50 (App. G.1), qualitative results for CLIPPING (App. G.3) and complete results for $m = 4, 3, 2$ bits precision (App. G.8). Also, we discuss a simple guarantee how the average RErr relates to the true expected robust error (App. C.2). Our **code** will be made publicly available.

5.1 Quantization Choice Impacts Robustness

Quantization schemes affect robustness significantly, even when not affecting accuracy. For example, Tab. 1 shows that per-layer quantization reduces RErr significantly for small bit error rates, e.g., $p = 0.05\%$. While asymmetric quantization further reduces the quantization range, RErr increases, especially for large bit error rates, e.g., $p = 0.5\%$ (marked in red). This is despite Fig. 4 showing a slightly smaller impact of bit errors. This is caused by an asymmetric quantization into *signed* integers: Bit flips in the most significant bit (MSB, i.e., sign bit) are not meaningful if the quantized range is not symmetric as the sign bit does not reflect the sign of the represented weight value, see App. G.2. Similarly, replacing integer conversion of $^{w_i}/_{\Delta}$ by proper rounding, $\lceil {}^{w_i}/_{\Delta} \rfloor$, reduces RErr significantly (resulting in our RQUANT). This becomes particularly important for $m = 4$. Here, rounding also improves clean Err slightly, but the effect is significantly less pronounced. Proper rounding generally reduces the approximation error of the quantization scheme. These errors are magnified when considering bit errors at test time, even though DNNs can compensate such differences during training to achieve good accuracy,

Table 3: **Fixed Pattern Bit Error Training.** RErr for training on an entirely fixed bit error pattern (PATTBET). *Top:* Evaluation on the same pattern; PATTBET trained on $p = 2.5\%$ does not generalize to $p = 1\%$ even though the bit errors for $p = 1\%$ are a subset of those seen during training for $p = 2.5\%$ (in red). *Bottom:* PATTBET also fails to generalize to completely random bit errors. This can be confirmed on real, profiled bit errors in App. G.5.

Model (CIFAR10)	RErr in %, p in % p=1	RErr in %, p in % p=2.5
Evaluation on Fixed Pattern	p=1	p=2.5
PATTBET p=2.5	14.14	7.87
PATTBET$_{0.15}$ p=2.5	8.50	7.41
Evaluation on *Random* Patterns	p=1	p=2.5
PATTBET$_{0.15}$ p=2.5	12.09	61.59

i.e., low Err. For $m = 4$ or lower, we also found weight clipping to help training, obtaining lower Err. Overall, we show that random bit errors induce unique error distributions, cf. Fig. 4 in DNN weights, heavily dependent on details of the employed fixed-point quantization scheme. We think that robustness against bit errors should become an important criterion for the design of DNN quantization. While our RQUANT performs fairly well, finding an "optimal" robust quantization scheme is an interesting open problem.

5.2 Weight Clipping Improves Robustness

While the quantization range adapts to the weight range after every update during training, weight clipping explicitly constraints the weights to $[-w_{\max}, w_{\max}]$. Tab. 2 shows the effect of different $w_{\max}$ for CIFAR10 with 8 bit precision. The clean test error is not affected for CLIPPING$_{\mathbf{w_{max}=0.15}}$ but one has already strong robustness improvements for $p = 1\%$ compared to RQUANT (RErr of 13.18% vs 32.05%). Further reducing $w_{\max}$ leads to a slow increase in clean Err and decrease in average clean confidence, while significantly improving RErr to 7.18% for $p = 1\%$ at $w_{\max} = 0.05$. For $w_{\max} = 0.025$ the DNN is no longer able to achieve high confidence (marked in red) which leads to stronger loss of clean Err. Interestingly, the gap between clean and perturbed confidences under bit errors for $p = 1\%$ is (almost) monotonically decreasing. These findings generalize to other datasets and precisions, see App. G.8. However, for low precision $m \leq 4$ the effects are stronger as RQUANT alone does not yield any robust models and weight clipping is essential for achieving robustness.

As discussed in Sec. 4.2 the robustness of the DNN originates in the cross-entropy loss enforcing high confidences on the training set and, thus, large logits while weight clipping works against having large logits. Therefore, the network has to utilize more weights with larger absolute values (compared to $w_{\max}$). In order to test this hypothesis, we limit the confidences that need to be achieved via label smoothing (Szegedy et al., 2016), targeting 0.9 for the true class and

Table 4: **Random Bit Error Training (RANDBET).** Average RErr (and standard deviation) of RANDBET evaluated at various bit error rates p and using $m = 8$ or 4 bit precision. For low p, weight clipping provides sufficient robustness. However for $p \geq 0.5$, RANDBET increases robustness significantly. This is pronounced for lower precisions.

	Model (CIFAR10) w_{max}=**0.1**, p in %	Err in %	RErr in % p=0.5	p=1	p=1.5
8bit	RQUANT	**4.32**	11.28 ±1.47	32.05 ±6	68.65 ±9.23
	CLIPPING	4.82	6.95 ±0.24	8.93 ±0.46	12.22 ±1.29
	RANDBET p=0.1	4.72	6.74 ±0.29	8.53 ±0.58	11.40 ±1.27
	RANDBET p=1	4.90	**6.36 ±0.17**	**7.41 ±0.29**	**8.65 ±0.37**
4bit	CLIPPING	**5.29**	7.71 ±0.36	10.62 ±1.08	15.79 ±2.54
	RANDBET p=1	5.39	**7.04 ±0.21**	**8.34 ±0.42**	**9.77 ±0.81**

Table 5: **Generalization to Profiled Bit Errors.** RErr for RANDBET on two different profiled chips. The bit error rates differ across chips due to measurements at different voltages, also see Fig. 3. Chip 2 exhibits a bit error distribution significantly different from uniform random bit errors: bit errors are strongly aligned along columns and biased towards 0-to-1 flips, cf. Fig. 3. Nevertheless, RANDBET generalizes surprisingly well.

Chip (Fig. 3)	Model (CIFAR10)	RErr in %	
Chip 1		p≈0.86	p≈2.75
	RANDBET$_{0.05}$ p=1.5	7.04	9.37
Chip 2		p≈0.14	p≈1.08
	RANDBET$_{0.05}$ p=1.5	6.00	9.00

$^{0.1}/_{9}$ for the other classes. According to Sec. 4.2, this should lead to less robustness, as the DNN has to use "less" weights. Indeed, in Tab. 2, RErr at $p = 1\%$ increases from 13.08% for CLIPPING$_{0.15}$ to 29.4% when using label smoothing (marked in blue). Moreover, the difference between average clean and perturbed confidence is significantly larger for DNNs trained with label smoothing.

In App. G.3 we show that robustness against bit errors also leads to robustness against L_∞ perturbations which generally affect all weights in contrast to random bit errors, and provide more qualitative results about the change of the weight distribution induced by clipping in Fig. 10.

5.3 RANDBET Yields Generalizable Robustness

Training on Profiled Errors Does Not Generalize: Codesign approaches such as (Kim et al., 2018; Koppula et al., 2019) combine training DNNs on profiled SRAM or DRAM bit errors with hardware-approaches to limit the errors' impact. However, profiling SRAM or DRAM requires expensive infrastructure, expert knowledge and time. More importantly, training on profiled bit errors does not generalize to previously unseen bit error distributions (e.g., other chips or voltages): Tab. 3 (top) shows RErr of PATTBET, i.e., pattern-specific bit error training. The main problem is that PATTBET does *not* even generalize to lower bit error rates (i.e., higher voltages) of the same pattern as trained on (marked in red). This is striking as, following Fig. 3, the bit errors form a subset of the bit errors seen during training: training with $p = 2.5\%$ bit errors does not provide robustness for $p = 1\%$, RErr increases 7.9% to 14.1%. It is not surprising, that Tab. 3 (bottom) also demonstrates that PATTBET does not generalize to random bit error patterns: RErr increases from 7.4% to 61.6% at $p = 2.5\%$. The same observations can be made when training on real, profiled bit errors corresponding to the chips in Fig. 3, see App. G.5. Overall, obtaining robustness that generalizes across voltages *and* chips is crucial for low-voltage operation to become practical.

RANDBET Improves Robustness Our RANDBET, combined with weight clipping, further improves robustness and additionally generalizes across chips and voltages. Tab. 4 shows results for weight clipping and RANDBET with $w_{max} = 0.1$ and $m = 8, 4$ bits precision. RANDBET is particularly effective against large bit error rates, e.g., $p = 1.5\%$, reducing RErr from 12.22% to 8.65% ($m = 8$ bits). The effect is pronounced for 4 bits or even lower precision, where models are generally less robust. The optimal combination of weight clipping and RANDBET depends on the bit error rate. For example, in Tab. 2, lowering w_{max} to 0.05 reduces RErr below RANDBET$_{0.1}$ with p=1% for some bit error rates. We also emphasize that RANDBET generalizes to lower bit errors than trained on, in stark contrast to the fixed-pattern training PATTBET. In App. G.7, we also show that RANDBET works on other architectures such as ResNet-50. On other datasets, e.g.,MNIST, RANDBET allows to operate at $p = 12.5\%$ bit error rate with 0.9% RErr and only $m = 2$ bits. At this point, weight clipping alone yields 90% RErr.

RANDBET Generalizes to Profiled Bit Errors: RANDBET also generalizes to bit errors profiled from real chips, corresponding to Fig. 3. Tab. 5 shows results on the two profiled chips of Fig. 3. Profiling was done at various voltage levels, resulting in different bit error rates for each chip. To simulate various weights to memory mappings, we apply various offsets before linearly mapping weights to the profiled SRAM arrays. Tab. 5 reports average RErr, showing that RANDBET generalizes quite well to these profiled bit errors. Regarding chip 1, RANDBET performs very well, even for large $p \approx 2.75$, as the bit error distribution of chip 1 largely matches our error model in Sec. 3, cf. Fig. 3 (left). In contrast, with chip 2 we picked a more difficult bit error distribution which is strongly aligned along columns, potentially hitting many MSBs simultaneously. Thus, RErr increases for chip 2 even for a lower bit error rate $p \approx 1.08$ (marked in red) but energy savings are still possible without degrading prediction performance.

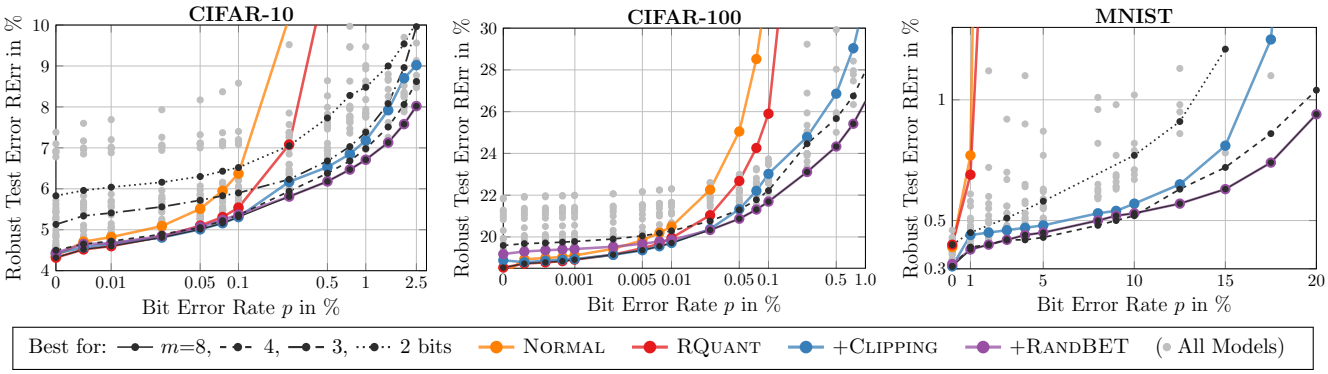

Figure 7: **Bit Error Robustness on CIFAR10, CIFAR100 and MNIST.** Average RErr plotted against bit error rate p, both in %. We considered various models (in • gray), corresponding to different $w_{\max}$ and p during training. We explicitly plot the best model for each bit error rate: for NORMAL (orange), RQUANT (red), CLIPPING (blue) and RANDBET (violet). Note that these might correspond to different $w_{\max}$ and p (also across datasets). Across all approaches, we plot the per-error-rate best model in black: for $m = 8, 4, 3, 2$ bits, depending on dataset. 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.

5.4 Summary and Discussion

Our experiments are summarized in Fig. 7. We consider NORMAL quantization vs. our robust quantization RQUANT, various CLIPPING and RANDBET models with different $w_{\max}$ and p during training (indicated in • gray) and plot RErr against bit error rate p at test time. On all datasets RQUANT outperforms NORMAL. On CIFAR10 (left), RErr increases significantly for RQUANT (red) starting at $p \approx 0.25\%$ bit error rate. While CLIPPING (blue) generally reduces RErr, only RANDBET (violet) can keep RErr around 6% or lower for a bit error rate of $p \approx 0.5\%$. The best model for each bit error rate p (black and solid for $m = 8$) might vary. CIFAR100 is generally more difficult, while significantly higher bit error rates are possible on MNIST. On CIFAR10, RErr increases slightly for $m = 4$. However, for $m = 3, 2$ RErr increases more significantly as clean Err increases by $1 - 2\%$. Nevertheless, RErr only increases slightly for larger bit error rates p. It remains future work whether RANDBET with a more sophisticated (but robust) quantization scheme can enable low-voltage operation even for $m = 2$ bits. In all cases, RErr increases monotonically, ensuring safe operation at higher voltages. The best trade-off between robustness and accuracy depends on the application: higher energy savings require a larger "sacrifice" in terms of RErr. Finally, App. C.2 provides a confidence-interval based guarantee on how strongly RErr is expected to deviate from the empirical results in Fig. 7.

Overall, the results in Fig. 7 enable robust low-voltage operation *without* requiring expensive error correcting codes (ECCs) or other circuit techniques (Reagen et al., 2016; Chandramoorthy et al., 2019). 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 (Kim et al., 2018; Koppula et al., 2019), we do not require expensive expert knowledge or profiling infrastructure. Moreover, RANDBET improves over these approaches by generalizing across chips and voltages. Besides RANDBET, we show that robust fixed-point quantization *only with* weight clipping can provide reasonable robustness, e.g., for $p = 0.1\%$ on CIFAR10. This is without sophisticated quantization scheme, e.g., with special treatment for outliers (Zhuang et al., 2018; Sung et al., 2015; Park et al., 2018), and complementary to (Murthy et al., 2019; Merolla et al., 2016; Sung et al., 2015; Alizadeh et al., 2020), focusing merely on robustness *to* quantization.

6 CONCLUSION

We propose a combination of **robust quantization**, **weight clipping** and **random bit error training (RANDBET)** to get DNNs robustness against random bit errors in their (quantized) weights, enabling low-voltage operation of DNN accelerators to save energy. Here, the accelerator memory is operated far below its rated voltage (Chandramoorthy et al., 2019; Koppula et al., 2019; Kim et al., 2018), inducing exponentially increasing rates of bit errors, directly affecting stored DNN weights. Weight clipping regularizes the weights to a small $[-w_{\max}, w_{\max}]$ during training, encouraging redundancy and increasing robustness. RANDBET further generalizes across chips, with different bit error patterns, and voltages without requiring expensive memory profiling or hardware mitigation strategies. These are important criteria for low-voltage operation in practice. Besides, we also discuss the impact of fixed-point quantization schemes on robustness, which has been neglected in prior work. We are able to train low-precision DNNs robust to significant rates of random bit errors which allow a reduction in energy consumption of roughly 20% or more on MNIST and CIFAR.

REFERENCES

C foreign function interface for python. https://cffi.readthedocs.io/en/latest/index.html.

Cupy: A numpy-compatible array library accelerated by cuda. https://cupy.dev/.

Nervana neural network distiller. https://github.com/nervanasystems/distiller.

NVIDIA Deep Learning Accelerator. http://nvdla.org/.

Nvidia tensorrt. https://developer.nvidia.com/tensorrt.

Alistarh, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: randomized quantization for communication-optimal stochastic gradient descent. *arXiv.org*, abs/1610.02132, 2016.

Alizadeh, M., Behboodi, A., van Baalen, M., Louizos, C., Blankevoort, T., and Welling, M. Gradient ℓ_1 regularization for quantization robustness. In *ICLR*, 2020.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Banner, R., Nahshan, Y., and Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *NeurIPS*, 2019.

Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *arXiv.org*, abs/1712.03141, 2018.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *SP*, 2017.

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. Unlabeled data improves adversarial robustness. *arXiv.org*, abs/1905.13736, 2019.

Chandramoorthy, N., Swaminathan, K., Cochet, M., Paidimarri, A., Eldridge, S., Joshi, R. V., Ziegler, M. M., Buyuktosunoglu, A., and Bose, P. Resilient low voltage accelerators for high energy efficiency. In *HPCA*, 2019.

Chang, K. K., Yaalikçi, A. G., Ghose, S., Agrawal, A., Chatterjee, N., Kashyap, A., Lee, D., O'Connor, M., Hassan, H., and Mutlu, O. Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms. 1(1), 2017.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *AISec*, 2017.

Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., and Temam, O. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. 2014.

Chen, Y., Emer, J. S., and Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ISCA*, 2016.

Cheney, N., Schrimpf, M., and Kreiman, G. On the robustness of convolutional neural networks to internal architecture and weight perturbations. *arXiv.org*, abs/1703.08245, 2017.

Chiang, P., Geiping, J., Goldblum, M., Goldstein, T., Ni, R., Reich, S., and Shafahi, A. Witchcraft: Efficient PGD attacks with random step size. *arXiv.org*, abs/1911.07989, 2019.

Chiu, C., Mehrotra, K., Mohan, C. K., and Ranka, S. Training techniques to obtain fault-tolerant neural networks. In *Annual International Symposium on Fault-Tolerant Computing*, 1994.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I., Srinivasan, V., and Gopalakrishnan, K. PACT: parameterized clipping activation for quantized neural networks. *arXiv.org*, abs/1805.06085, 2018.

Courbariaux, M., Bengio, Y., and David, J. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, 2015.

Croce, F. and Hein, M. Sparse and imperceivable adversarial attacks. In *ICCV*, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv.org*, abs/1805.09501, 2018.

Deodhare, D., Vidyasagar, M., and Keerthi, S. S. Synthesis of fault-tolerant feedforward neural networks using minimax optimization. *TNN*, 9(5):891–900, 1998.

Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv.org*, abs/1708.04552, 2017.

Dong, Y., Liao, F., Pang, T., Hu, X., and Zhu, J. Boosting adversarial attacks with momentum. *arXiv.org*, abs/1710.06081, 2017.

Du, Z., Fasthuber, R., Chen, T., Ienne, P., Li, L., Luo, T., Feng, X., Chen, Y., and Temam, O. Shidiannao: shifting vision processing closer to the sensor. In *ISCA*, 2015.

Duddu, V., Pillai, N. R., Rao, D. V., and Balas, V. E. Fault tolerance of neural networks in adversarial settings. *arXiv.org*, abs/1910.13875, 2019a.

Duddu, V., Rao, D. V., and Balas, V. E. Adversarial fault tolerant training for deep neural networks. *arXiv.org*, abs/1907.03103, 2019b.

Dumford, J. and Scheirer, W. J. Backdooring convolutional neural networks via targeted weight perturbations. *arXiv.org*, abs/1812.03128, 2018.

Galloway, A., Golubeva, A., Tanay, T., Moussa, M., and Taylor, G. W. Batch normalization is a cause of adversarial vulnerability. *arXiv.org*, abs/1905.02161, 2019.

Ganapathy, S., Kalamatianos, J., Kasprak, K., and Raasch, S. On characterizing near-threshold SRAM failures in FinFET technology. 2017.

Ganapathy, S., Kalamatianos, J., Beckmann, B. M., Raasch, S., and Szafaryn, L. G. Killi: Runtime fault classification to deploy low voltage caches without MBIST. In *HPCA*, 2019.

Goncharenko, A., Denisov, A., Alyamkin, S., and Terentev, E. Fast adjustable threshold for uniform neural network quantization. *arXiv.org*, abs/1812.07872, 2018.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T. A., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv.org*, abs/1810.12715, 2018.

Guo, Y. A survey on methods and theories of quantized neural networks. *arXiv.org*, abs/1808.04752, 2018.

Guo, Z., Carlson, A., Pang, L., Duong, K., Liu, T. K., and Nikolic, B. Large-scale SRAM variability characterization in 45 nm CMOS. *JSSC*, 44(11), 2009.

HasanPour, S. H., Rouhani, M., Fayyaz, M., and Sabokrou, M. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv.org*, abs/1608.06037, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

He, Z., Rakin, A. S., Li, J., Chakrabarti, C., and Fan, D. Defending and harnessing the bit-flip based adversarial weight attack. In *CVPR*, 2020.

Hendrycks, D. and Dietterich, T. G. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv.org*, abs/1903.12261, 2019.

Huang, R., Xu, B., Schuurmans, D., and Szepesvári, C. Learning with a strong adversary. *arXiv.org*, abs/1511.03034, 2015.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR*, 18, 2017.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. In *ICML*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A. G., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, 2018.

Ji, Y., Zhang, X., Ji, S., Luo, X., and Wang, T. Model reuse attacks on deep learning systems. In *CCS*, 2018.

Kang, D., Sun, Y., Hendrycks, D., Brown, T., and Steinhardt, J. Testing robustness against unforeseen adversaries. *arXiv.org*, abs/1908.08016, 2019.

Khalil, E. B., Gupta, A., and Dilkina, B. Combinatorial attacks on binarized neural networks. In *ICLR*, 2019.

Kim, S., Howe, P., Moreau, T., Alaghi, A., Ceze, L., and Sathe, V. MATIC: learning around errors for efficient low-voltage neural network accelerators. In *DATE*, 2018.

Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J., Lee, D., Wilkerson, C., Lai, K., and Mutlu, O. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ISCA*, 2014.

Koppula, S., Orosa, L., Yaglikçi, A. G., Azizi, R., Shahroodi, T., Kanellopoulos, K., and Mutlu, O. EDEN: enabling energy-efficient, high-performance deep neural network inference using approximate DRAM. In *MICRO*, pp. 166–181, 2019.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv.org*, abs/1806.08342, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv.org*, abs/1607.02533, 2016.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

Li, H., De, S., Xu, Z., Studer, C., Samet, H., and Goldstein, T. Training quantized nets: A deeper understanding. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *NeurIPS*, 2017.

Liao, C., Zhong, H., Squicciarini, A. C., Zhu, S., and Miller, D. J. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv.org*, abs/1808.10307, 2018.

Lin, D. D., Talathi, S. S., and Annapureddy, V. S. Fixed point quantization of deep convolutional networks. In *ICML*, 2016.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. *arXiv.org*, abs/1611.02770, 2016.

Liu, Y., Ma, S., Aafer, Y., Lee, W., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. In *NDSS*, 2018.

Lopes, R. G., Yin, D., Poole, B., Gilmer, J., and Cubuk, E. D. Improving robustness without sacrificing accuracy with patch gaussian augmentation. In *ICML Workshops*, 2019.

Lu, J., Sibai, H., Fabry, E., and Forsyth, D. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv.org*, abs/1707.03501, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018.

Maini, P., Wong, E., and Kolter, J. Z. Adversarial robustness against the union of multiple perturbation models. *arXiv.org*, abs/1909.04068, 2019.

Merolla, P., Appuswamy, R., Arthur, J. V., Esser, S. K., and Modha, D. S. Deep neural networks are robust to weight binarization and other non-linear distortions. *arXiv.org*, abs/1606.01981, 2016.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. Distributional smoothing with virtual adversarial training. *arXiv.org*, abs/1507.00677, 2015.

Mu, N. and Gilmer, J. Mnist-c: A robustness benchmark for computer vision. *ICML Workshops*, 2019.

Murdock, K., Oswald, D., Garcia, F. D., Van Bulck, J., Gruss, D., and Piessens, F. Plundervolt: Software-based fault injection attacks against intel sgx. In *SP*, 2020.

Murthy, A., Das, H., and Islam, M. A. Robustness of neural networks to parameter quantization. *arXiv.org*, abs/1903.10672, 2019.

Neti, C., Schneider, M. H., and Young, E. D. Maximally fault tolerant neural networks. *TNN*, 3(1):14–23, 1992.

Park, E., Kim, D., and Yoo, S. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *ISCA*, 2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NeurIPS Workshops*, 2017.

Peck, J., Roels, J., Goossens, B., and Saeys, Y. Lower bounds on the robustness to adversarial perturbations. In *NeurIPS*, 2017.

Rahman, F. U., Vasu, B., and Savakis, A. E. Resilience and self-healing of deep convolutional object detectors. In *ICIP*, 2018.

Rakin, A. S., He, Z., and Fan, D. Bit-flip attack: Crushing neural network with progressive bit search. In *ICCV*, 2019.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

Reagen, B., Whatmough, P. N., Adolf, R., Rama, S., Lee, H., Lee, S. K., Hernández-Lobato, J. M., Wei, G., and Brooks, D. M. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *ISCA*, 2016.

Sampson, A., Dietl, W., Fortuna, E., Gnanapragasam, D., Ceze, L., and Grossman, D. Enerj: Approximate data types for safe and general low-power computation. *SIGPLAN Not.*, 46(6), 2011.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *INTERSPEECH*, 2014.

Sharma, H., Park, J., Suda, N., Lai, L., Chau, B., Kim, J. K., Chandra, V., and Esmaeilzadeh, H. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks. In *ISCA*, 2018.

Shin, S., Boo, Y., and Sung, W. Fixed-point optimization of deep neural networks with adaptive step size retraining. In *ICASSP*, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Srinivasan, G., Wijesinghe, P., Sarwar, S. S., Jaiswal, A., and Roy, K. Significance driven hybrid 8t-6t SRAM for energy-efficient synaptic storage in artificial neural networks. In *DATE*, 2016.

Stutz, D., Hein, M., and Schiele, B. Disentangling adversarial robustness and generalization. *CVPR*, 2019.

Stutz, D., Hein, M., and Schiele, B. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *ICML*, 2020.

Sung, W., Shin, S., and Hwang, K. Resiliency of deep neural networks under quantization. *arXiv.org*, abs/1511.06488, 2015.

Sze, V., Chen, Y., Yang, T., and Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *IEEE*, 105(12), 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv.org*, abs/1312.6199, 2013.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Tang, A., Sethumadhavan, S., and Stolfo, S. J. CLKSCREW: exposing the perils of security-oblivious energy management. In *USENIX*, 2017.

Torres-Huitzil, C. and Girau, B. Fault and error tolerance in neural networks: A review. *IEEE Access*, 5, 2017.

Tramèr, F. and Boneh, D. Adversarial training and robustness for multiple perturbations. *arXiv.org*, abs/1904.13000, 2019.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. *arXiv.org*, abs/1805.12152, 2018.

Uesato, J., Alayrac, J., Huang, P., Stanforth, R., Fawzi, A., and Kohli, P. Are labels required for improving adversarial robustness? *arXiv.org*, abs/1905.13725, 2019.

Weng, T.-W., Zhao, P., Liu, S., Chen, P.-Y., Lin, X., and Daniel, L. Towards certificated model robustness against weight perturbations. In *AAAI*, 2020.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

Wu, Y. and He, K. Group normalization. In *ECCV*, pp. 3–19, 2018.

Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., and Jain, A. K. Adversarial attacks and defenses in images, graphs and text: A review. *arXiv.org*, abs/1909.08072, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*, 2016.

Zhang, H., Weng, T., Chen, P., Hsieh, C., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *NeurIPS*, pp. 4944–4953, 2018a.

Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M. P., Huang, H., and Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In *AsiaCCS*, 2018b.

Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv.org*, abs/1606.06160, 2016.

Zhuang, B., Shen, C., Tan, M., Liu, L., and Reid, I. D. Towards effective low-bitwidth convolutional neural networks. In *CVPR*, 2018.

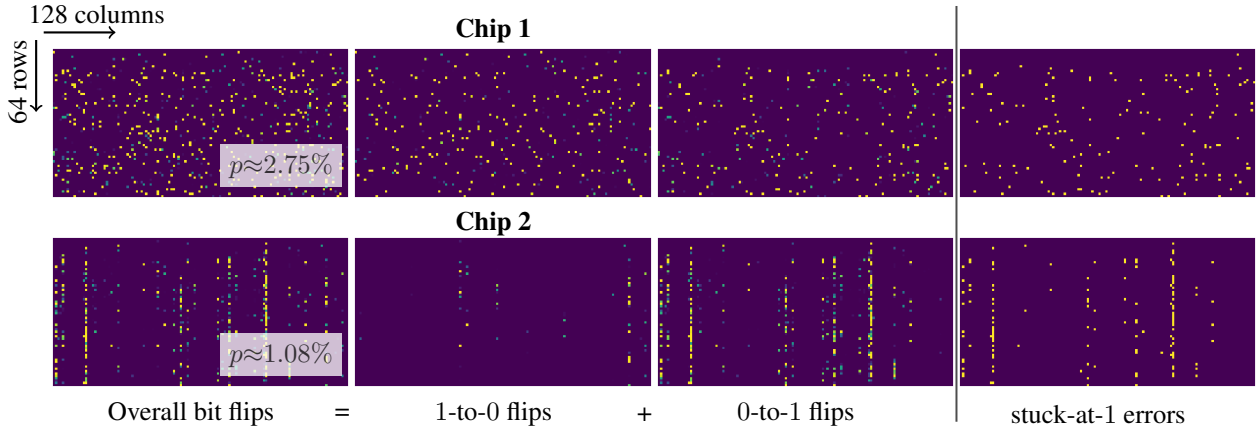

Figure 8: **Low-Voltage Induced Bit Errors on Profiled Chips.** Complementary to Fig. 3, we break the the bit error distribution of chips 1 and 2 down into 1-to-0 and 0-to-1 bit flips. Additionally, we show that most of the bit errors are actually stuck-at-errors and, thus, not subject to randomness. As before, we show a sub-array of size 64×128 from all profiled bit cells (i.e., across all SRAM arrays). Sec. A includes details on profiling.

A Energy Savings in Fig. 1

Fig. 1 shows bit error rate characterization results of SRAMs in the DNN accelerator chip described in (Chandramoorthy et al., 2019), fabricated using 14nm FinFET technology. The average bit error rate is measured from 32 SRAMs, each SRAM array of size 4KB (512×64 bit), as supply voltage is scaled down. *Bit error rate* p (in %) at a given supply voltage is measured as the count of read or write bit cell failures averaged over the total number of bit cells in the SRAM. A bit cell failure refers to reading 1 on writing 0 or reading 0 on writing 1. For a more comprehensive characterization of SRAMs in 14nm technology, the reader is referred to (Ganapathy et al., 2017). Fig. 1 also shows the energy per write and read access of a 4KB (512×64 bit) SRAM, obtained from Cadence Spectre simulations. Energy is obtained at the same constant clock frequency at all supply voltages. The voltage (x-axis) shown is normalized over V_{min} which is the lowest measured voltage at which there are no bit cell failures. Energy shown in the graph (secondary axis on the right) is also normalized over the energy per access at V_{min}.

Accelerators such as (Chen et al., 2016; 2014; Chandramoorthy et al., 2019; Reagen et al., 2016; nvd; Du et al., 2015; Sharma et al., 2018) have a large amount of on-chip SRAM to store weights and intermediate computations. Total dynamic energy of accelerator SRAMs can be obtained as the total number of SRAM accesses times the energy of a single SRAM access. Optimized dataflow in accelerators leads to better re-use of weights read from memories in computation, reducing the number of such memory accesses (Chen et al., 2016; 2014; nvd). Low voltage operation focuses on reducing the memory access energy, leading to significant energy savings as shown.

B Related Work

In the following, we briefly review work on adversarial robustness, fault tolerance, backdooring and quantization. These areas are broadly related to the topic of the main paper.

Adversarial and Corruption Robustness: Robustness of DNNs against adversarially perturbed or randomly corrupted inputs received considerable attention in recent years, see, e.g., relevant surveys (Biggio & Roli, 2018; Xu et al., 2019). Adversarial examples (Szegedy et al., 2013), i.e., nearly imperceptibly perturbed inputs causing misclassification, consider an adversarial environment where potential attackers can actively manipulate inputs. This has been shown to be possible in the white-box setting, with full access to the DNN, e.g., (Madry et al., 2018; Carlini & Wagner, 2017; Dong et al., 2017; Chiang et al., 2019; Croce & Hein, 2020), as well as in the black-box setting, without access to DNN weights and gradients, e.g., (Chen et al., 2017; Ilyas et al., 2018; Croce & Hein, 2019; Andriushchenko et al., 2020). Such attacks are also transferable between models (Liu et al., 2016) and can be applied in the physical world (Lu et al., 2017; Kurakin et al., 2016). Obtaining robustness against adversarial inputs is challenging, recent work focuses on achieving certified/provable robustness (Peck et al., 2017; Zhang et al., 2018a; Wong & Kolter, 2018; Gowal et al., 2018) and variants of adversarial training (Miyato et al., 2015; Huang et al., 2015; Madry et al., 2018), i.e., training on adversarial inputs generated on-the-fly. Adversarial training has been shown to work well empirically, and flaws such as reduced accuracy (Stutz et al., 2019; Tsipras et al., 2018) or generalization to attacks not seen during training has been addressed repeatedly (Carmon et al., 2019; Uesato et al., 2019; Stutz et al., 2020; Tramèr & Boneh, 2019; Maini

Table 6: **Architectures, Number of Weights W, Expected Number of Bit Errors.** *Left and Middle:* SimpleNet architectures used for MNIST and CIFAR10 with the corresponding output sizes, channels N_C, height N_H and width N_W, and the total number of weights W. We use group normalization *with* learnable scale/bias, but reparameterized as outlined in App. E. *Right:* The number of expected bit errors for random bit errors, i.e., pmW.

SimpleNet on **MNIST**	
Layer	Output Size N_C, N_H, N_W
Conv+GN+ReLU	32, 28, 28
Conv+GN+ReLU	64, 28, 28
Conv+GN+ReLU	64, 28, 28
Conv+GN+ReLU	64, 28, 28
Pool	64, 14, 14
Conv+GN+ReLU	64, 14, 14
Conv+GN+ReLU	64, 14, 14
Conv+GN+ReLU	128, 14, 14
Pool	128, 7, 7
Conv+GN+ReLU	256, 7, 7
Conv+GN+ReLU	1024, 7, 7
Conv+GN+ReLU	128, 7, 7
Pool	128, 3, 3
Conv+GN+ReLU	128, 3, 3
Avg Pool	128, 1, 1
FC	10
W	1,082,826

SimpleNet on **CIFAR10**	
Layer	Output Size N_C, N_H, N_W
Conv+GN+ReLU	64, 32, 32
Conv+GN+ReLU	128, 32, 32
Conv+GN+ReLU	128, 32, 32
Conv+GN+ReLU	128, 32, 32
Pool	128, 16, 16
Conv+GN+ReLU	128, 16, 16
Conv+GN+ReLU	128, 16, 16
Conv+GN+ReLU	256, 16, 16
Pool	256, 8, 8
Conv+GN+ReLU	256, 8, 8
Conv+GN+ReLU	256, 8, 8
Pool	256, 4, 4
Conv+GN+ReLU	512, 4, 4
Pool	512, 2, 2
Conv+GN+ReLU	2048, 2, 2
Conv+GN+ReLU	256, 2, 2
Pool	256, 1, 1
Conv+GN+ReLU	256, 1, 1
Avg Pool	256, 1, 1
FC	10
W	5,498,378

p on **MNIST**	
p in %	$pmW, m = 8$
Random Bit Errors	
10	866260
5	433130
1.5	129939
1	86626
0.5	43313

p on **CIFAR**	
p in %	$pmW, m = 8$
Random Bit Errors	
1	439870
0.5	219935
0.01	43987

et al., 2019). Adversarial inputs have also been considered for quantized DNNs (Khalil et al., 2019). Corrupted inputs, in contrast, consider "naturally" occurring corruptions to which robustness/invariance is desirable for practical applications. Popular benchmarks such as MNIST-C (Mu & Gilmer, 2019), Cifar10-C or ImageNet-C (Hendrycks & Dietterich, 2019) promote research on corruption robustness by extending standard datasets with common corruptions, e.g., blur, noise, saturation changes etc. It is argued that adversarial robustness, and robustness to random corruptions is related. Approaches are often similar, e.g., based on adversarial training (Stutz et al., 2020; Lopes et al., 2019; Kang et al., 2019). In contrast, we consider random bit errors in the weights, not the inputs.

Fault Tolerance: Fault tolerance, describes structural changes such as removed units, and has been studied in early works such as (Neti et al., 1992; Chiu et al., 1994). These approaches obtain fault tolerant NNs using approaches similar to adversarial training (Chiu et al., 1994; Neti et al., 1992; Deodhare et al., 1998). Recently, weight dropping regularization (Rahman et al., 2018) or GAN-based training (Duddu et al., 2019b) has been explored. Additionally, fault tolerance of adversarially robust models has been considered in (Duddu et al., 2019a). We refer to (Torres-Huitzil & Girau, 2017) for a comprehensive survey. In contrast, we do *not* consider structural changes/errors in DNNs.

Backdooring: The goal of backdooring is to introduce a backdoor into a DNN, allowing to control the classification result by fixed input perturbations at test time. This is usually achieved through data poisoning (Liu et al., 2018; Liao et al., 2018; Zhang et al., 2018b). However, some works also consider directly manipulating the weights (Ji et al., 2018; Dumford & Scheirer, 2018). However, such weight perturbations are explicitly constructed not to affect accuracy on test examples without backdoor. In contrast, we consider random bit errors (i.e., weight perturbations) that degrade accuracy significantly.

C Low-Voltage Induced Random Bit Errors in Quantized DNN Weights

We provide a more detailed discussion of the considered error model: random bit errors, induced through low-voltage operation of SRAM or DRAM commonly used on DNN accelerators (Kim et al., 2018; Koppula et al., 2019). Work such as (Chandramoorthy et al., 2019; Koppula et al., 2019) model the effect of low-voltage induced bit errors using two parameters: the probability p_{flt} of bit cells in accelerator memory, being faulty and the probability p_{err} that a faulty bit cell results in a bit error on access. Following measurements in works such as (Ganapathy et al., 2019; Kim et al., 2018), we assume that these errors are *not* transient errors by setting $p_{\text{err}} = 100\%$ such that the overall probability of bit errors is $p := p_{\text{flt}} \cdot p_{\text{err}} = p_{\text{flt}}$. In doing so, we consider the worst-case

Table 7: **Quantization-Aware Training Accuracies.** Clean Err for $m = 8$ bits or lower using our robust fixed-point quantization. We obtain competitive performance for $m = 8$ and $m = 4$ bits. On CIFAR100, a Wide ResNet (WRN) clearly outperforms our standard SimpleNet model. Batch normalization (BN), improving Err slightly on CIFAR10, is significantly less robust than group normalization (GN), cf. Tab. 10. * For $m \leq 4$, we report results with weight clipping, CLIPPING$_{0.1}$.

CIFAR10 SimpleNet+GN

Quant. m	Err in %
–	4.34
8	**4.32**
4*	5.29
3*	5.71

CIFAR10 Arch. Comparison

Model	no Quant.	$m = 8$
SimpleNet+GN	4.34	4.32
SimpleBet+BN	4.04	3.83
ResNet-50+GN	5.88	6.81
ResNet-50+BN	**3.91**	**3.67**

MNIST

Quant. m	Err in %
4	0.4
2*	0.47

CIFAR100

Quant. m, Model	Err in %
8, SimpleNet	23.68
8, WRN	18.53

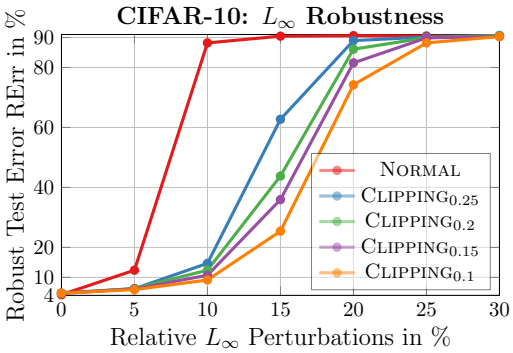

Figure 9: **Weight Clipping Improves L_∞ Robustness.** On CIFAR10, we plot RErr for *relative* L_∞ perturbations on weights: Random noise with L_∞-norm smaller than or equal to $x\%$ of the weight range is applied. CLIPPING clearly improves robustness. Again, the relative magnitude of noise is not affected by weight clipping. Note that L_∞ noise usually affects all weights, while random bit errors affect only a portion of the weights.

where faulty bit cells *always* induce bit errors. However, the noise model from the main paper remains valid for any arbitrary but fixed $p_{\text{err}} \neq 100\%$. For the reminder of this document, we assume the probability of bit error $p = p_{\text{flt}}$, with $p_{\text{err}} = 100\%$, as in the main paper. In the following, we describe the two parameters, p_{flt} and p_{err}, in more details.

Faulty Bit Cells. Due to variations in the fabrication process, SRAM bit cells become more or less vulnerable to low-voltage operation. For a specific voltage, the resulting bit cell failures can be assumed to be random and independent of each other. We assume a bit to be faulty with probability p_{flt} increasing exponentially with decreased voltage (Ganapathy et al., 2017; 2019; Kim et al., 2018; Chandramoorthy et al., 2019). Furthermore, the faulty bits for $p'_{\text{flt}} \leq p_{\text{flt}}$ can be assumed to be a subset of those for p_{flt}. For a fixed chip, consisting of multiple memory arrays, the pattern (spatial distribution) of faulty cells is fixed for a specific supply voltage. Across chips/memory arrays, however, faulty cells are assumed to be random and independent of each other.

Bit Errors in Faulty Bit Cells: Faulty cells may cause bit errors with probability p_{err} upon read/write access. We note that bit errors read from memory affect *all* computations performed on the read weight value. We assume that a bit error flips the currently stored bit, where flips 0-to-1 and 1-to-0 are assumed equally likely.

C.1 Profiled Bit Errors

Fig. 8 splits the bit error distributions of Fig. 3 into a 0-to-1 flip and a 1-to-0 bit flip map. The obtained maps, $p_{1\text{t}0}$ and $p_{0\text{t}1}$, contain per-bit flip probabilities for 1-to-0 and 0-to-1 bit flips. In this particular chip profiled, Fig. 8 (bottom), 0-to-1 flips are more likely. Similarly, Fig. 8 (right) shows that most 0-to-1 flips are actually persistent across time i.e., not random transient errors. The following table summarizing the key statistics of the profiled chips: the overall bit error rate p, the rate of 1-to-0 and 0-to-1 flips $p_{1\text{t}0}$ and $p_{0\text{t}1}$, and the rate of persistent errors p_{sa}, all in %:

Chip	p	$p_{0\text{t}1}$	$p_{1\text{t}0}$	p_{sa}
1	2.744	1.27	1.47	1.223
	0.866	0.38	0.49	0.393
2	4.707	3.443	1.091	0.627
	1.01	0.82	0.19	0.105
	0.136	0.115	0.021	0.01
3	2.297	1.81	0.48	0.204
	0.597	0.496	0.0995	0.206

For evaluation, we assume that the DNN weights are mapped linearly onto the memory of these chips. The bit error maps are of size 8192×128 bits for chips 2 and 3 and 2048×128 bits for chip 1. Furthermore, to simulate various different mappings, we repeat this procedure with various offsets and compute average RErr across all mappings. For results, we refer to App. G.5.

C.2 Bounding Generalization to Random Bit Errors

Let w denote the final weights of a trained DNN f. We test f using n i.i.d. test examples, i.e., $(x_i, y_i)_{i=1}^n$. We denote by w' the weights where each bit of w is flipped with probability p uniformly at random, corresponding to the error model from Sec. 3. The expected *clean* error of f is given by

$$\mathbb{E}[\mathbb{1}_{f(x;w)\neq y}] = \mathrm{P}(f(x;w) \neq y).$$

The expected *robust* error (regarding i.i.d. test examples drawn from the data distribution) with random bit errors in

Table 8: **Impact of Quantization Scheme on Robustness.** Complementary to Tab. 1, we report Err and RErr for various bit error rates p for the quantization scheme in Eq. (1) with global, per-layer and asymmetric quantization, $m = 8$ bits. Instead of quantizing into signed integer, using unsigned integers works better for asymmetric quantization. Furthermore, proper rounding instead of integer conversion also improves robustness. Note that influence on clean Err is neglegible, i.e., the DNN can "learn around" these difference in quantization-aware training. Especially for $m = 4$ bit, the latter makes a significant difference in terms of robustness.

CIFAR10: quantization robustness								
	Model (see text)	Err in %	RErr in %, p in % p=0.01					
			0.01	0.05	0.1	0.5	1	1.5
$m = 8$ bit	Eq. (1), global	4.63	10.70 ±1.37	86.01 ±3.65	90.36 ±0.66	90.71 ±0.49	90.57 ±0.43	–
	Eq. (1), per-layer (= NORMAL)	4.36	4.82 ±0.07	5.51 ±0.19	6.37 ±0.32	24.76 ±4.71	72.65 ±6.35	87.40 ±2.47
	+asymmetric	4.36	5.76 ±0.09	6.47 ±0.22	7.85 ±0.46	40.78 ±7.56	76.72 ±7.01	85.83 ±2.58
	+unsigned	4.42	6.58 ±0.13	6.97 ±0.28	7.49 ±0.41	17.00 ±2.77	54.57 ±8.58	83.18 ±3.94
	+rounded (= RQUANT)	4.32	4.60 ±0.08	5.10 ±0.13	5.54 ±0.2	11.28 ±1.47	32.05 ±6	68.65 ±9.23
4 bit	integer conversion	5.81	90.46 ±0.2	90.40 ±0.21	90.39 ±0.22	90.36 ±0.2	90.36 ±0.22	90.39 ±0.22
	proper rounding	5.29	5.49 ±0.04	5.75 ±0.06	5.99 ±0.09	7.71 ±0.36	10.62 ±1.08	15.79 ±2.54

the (quantized) weights is

$$\mathbb{E}[\mathbb{1}_{f(x;w')\neq y}] = \mathrm{P}(f(x;w') \neq y).$$

Here, the weights of the neural network are themselves random variables. Therefore, with x, y, w, and w' we denote the random variables corresponding to test example, test label, weights and weights bit random bit errors. With x_j, y_j, w_i and w'_i we denote actual examples. Then, the following proposition derives a simple, probabilistic bound on the deviation of expected robust error from the empirically measured one (i.e., RErr in our experiments):

Proposition 1. *Let w'_i, $i = 1, \dots, l$ be l examples of weights bit random bit errors (each bit flipped with probability p). Then it holds*

$$\mathrm{P}\Big(\frac{1}{nl}\sum_{j=1}^{n}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;w'_i)\neq y_j} - \mathrm{P}(f(x;w') \neq y) \geq \epsilon\Big) \leq (n+1)e^{-n\epsilon^2\frac{l}{(\sqrt{l}+\sqrt{n})^2}}.$$

As alternative formulation, with probability $1-\delta$ it holds

$$\mathrm{P}(f(x;w'_i) \neq y) < \frac{1}{nl}\sum_{j=1}^{n}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;w'_i)\neq y_j} + \sqrt{\frac{\log\left(\frac{n+1}{\delta}\right)}{n}}\frac{\sqrt{l}+\sqrt{n}}{\sqrt{l}}.$$

Proof. Let $0 < \alpha < 1$. Using the Hoeffding inequality and union bound, we have:

$$\mathrm{P}\Big(\max_{j=1,\dots,n}\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;w'_i)\neq y_j} - \mathbb{E}_{w'}[\mathbb{1}_{f(x_j;w')\neq y_j}] > \alpha\epsilon\Big)$$
$$= \mathrm{P}\Big(\bigcup_{j=1,\dots,n}\{\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;w'_i)\neq y_j} - \mathbb{E}_{w'}[\mathbb{1}_{f(x_j;w')\neq y_j}] > \alpha\epsilon\}\Big)$$
$$\leq n\, e^{-l\alpha^2\epsilon^2}.$$

Then, again by Hoeffing's inequality, it holds:

$$\mathrm{P}\Big(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{w'}[\mathbb{1}_{f(x_j;w')\neq y_j}] - \mathbb{E}_{x,y}[\mathbb{E}_{w'}[\mathbb{1}_{f(x;w')\neq y}]] > (1-\alpha)\epsilon\Big) \leq e^{-n\epsilon^2(1-\alpha)^2}.$$

Thus, using

$$a + b > \epsilon \Longrightarrow \{a > \alpha\epsilon\} \cup \{b > (1-\alpha)\epsilon\}$$

gives us:

$$\mathrm{P}\Big(\frac{1}{nl}\sum_{j=1}^{n}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;w'_i)\neq y_j} - \mathrm{P}(f(x;w') \neq y) \geq \epsilon\Big)$$
$$= \mathrm{P}\Big(\frac{1}{n}\sum_{j=1}^{n}\big(\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f_{w'_i}(x_j)\neq y_j} - \mathbb{E}_{w'}[\mathbb{1}_{f(x_j;w'_i)\neq y_j}]\big)$$
$$+ \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{w'}[\mathbb{1}_{f(x_j;w'_i)\neq y_j}] - \mathrm{P}(f(x;w') \neq y) \geq \epsilon\Big)$$
$$\leq \mathrm{P}\Big(\frac{1}{n}\sum_{j=1}^{n}\big(\frac{1}{l}\sum_{i=1}^{l}\mathbb{1}_{f(x_j;w'_i)\neq y_j} - \mathbb{E}_{w'}[\mathbb{1}_{f(x_j;w')\neq y_j}]\big) > \alpha\epsilon\Big)$$
$$+ \mathrm{P}\Big(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{w'}[\mathbb{1}_{f(x_j;w')\neq y_j}] - \mathrm{P}(f(x;w') \neq y) \geq (1-\alpha)\epsilon\Big)$$
$$\leq n\, e^{-l\alpha^2\epsilon^2} + e^{-n\epsilon^2(1-\alpha)^2}$$

Having both exponential terms have the same exponent yields $\alpha = \frac{\sqrt{n}}{\sqrt{l}+\sqrt{n}}$ and we get the upper bound of the proposition. □

Remarks: The samples of bit error injected weights $\{w'_i\}_{i=1}^{l}$ can actually be different for any test example (x_j, y_j), even though this is not the case in our evaluation. Thus, the above bound involves a stronger result: for any test example, the empirical test error with random bit errors (i.e., robust test error RErr) and the expected one have to be similar with the same margin. Note also that this bound holds for any fixed bit error distribution as the only requirement is that the bit error patterns we draw are i.i.d. but

Table 9: **Weight Clipping Improves Robustness.** We report Err and RErr for various experiments on the robustness of weight clipping with $w_{\max}$, i.e., CLIPPING$_{w_{\max}}$. First, we show that the robustness benefit of CLIPPING is independent of quantization-aware training, robustness also improves when applying post-training quantization. Then, we show results for both symmetric and asymmetric quantization. For the latter we demonstrate that label smoothing (Szegedy et al., 2016) reduces the obtained robustness. This supports our hypothesis that weight clipping, driven by minimizing cross-entropy loss during training, improves robustness through redundancy.

CIFAR10 ($m = 8$ bit): clipping robustness for post- and during-training quantization								
	Model	Err in %	RErr in %, p in % p=0.01					
			0.01	0.05	0.1	0.5	1	1.5
Post-Training Asymmetric	NORMAL	4.37	4.95 ±0.11	5.47 ±0.17	6.03 ±0.22	15.42 ±3.4	51.83 ±9.92	81.74 ±5.14
	RQUANT	4.27	4.59 ±0.08	5.10 ±0.13	5.54 ±0.15	10.59 ±1.11	30.58 ±6.05	63.72 ±6.89
	CLIPPING$_{0.25}$	4.96	5.24 ±0.07	5.73 ±0.14	6.16 ±0.21	10.51 ±0.91	26.27 ±5.65	61.49 ±9.03
	CLIPPING$_{0.2}$	5.24	5.48 ±0.05	5.87 ±0.09	6.23 ±0.13	9.47 ±0.7	19.78 ±3.58	43.64 ±8.2
	CLIPPING$_{0.15}$	5.38	5.63 ±0.05	6.03 ±0.09	6.38 ±0.13	8.80 ±0.41	15.74 ±2.24	36.29 ±7.34
	CLIPPING$_{0.1}$	5.32	5.52 ±0.04	5.82 ±0.06	6.05 ±0.07	7.45 ±0.26	9.80 ±0.62	17.56 ±3.08
Symmetric (during training)	NORMAL	4.36	4.82 ±0.07	5.51 ±0.19	6.37 ±0.32	24.76 ±4.71	72.65 ±6.35	87.40 ±2.47
	RQUANT	4.39	4.77 ±0.08	5.43 ±0.21	6.10 ±0.32	17.11 ±3.07	55.35 ±9.4	82.84 ±4.52
	CLIPPING$_{0.25}$	4.63	4.99 ±0.07	5.53 ±0.1	6.06 ±0.16	13.55 ±1.42	41.64 ±7.35	73.39 ±7.15
	CLIPPING$_{0.2}$	4.50	4.79 ±0.06	5.25 ±0.09	5.65 ±0.16	9.64 ±0.99	21.37 ±4.23	45.68 ±7.9
	CLIPPING$_{0.15}$	5.18	5.42 ±0.05	5.76 ±0.08	6.07 ±0.09	8.36 ±0.43	13.80 ±1.45	24.70 ±3.77
	CLIPPING$_{0.1}$	4.86	5.07 ±0.04	5.34 ±0.06	5.59 ±0.1	7.12 ±0.3	9.44 ±0.7	13.14 ±1.79
	CLIPPING$_{0.05}$	5.56	5.70 ±0.03	5.89 ±0.06	6.03 ±0.08	6.68 ±0.14	7.31 ±0.2	8.06 ±0.36
Asymmetric (default) quant. (during training)	NORMAL	4.36	4.82 ±0.07	5.51 ±0.19	6.37 ±0.32	24.76 ±4.71	72.65 ±6.35	87.40 ±2.47
	RQUANT	4.32	4.60 ±0.08	5.10 ±0.13	5.54 ±0.2	11.28 ±1.47	32.05 ±6	68.65 ±9.23
	CLIPPING$_{0.25}$	4.58	4.84 ±0.05	5.29 ±0.12	5.71 ±0.16	10.52 ±1.14	27.95 ±4.16	62.46 ±8.89
	CLIPPING$_{0.2}$	4.63	4.91 ±0.05	5.28 ±0.08	5.62 ±0.11	8.27 ±0.35	18.00 ±2.84	53.74 ±8.89
	CLIPPING$_{0.15}$	4.42	4.66 ±0.05	5.01 ±0.09	5.31 ±0.12	7.81 ±0.6	13.08 ±2.21	23.85 ±5.07
	CLIPPING$_{0.1}$	4.82	5.04 ±0.04	5.33 ±0.07	5.58 ±0.1	6.95 ±0.24	8.93 ±0.46	12.22 ±1.29
	CLIPPING$_{0.05}$	5.44	5.59 ±0.04	5.76 ±0.07	5.90 ±0.07	6.53 ±0.13	7.18 ±0.16	7.92 ±0.25
	CLIPPING$_{0.2}$ +LS	4.48	4.77 ±0.05	5.19 ±0.1	5.55 ±0.12	9.46 ±0.82	32.49 ±5.07	68.60 ±7.33
	CLIPPING$_{0.15}$ +LS	4.67	4.86 ±0.05	5.23 ±0.08	5.83 ±0.12	7.99 ±0.43	29.40 ±6.99	68.99 ±8.48
	CLIPPING$_{0.1}$ +LS	4.82	5.05 ±0.04	5.37 ±0.08	6.10 ±0.11	7.36 ±0.4	10.59 ±1.01	18.31 ±2.84
	CLIPPING$_{0.05}$ +LS	5.30	5.43 ±0.03	5.63 ±0.06	6.43 ±0.07	6.51 ±0.15	7.30 ±0.23	8.06 ±0.38

not the bit errors on the pattern. In App. G.6, we consider results with $l = 10^6$, i.e., $l \gg n$ with $n = 10^4$ on CIFAR10 such that $l/(\sqrt{l}+\sqrt{n})^2$ tends towards one. With $\delta = 0.99$ the excess term $\sqrt{\frac{\log\left(\frac{n+1}{\delta}\right)}{n}}\frac{\sqrt{l}+\sqrt{n}}{\sqrt{l}}$ in the Proposition is equal to 4.1%. Thus larger test sets would be required to get stronger guarantees e.g. for $n = 10^5$ one would get 1.7%.

D QUANTIZATION AND BIT MANIPULATION IN PYTORCH

Our fixed-point quantization Q in Eq. (1) quantizes weights $w_i \in [-q_{\max}, q_{\max}] \subset \mathbb{R}$ into signed integers $\{-2^{m-1}-1, \dots, 2^{m-1}-1\}$. Here, the quantization range $[-q_{\max}, q_{\max}]$ is symmetric around zero. Note that zero is represented exactly. To implement asymmetric quantization, as outlined in Sec. 4.1, the same scheme can be used to quantize weights $w_i \in [q_{\min}, q_{\max}]$ within any arbitrary, potentially asymmetric, interval. To this end, Eq. (1) with $q_{\max} = 1$ is used and the weights in $[q_{\min}, q_{\max}]$ are mapped linearly to $[-1, 1]$ using the transformation N:

$$N(w_i) = \left(\frac{w_i - q_{\min}}{q_{\max} - q_{\min}}\right) \cdot 2 - 1. \tag{3}$$

Generally, $q_{\min}$ and $q_{\max}$ are chosen to reflect minimum and maximum weight value – either from all weights (global quantization) or per-layer. Furthermore, we argue that asymmetric quantization becomes more robust when using *unsigned* integers as representation. In this case, Eq. (1) can be adapted using a simple additive term:

$$\begin{aligned} Q(w_i) &= \left\lceil \frac{w_i}{\Delta} \right\rfloor + (2^{m-1} - 1) \\ Q^{-1}(v_i) &= \Delta(v_i - (2^{m-1} - 1)) \end{aligned} \tag{4}$$

We use asymmetric quantization using N in Eq. (3) with Eq. (4) as our *robust* fixed-point quantization.

Following Sec. 4.1, we implement "fake" fixed-point quantization for quantization-aware training and bit error injection directly in PyTorch (Paszke et al., 2017). Here, fake quantization means that computation is performed in floating point, but before doing a forward pass, the DNN is quantized and dequantized, i.e., $w_q = Q^{-1}(Q(w))$ in Alg. 1. Note that we

Table 10: **Batch Normalization not Robust.** RErr with group normalization (GN) or batch normalization (BN). RErr increases when using BN even though clean Err improves slightly compared GN. However, using batch statistics at test time (i.e., "training mode" in PyTorch) improves RErr significantly indicating that the statistics accumulated throughout training do not account for random bit errors. We use **group normalization as default.**

CIFAR10 ($m = 8$ bit): robustness of BN				
		Err	RErr in %	
		in %	p=0.1	p=0.5
GN	NORMAL	4.32	5.54	11.28
	CLIPPING$_{0.1}$	4.82	5.58	6.95
BN w/ *Accumulated* Statistics				
BN	NORMAL	3.83	6.36	52.52
	CLIPPING$_{0.1}$	4.46	5.32	8.25
BN w/ Batch Statistics at Test Time				
BN	NORMAL	3.83	6.65	9.63
	CLIPPING$_{0.1}$	4.46	6.57	7.29

quantize into *unsigned* 8 bit integers, irrespective of the target precision $m \leq 8$. To later induce random bit errors, the $8 - m$ most significant bits (MSBs) are masked for $m < 8$. Bit manipulation of unsigned 8 bit integers is then implemented in C/CUDA and interfaced to Python using CuPy (cup) or CFFI (cff). These functions can directly operate on PyTorch tensors, allowing bit manipulation on the CPU as well as the GPU. We will make our code publicly available to faciliate research into DNN robustness against random bit errors.

E WEIGHT CLIPPING WITH GROUP/BATCH NORMALIZATION

While weight clipping, i.e., globally constraining weights to $[-w_{\text{max}}, w_{\text{max}}]$ during training, is easy to implement, we make a simple adjustment to group and batch normalization layers: we reparameterize the scale parameter α of batch/-group normalization, which usually defaults to $\alpha = 1$ and may cause problems when clipped, e.g., to $[-0.1, 0.1]$. In particular with aggressive weight clipping, $\alpha \leq w_{\text{max}} < 1$, the normalization layers loose their ability to represent the identity function, considered important for batch normalization in (Ioffe & Szegedy, 2015). Our reparameterization introduces a learnable, auxiliary parameter α' such that α as $\alpha = 1 + \alpha'$ to solve this problem.

F EXPERIMENTAL SETUP

Datasets: We conduct experiments on MNIST[1] (LeCun et al., 1998) and CIFAR[2] (Krizhevsky, 2009). MNIST consists of 60k training and 10k test images from 10 classes. These are gray-scale and of size 28×28 pixels. CIFAR consists of 50k training and 10k test images of size $32 \times 32 \times 3$ (i.e., color images). CIFAR10 has images corresponding to 10 classes, CIFAR100 containts imaves from 100 classes.

Table 11: **Weight Clipping with Weight Scaling.** For group normalization (GN) without the reparameterization in Sec. 4.2, using fixed scale/bias instead, our DNNs are scale-invariant. Scaling RQUANT down to the weight range of CLIPPING$_{0.25}$, however, does not improve robustness. Thus, the robustness benefit of CLIPPING is *not* due to reduced quantization range or smaller absolute errors.

CIFAR10 ($m = 8$ bit): scaling w/o reparameterized GN			
Model	Err	RErr in %, p in %	
(see text)	in %	p=0.1	p=1
RQUANT	4.67	6.12 ±0.2	35.25 ±6.41
CLIPPING$_{0.25}$	4.96	6.13 ±0.16	16.09 ±1.85
RQUANT → CLIPPING$_{0.25}$	4.64	6.10 ±0.18	35.28 ±5.82

Architecture: The used SimpleNet architectures (Hasan-Pour et al., 2016) are summarized in Tab. 6, including the total number of weights W. On CIFAR, this results in a total of roughly $W \approx 5.5$M weights. Due to the lower resolution on MNIST, channel width in each convolutional layer is halved, and one stage of convolutional layers including a pooling layer is skipped. This results in a total of roughly $W \approx 1$M weights. In both cases, we replaced batch normalization (BN) (Ioffe & Szegedy, 2015) with group normalization (GN) (Wu & He, 2018). The GN layers are reparameterized as in App. E to facilitate weight clipping. Tab. 6 also includes the expected number of bit errors given various rates p for random bit errors. Regarding the number of weights W, SimpleNet compares favorably to, e.g., VGG (Simonyan & Zisserman, 2015): VGG-16 has 14M weights on CIFAR. Additionally, we found SimpleNet to be easier to train without BN, which is desirable as BN reduces robustness to bit errors significantly, cf. App. G.1. The ResNet-50 (He et al., 2016) used for experiments in App. G.7 follows the official PyTorch (Paszke et al., 2017) implementation. The Wide ResNet (WRN) (Zagoruyko & Komodakis, 2016) used on CIFAR100 is adapted from[3], but we use 12 base channels, instead of 16, reducing W from roughly 36.5Mio to 20.5Mio.

Training: As outlined in Sec. 5, we use stochastic gradient descent to minimize cross-entropy loss. We use an initial learning rate of 0.05, multiplied by 0.1 after $2/5$, $3/5$ and $4/5$ of 100/250 epochs on MNIST/CIFAR. Our batch size is 128 and momentum of 0.9 is used together with weight decay of $5 \cdot 10^{-4}$. On CIFAR, we whiten the input images

[1] http://yann.lecun.com/exdb/mnist/

[2] https://www.cs.toronto.edu/~kriz/cifar.html

[3] https://github.com/meliketoy/wide-resnet.pytorch

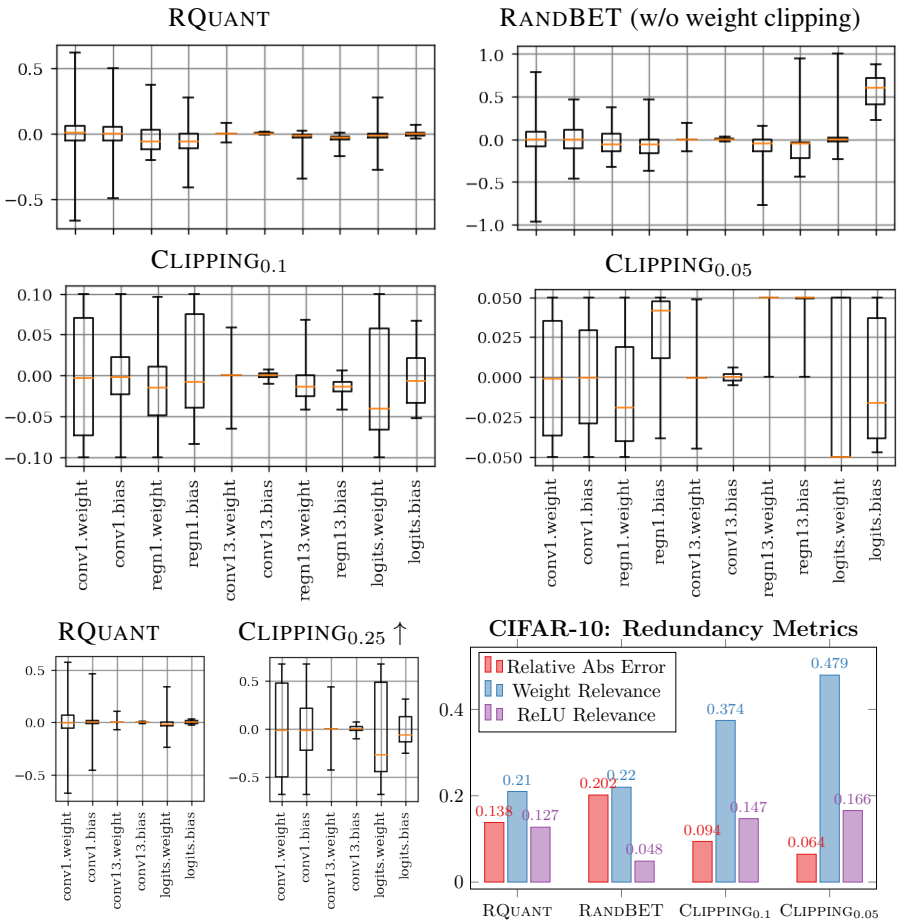

Figure 10: **Weight Clipping Increases Redundancy.** We show weight distributions of selected layers (top) for RQUANT, RANDBET (without weight clipping) as well as CLIPPING$_{0.1}$ and CLIPPING$_{0.05}$. We show weights and biases for the logit layer as well as the first and last (13th) convolutional layer. Scale/Bias parameters of GN are also included. Below (left), this is shown for the scaling experiment from Tab. 11 (GN parameters are fixed). Note that RANDBET only affects the logit layer, while CLIPPING increases the used (relative) weight range significantly. On the bottom (right), we plot various measures of redundancy, see the text for discussion and details. The relative absolute error is computed considering random bit errors with probability $p = 1\%$.

and use AutoAugment[4] (Cubuk et al., 2018) with Cutout (Devries & Taylor, 2017). Cutout is applied with a window size of 16×16, and independent of AutoAugment, we apply random cropping with up to 4 pixels. Created black spaces are filled using the mean image color (grayish). Initialization follows (He et al., 2015). The full training set is used for training, and we do *not* rely on early stopping. For RANDBET, we use $\lambda = 1$ and start injecting bit errors when the loss is below 1.75 on MNIST/CIFAR10 or 3.5 on CIFAR100. Tab. 7 highlights clean test error (Err) obtained for various precision m and compared to other architectures, e.g., ResNet-50, on CIFAR10, which performs worse when using GN.

Random Bit Errors: We simulate 50 different chips with enough memory arrays to accomodate all weights by drawing uniform samples $u^{(c)} \sim U(0,1)^{W \times m}$ for each chip c and all m bits for a total of W weights. Then, for chip c, bit j in weight w_i is flipped iff $u_{ij}^{(c)} \leq p$. This assumes a linear memory layout of all W weights. The pattern, i.e., spatial distribution, of bit errors for chip c is fixed by $u^{(c)}$, while across all 50 chips, bit errors are uniformly distributed. We emphasize that we pre-determine $u^{(c)}$, $c = 1, \ldots, 50$, once for all our experiments using fixed random seeds. Thus, our robustness results are entirely comparable across all models as well as bit error rates p. Also note that, as explained in Sec. 3, the bit errors for a fixed chip c at probability $p' < p$ are a subset of those for bit error rate p. The expected number of bit errors for various rates p is summarized in Tab. 6.

Implementation Details are covered in Sec. D.

[4] https://github.com/DeepVoltaire/AutoAugment

Table 12: **RANDBET Robustness with *Symmetric* Quantization.** Average RErr and standard deviation for CLIPPING and RANDBET with $w_{\max} = 0.1$ and **s**ymmetric quantization, i.e., larger quantization range than asymmetric quantization. Also cf. Tab. 9 and Tab. 18. Robustness decreases slightly compared to asymmetric quantization, however, CLIPPING and RANDBET are still effective in reducing RErr against high bit error rates p.

CIFAR10 ($m = 8$ bit): RANDBET with symmetric quantization							
Model	Err in %	RErr in %, p in % p=0.01					
		0.01	0.05	0.1	0.5	1	1.5
NORMAL	4.36	4.82 ±0.07	5.51 ±0.19	6.37 ±0.32	24.76 ±4.71	72.65 ±6.35	87.40 ±2.47
RQUANT	4.39	4.77 ±0.08	5.43 ±0.21	6.10 ±0.32	17.11 ±3.07	55.35 ±9.4	82.84 ±4.52
CLIPPING$_{0.1}$	4.86	5.07 ±0.04	5.34 ±0.06	5.59 ±0.1	7.12 ±0.3	9.44 ±0.7	13.14 ±1.79
RANDBET$_{0.1}$ p=0.01	5.07	5.27 ±0.04	5.54 ±0.07	5.73 ±0.11	7.18 ±0.29	9.63 ±0.9	13.81 ±2.2
RANDBET$_{0.1}$ p=0.1	4.62	4.83 ±0.04	5.09 ±0.08	5.31 ±0.08	6.70 ±0.28	8.89 ±0.59	12.20 ±1.33
RANDBET$_{0.1}$ p=1	5.03	5.22 ±0.04	5.43 ±0.06	5.61 ±0.07	6.56 ±0.13	7.70 ±0.26	8.99 ±0.42
RANDBET$_{0.1}$ p=1.5	5.24	5.37 ±0.03	5.57 ±0.06	5.76 ±0.07	6.66 ±0.14	7.62 ±0.25	8.71 ±0.42
RANDBET$_{0.1}$ p=2	5.82	5.97 ±0.04	6.19 ±0.07	6.37 ±0.09	7.22 ±0.19	8.03 ±0.23	8.96 ±0.38

G EXPERIMENTS

G.1 Batch Normalization

We deliberately replace batch normalization (BN) (Ioffe & Szegedy, 2015) by group normalization (GN) (Wu & He, 2018) in our experiments. Tab. 10 demonstrates that RErr increases significantly when using BN compared to GN indicating that BN is more vulnerable to bit errors in DNN weights. For example, without clipping, RErr increases from 11.28% to staggering 52.52% when replacing GN with BN. Note that, following App. E, the BN/GN parameters (i.e., scale/bias) are reparameterized to account for weight clipping. The observations in Tab. 10 can also be confirmed without quantization, e.g., considering random L_∞ noise in the weights. We suspect that the running statistics accumulated during training do not account for the random bit errors at test time, even for RANDBET. This is confirmed in Tab. 10 (bottom) showing that RErr reduces significanlty when using the batch statistics at test time. Generally, robustness improves accuracy, but might not be beneficial in terms of robustness, as also discussed for adversariale examples (Galloway et al., 2019). Using GN also motivates our use of SimpleNet instead of, e.g., ResNet-50, which generally performs worse with GN, cf. Tab. 7.

G.2 Robust Quantization

Tab. 8 shows results complementary to the main paper, considering additional bit error rates p. Note that, for $m = 8$ bit, changes in the quantization has neglegible impact on clean Err. Only the change from gloabl to per-layer quantization makes a difference. However, considering RErr for larger bit error rates, reducing the quantization range, e.g., through per-layer and asymmetric quantization, improves robustness significantly. Oher aspects of the quantization scheme also play an important role, especially for low-precision such as $m = 4$ bit, cf. Tab. 8, as outlined in the following.

For example, using asymmetric quantization into *signed* integers actually increases RErr for larger p compared to "just" using symmetric per-layer quantization (rows 2 and 3). Using *unsigned* integers instead reduces RErr significantly. We belive this to be due to the two's complement representation of signed integers being used with an **a**symmetric quantization range. In symmetric quantization (around 0, i.e., $[-q_{\max}, q_{\max}]$), bit errors in the sign bit incur not only a change of the integer's sign, but also the corresponding change in the weights sign[5]. Assuming an asymmetric quantization of $[q_{\min}, q_{\max}]$ with $0 < q_{\min} < q_{\max}$, bit errors in sign bits are less meaningful. For example, flipping any bit 0-to-1 usually increases the value of the integer. However, a 0-to-1 flip in the sign bit actually decreases the value and produces a *negative* integer. However, this change from positive to negative is not reflected in the corresponding weight value (as $q_{\min} > 0$). For high bit error rates $p\%$, this happens more and more frequently and these changes seem to have larger impact on DNN performance, i.e., RErr.

Additionally, we considered the difference between using integer conversion for w_i/Δ and using proper rounding, i.e., $\lceil w_i/\Delta \rfloor$. We emphasize that, for $m = 8$ bit, there is *no* significant difference in terms of clean Err. However, using proper rounding reduces the approximation error slightly. For $m = 8$ bit, using $p = 2.5\%$ bit error rate, the average absolute error (in the weights) across 10 random bit error patterns reduces by 2%. Nevertheless, it has significantly larger impact on RErr. For $m = 4$, this is more pronounced: rounding reduces the average absolute error by roughly 67%. Surprisingly, this is not at all reflected in the clean Err, which only decreases from 5.81% to 5.29%. It seems that the DNN learns to compensate these errors during training. At test time, however, RErr reflects this difference in terms

[5] An *un*signed integer of value 127 is represented as 01111111. Flipping the most (left-most) significant bit results in 11111111 corresponding to 255, i.e., the value increases. For a signed integer in two's complement representation, the same bit flip changes the value from 127 to -1, while 0-to-1 not affecting the sign bit generally increase value (also for negative integers).

Table 13: **RANDBET Variants.** Err and RErr for RANDBET and two variants: curricular RANDBET, with p being increased slowly from $p/20$ to p during the first half of training; and "alternating" RANDBET where weight updates increasing quantization range, i.e., increasing the maximum absolute weight per layer, are not possible based on gradients from perturbed weights, see Sec. G.4 for details. Both variants decrease robustness slightly. This is in contrast to, e.g., (Koppula et al., 2019), using curricular training on profiled bit errors with success.

CIFAR10 ($m = 8$ bit): RANDBET variants	Err	RErr in %	
	in %	p=0.1	p=1
RANDBET p=0.1, $w_{max} = 0.1$	4.93	5.67	8.65
RANDBET p=1, $w_{max} = 0.1$	5.06	5.87	7.60
Curr. RANDBET p=1, $w_{max} = 0.1$	4.89	5.78	8.51
Curr. RANDBET p=1, $w_{max} = 0.1$	5.32	6.13	7.98
Alt. RANDBET p=1, $w_{max} = 0.1$	5.07	5.91	8.93
Alt. RANDBET p=1, $w_{max} = 0.1$	5.24	6.25	8.02

Table 14: **RANDBET with ResNets.** We report RErr for RQUANT, CLIPPING and RANDBET using ResNet-20 and ResNet-50. According to Tab. 7, Err increases significantly when using group normalization for ResNet-50, explaining the generally higher RErr. However, using ResNets, CLIPPING and RANDBET continue to improve robustness significantly, despite a ResNet-50 having roughly 23.5Mio weights.

CIFAR10 ($m = 8$ bit): ResNet architectures	Err	RErr in %	
	in %	p=0.5	p=1.5
ResNet-20			
RQUANT	4.34	13.89 ±2.45	81.25 ±5.08
CLIPPING$_{0.1}$	4.83	6.76 ±0.16	11.23 ±0.97
RANDBET$_{0.1}$, p=1	5.28	6.72 ±0.19	8.96 ±0.49
ResNet-50			
RQUANT	6.81	32.94 ±5.51	90.98 ±0.67
CLIPPING$_{0.1}$	5.99	9.27 ±0.44	36.39 ±7.03
RANDBET$_{0.1}$, p=1	6.04	7.87 ±0.22	11.27 ±0.6

of robustness.

G.3 Weight Clipping

In Tab. 9 we present robustness results, i.e., RErr, for weight clipping. Note that weight clipping constraints the weights during training to $[-w_{max}, w_{max}]$ through projection. We demonstrate that weight clipping can also be used independent of quantization. To this end, we train DNNs with weight clipping, but without quantization. We apply post-training quantization and evaluate bit error robustness. While the robustness is reduced slightly compared to quantization-aware training *and* weight clipping, the robustness benefits of weight clipping are clearly visible. For example, clipping at $w_{max} = 0.1$ improves RErr from 30.58% to 9.8% against $p = 1\%$ bit error rate when performing post-training quantization. With symmetric quantization-aware training, CLIPPING$_{0.1}$ improves slightly to 7.31%. Below (middle), we show results for weight clipping and symmetric quantization. These results are complemented in Tab. 12 with RANDBET. Symmetric quantization might be preferable due to reduced computation and energy cost compared to asymmetric quantization. However, this also increases RErr slightly. Nevertheless, CLIPPING consistently improves robustness, independent of the difference in quantization. Finally, on the bottom, we show complementary results to Tab. 2, confirming the adverse effect of label smoothing (Szegedy et al., 2016) on RErr, cf. Sec. 5.2. Fig. 9 also shows that the obtained robustness generalizes to other error models such as L_∞ weight perturbations, see caption for details.

We hypothesize that weight clipping improves robustness as it encourages redundancy in weights and activations during training. This is because cross-entropy loss encourages large logits and weight clipping forces the DNN to "utilize" many different weights to produce large logits. Tab. 11 presents a simple experiment in support of our hypothesis. We already emphasized that, relatively, weight clipping does *not* reduce the impact of bit errors. Nevertheless, when using group normalization(GN) without our reparameterization, the trained DNNs are scale-invariant in their weights. Thus, we downscale NORMAL to have the same maximum absolute weight value as CLIPPING$_{0.25}$ (NORMAL $\rightarrow$ CLIPPING$_{0.25}$). This scaling is applied globally, not per layer. Tab. 11 shows that "just" down-scaling does not induce robustness, as expected. Thus, the benefit of CLIPPING in terms of robustness does not come from the reduced quantization range.

Fig. 10 presents further supporting evidence for our hypothesis: While RANDBET mainly affects the logits layer, CLIPPING clearly increases the weight range used by the DNN. Here, the weight range is understood relative to w_{max} (or the maxmimum absolute weight value for NORMAL). This is pronounced in particular when up-scaling the clipped model (bottom left). Finally, Fig. 10 (bootom right) also considers three attempts to measure redundancy in weights and activations. The relative absolute error is computed with respect to $p = 1\%$ bit error rate and decreases for CLIPPING, meaning that random bit errors have less impact. *Weight relevance* is computed as the sum of absolute weights, i.e., $\sum_i |w_i|$, normalized by the maximum absolute weight: $\sum_i |w_i| / \max_i |w_i|$. This metric measures how many weights are, considering their absolute value, relevant. Finally, We also measure activation redundancy using ReLU relevance, computing the fraction of non-zero activations after the final ReLU activation. CLIPPING increases redundancy in the final layer significantly. Finally, Fig. 10 (bottom left) shows the difference in weight distributions by upscaling CLIPPING$_{0.25}$ to the same weight range as NORMAL. Clearly, CLIPPING

Table 15: **Generalization to Profiled Bit Errors.** Complementary to Tab. 5, we show RErr on profiled bit errors, chips 1-3, for RANDBET as well as CLIPPING. Note that for chip 3, CLIPPING$_{0.05}$ performs slightly better than RANDBET.

CIFAR10: Generalization to Profiled Bit Errors				
Chip	Model (CIFAR10)	Err in %	RErr in %	
			$p\approx0.86$	$p\approx2.7$
1	RQUANT	4.32	23.57	89.84
	CLIPPING$_{0.05}$	5.44	7.17	10.50
	RANDBET$_{0.05}$ p=1.5	5.62	7.04	9.37
			$p\approx0.14$	$p\approx1$
2	RQUANT	4.32	6.00	74.00
	CLIPPING$_{0.05}$	5.44	5.98	10.02
	RANDBET$_{0.05}$ p=1.5	5.62	6.00	9.00
			$p\approx0.03$	$p\approx0.5$
3	RQUANT	4.32	5.47	80.49
	CLIPPING$_{0.05}$	5.44	5.78	11.88
	RANDBET$_{0.05}$ p=1.5	5.62	5.85	12.44

causes more non-zero weights be learned by the DNN. This can be observed across all types of parameters, i.e., weights or biases as well as convolutional or fully connected layers.

G.4 Random Bit Error Training (RANDBET)

Tab. 12 shows complementary results for RANDBET using *symmetric* quantization. Symmetric quantization generally tends to reduce robustness, i.e., increase RErr, across all bit error rates p, cf. Tab. 4 in the main paper. Thus, the positive impact of RANDBET is pronounced, i.e., RANDBET becomes more important to obtain high robustness when less robust fixed-point quantization is used. These experiments also demonstrate the utility of RANDBET independent of the quantization scheme at hand.

We consider two variants of RANDBET motivated by related work (Koppula et al., 2019). Specifically, in (Koppula et al., 2019), the bit error rate seen during training is increased slowly during training. Note that (Koppula et al., 2019) trains on fixed bit error patterns. Thus, increasing the bit error rate during training is essential to avoid the effect shown in Tab. 3: the DNN is supposed to be robustness to any bit error rate $p' < p$ smaller than the target bit error rate. While this is generally the case using our RANDBET, Tab. 13 shows that slowly increasing the random bit error rate during training, called "curricular" RANDBET, has no significant benefit over standard RANDBET. In fact, RErr increases slightly. Similarly, we found that RANDBET tends to increase the range of weights: the weights are "spread out", cf. Fig. 10 (top right). This also increases the quantization range, which has negative impact on robustness as discussed in Sec. 5.1. Thus, we experimented with RANDBET using two weight updates per iteration: one using clean weights, one on weights with bit errors. This is in contrast to averaging both updates as in Alg. 1. Updates computed from perturbed weights are limited to the current quantization ranges, i.e., the maximum absolute error cannot change. This is ensured through projection. This makes sure that RANDBET does not increase the quantization range during training as changes in the quantization range are limited to updates from clean weights. Again, Tab. 13 shows this variant to perform slightly worse.

Table 16: **Fixed Pattern Bit Error Training.** Complementary results for Tab. 3, reporting RErr for training on fixed (e.g., profiled) bit error patterns (PATTBET). We show additional results for chip 2 from Tab. 5. Note that for PATTBET on chip 1/2 we used only the stuck-at-errors shown in Fig. 8, which is why the bit error rates deviate from Tab. 5.

Model (CIFAR10)	RErr in %, p in %	
***Profiled* Bit Errors (Chip 1)**	$p\approx0.1$	$p\approx0.6$
PATTBET, $p\approx1.22$	9.52	7.20
PATTBET, $p\approx0.39$	5.77	67.87
PATTBET$_{0.15}$, $p\approx1.22$	7.67	6.52
PATTBET$_{0.15}$, $p\approx0.39$	5.94	30.96
***Profiled* Bit Errors (Chip 2)**	$p\approx0.1$	$p\approx0.63$
PATTBET $p\approx0.63$	85.84	10.76
PATTBET, $p\approx0.1$	90.56	5.93
PATTBET$_{0.15}$ $p\approx0.63$	12.02	8.70
PATTBET$_{0.15}$ $p\approx0.1$	90.68	6.51

Table 17: **Results for Probabilistic Guarantees.**. Average RErr and standard deviation for $l = 1$Mio random bit error patterns. In comparison with the results for $l = 50$ from the main paper, there are no significant changes in RErr. However, standard deviation increases slightly, from 0.11 to 0.15 against RANDBET.

CIFAR10: Stress Test for Guarantees			
Model	Err	RErr in %, $p = 1\%$	
(CIFAR10)	in %	$l = 50$	$l = 1$Mio
RQUANT	4.32	32.05 ±6	31.97 ±6.35
CLIPPING$_{0.05}$	5.44	7.18 ±0.16	7.19 ±0.2
RANDBET$_{0.05}$ p=2	5.42	6.71 ±0.11	**6.73 ±0.15**

G.5 Profiled Bit Errors

Following the evaluation on profiled bit errors outlined in App. C.1, Tab. 15 shows complementary results for CLIPPING$_{0.05}$ and RANDBET$_{0.05}$ trained with $p = 1.5\%$ on all profiled chips. Note that for particularly extreme cases, such as chip 3, CLIPPING might perform slightly better than RANDBET. Overall, however, RANDBET generalizes reasonably well, with very good results on chip 1 which is closest to our bit error model. Results on chip 2 and 3, due to bit errors being strongly aligned along columns (cf. Fig. 8), are slightly worse. However, RANDBET does not fail catastrophically. Instead, RErr degrades slowly.

In Tab. 16, we follow the procedure of App. C.1 considering only stuck-at-0 and stuck-at-1 bit errors (i.e., where $p_{1\to0}$ and $p_{0\to1}$ are 1). This is illustrated in Fig. 8 (right). Thus, the bit error rates deviate slightly from those reported in Tab. 15,

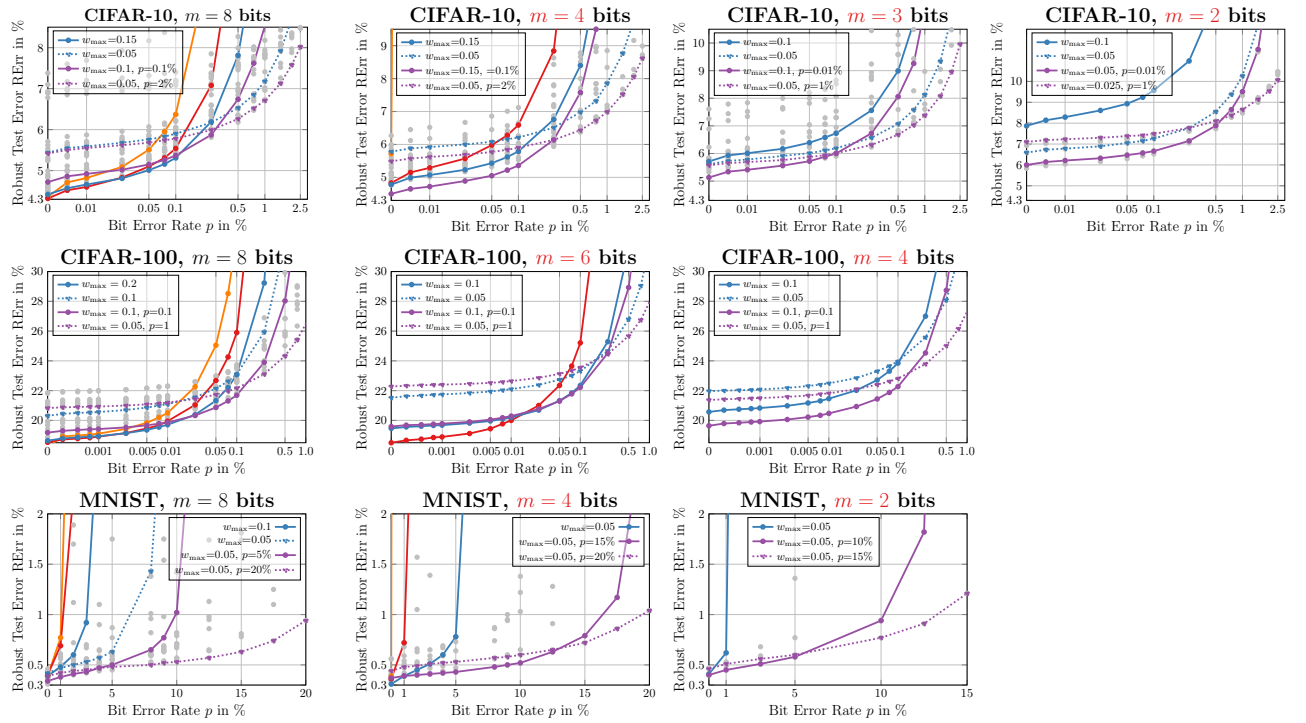

Figure 11: **Summary Results on CIFAR10, CIFAR100 and MNIST.** Complementary to Fig. 7, we highlight individual CLIPPING and RANDBET models. Note that Fig. 7, in contrast, presents the best, i.e., lowest RErr, model for each bit error rate p individually. Instead, individual models help to illustrate the involved trade-offs: CLIPPING with small w_{max} or RANDBET with high bit error rate p increases the clean Err, thereby also increasing RErr for very small bit error rates. However, RErr against large bit error rates can be reduced significantly.

see the table in App. C.1 for details. Furthermore, We consider only one weight-to-SRAM mapping, i.e., without offset. PATTBET is trained and evaluated on the exact same bit error pattern, but potentially with different bit error rates p. Note that the bit errors for $p' < p$ are a subset of those for bit error rate p. Thus, it is surprising that, on both chips 1 and 2, PATTBET trained on higher bit error rates does not even generalize to lower bit error rates (i.e., higher voltage). This is problematic in practice as the DNN accelerator should not perform worse when increasing voltage.

G.6 Guarantees from Prop. 1

Based on the bound derived in Sec. C.2, we conduct experiments with $l = 1$Mio random bit error patterns, such that $l \gg n$ where $n = 10$k is the number of test examples on CIFAR10. Considering Prop. 1, this would guarantee a deviation in RErr of at most 4.1% with probability at least 99%. As shown in Tab. 17, the obtained RErr with 1Mio random bit error patterns deviates insignificantly from the results in the main paper. Only standard deviation of RErr increases slightly. These results emphasize that the results for CLIPPING and RANDBET from the main paper generalize well.

G.7 Other Architectures

Tab. 14 shows results on CIFAR10 using ResNet-20 and ResNet-50. We note that, in both cases, we use group normalization (GN) instead of batch normalization (BN) as outlined in Sec. G.1. ResNet-50, in particular, suffers from using GN due to the significant depth: the clean Err reduces from 3.67% to 6.81% in Tab. 7. Nevertheless, CLIPPING and RANDBET remain effective against random bit errors, even for higher bit error rates of $p = 1.5\%$. This is triking as ResNet-50 consists of roughly 23.5Mio weights, compared to 5.5Mio of the used SimpleNet in the main paper.

G.8 Summary Results

Fig. 11 summarizes our results: In contrast to Fig. 7, we consider individual CLIPPING and RANDBET models instead of focusing on the best results per bit error rate p. Additionally, we show our complete results for lower precisions, i.e., $m = 4, 3, 2$ on CIFAR10 and MNIST. Note that these

results, in tabular form, are included in Tab. 18 to 21. Moderate CLIPPING, e.g., using $w_{\text{max}} = 0.15$ on CIFAR10 (in red), has negligible impact on clean Err (i.e., $p = 0$ on the x-axis) while improving robustness beyond $p = 0.1\%$ bit error rate. Generally, however, higher robustness is obtained at the cost of increased clean Err, e.g., for $w_{\text{max}} = 0.05$ (in blue). Here, it is important to note that in low-voltage operation, only RErr matters – clean Err is only relevant for voltages higher than V_{min}. RANDBET further improves robustness for high bit error rates, while continuing to increase clean Err slightly. For example, RANDBET with $w_{\text{max}} = 0.05$ and trained with $p = 2\%$ bit errors increases clean Err to 5.42% but is also able to keep RErr below 7% up to $p = 1\%$ bit error rate (in orange). Reducing precision generally increases Err and RErr, especially for $m = 2$ bit. Here, our simple fixed-point quantization scheme is clearly limited compared to state-of-the-art. Nevertheless, even for $m = 2$ bits, RANDBET (violet or orange) is able to keep RErr low until roughly $p = 0.1\%$ bit error rate. Note that for $m = 2$, more aggressive clipping generally helps during training and, thus, also reduces clean Err (cf. $w_{\text{max}} = 0.1$ and $w_{\text{max}} = 0.05$ in red and blue).

Similar trade-offs can be observed on CIFAR100 and MNIST. On CIFAR100, we see that task difficulty also reduces the bit error rate that is tolerable without significant increase it RErr. Here, $p = 0.1\%$ increases RErr by more than 3%, even with RANDBET (and weight clipping). Furthermore, CIFAR100 demonstrates that CLIPPING and RANDBET are applicable to significantly larger architectures such as Wide ResNets with. On MNIST, in contrast, bit error rates of up to $p = 20\%$ are easily possible. At such bit error rates, the benefit of RANDBET is extremely significant as even $\text{CLIPPING}_{0.025}$ exhibits very high RErr of 32.68% at $p = 20\%$, cf. Tab. 21.

Table 18: **Overall Robustness Results on CIFAR10.** Tabular results corresponding to Fig. 7 and 11 for $m = 8$ and $m = 4$ bits. We show RErr for NORMAL, CLIPPING and RANDBET with various $w_{\max}$ and p across a subset of test bit error rates.

CIFAR10: summary results for $\mathbf{m = 8}$ and $\mathbf{m = 4}$ bit										
	Model	Err in %	RErr in %, p in % p=0.01							
			0.01	0.05	0.1	0.5	1	1.5	2	2.5
$m = 8$ bit	NORMAL	4.36	4.82	5.51	6.37	24.76	72.65	87.40	89.76	90.15
	RQUANT	4.32	4.60	5.10	5.54	11.28	32.05	68.65	85.28	89.01
	CLIPPING$_{0.25}$	4.58	4.84	5.29	5.71	10.52	27.95	62.46	82.61	88.08
	CLIPPING$_{0.2}$	4.63	4.91	5.28	5.62	8.27	18.00	53.74	82.02	88.27
	CLIPPING$_{0.15}$	4.42	4.66	5.01	5.31	7.81	13.08	23.85	42.12	61.20
	CLIPPING$_{0.1}$	4.82	5.04	5.33	5.58	6.95	8.93	12.22	17.80	27.02
	CLIPPING$_{0.05}$	5.44	5.59	5.76	5.90	6.53	7.18	7.92	8.70	9.56
	CLIPPING$_{0.025}$	7.10	7.20	7.32	7.40	7.82	8.18	8.43	8.74	–
	RANDBET$_1$ p=0.01	4.56	4.93	5.50	6.06	14.14	66.07	86.86	89.80	90.35
	RANDBET$_1$ p=0.1	4.50	4.80	5.27	5.72	10.33	41.10	75.90	86.52	89.03
	RANDBET$_1$ p=1	7.38	7.69	8.17	8.58	11.10	14.90	21.08	41.11	71.09
	RANDBET$_{0.2}$ p=0.01	4.44	4.67	5.09	5.48	8.64	17.97	41.53	68.95	82.48
	RANDBET$_{0.2}$ p=0.1	4.51	4.73	5.07	5.39	7.99	19.21	54.94	80.12	86.55
	RANDBET$_{0.2}$ p=1	5.46	5.68	5.97	6.20	7.63	9.47	12.38	21.47	50.86
	RANDBET$_{0.15}$ p=0.01	4.64	4.87	5.17	5.45	7.54	15.83	54.07	81.41	86.75
	RANDBET$_{0.15}$ p=0.1	4.86	5.07	5.36	5.64	7.74	12.33	22.38	40.09	60.78
	RANDBET$_{0.15}$ p=1	5.27	5.44	5.68	5.88	7.11	8.63	11.13	27.74	64.97
	RANDBET$_{0.1}$ p=0.01	4.99	5.15	5.39	5.62	6.93	9.01	12.83	22.81	41.04
	RANDBET$_{0.1}$ p=0.1	4.72	4.92	5.15	5.37	6.74	8.53	11.40	15.97	23.59
	RANDBET$_{0.1}$ p=1	4.90	5.05	5.26	5.43	6.36	7.41	8.65	12.25	27.21
	RANDBET$_{0.1}$ p=1.5	5.53	5.67	5.87	6.03	6.84	7.76	8.80	10.03	11.68
	RANDBET$_{0.1}$ p=2	5.71	5.87	6.07	6.22	7.00	7.83	8.69	9.70	10.91
	RANDBET$_{0.05}$ p=0.1	5.32	5.41	5.59	5.72	6.34	6.96	7.62	8.28	9.13
	RANDBET$_{0.05}$ p=1	5.24	5.36	5.50	5.60	6.18	6.73	7.26	7.88	8.49
	RANDBET$_{0.05}$ p=1.5	5.62	5.71	5.84	5.95	6.50	7.02	7.52	7.97	8.51
	RANDBET$_{0.05}$ p=2	5.42	5.55	5.68	5.78	6.26	6.71	7.13	7.58	8.02
	RANDBET$_{0.025}$ p=1	6.78	6.88	7.00	7.08	7.46	7.75	8.02	8.24	8.47
	RANDBET$_{0.025}$ p=1.5	6.89	6.99	7.11	7.19	7.58	7.94	8.26	8.52	8.77
	RANDBET$_{0.025}$ p=2	6.93	7.02	7.12	7.20	7.57	7.87	8.11	8.33	8.58
	RANDBET$_{0.025}$ p=2.5	6.91	6.99	7.08	7.14	7.50	7.83	8.10	8.36	8.63
$m = 4$ bit	RQUANT	4.83	5.29	5.98	6.59	15.72	50.45	79.86	87.17	89.47
	CLIPPING$_{0.25}$	4.78	5.16	5.75	6.26	12.08	30.62	60.52	80.07	87.01
	CLIPPING$_{0.2}$	4.90	5.20	5.65	6.04	9.67	27.24	63.96	82.63	87.21
	CLIPPING$_{0.15}$	4.78	5.07	5.43	5.79	8.40	14.61	28.53	50.83	70.32
	CLIPPING$_{0.1}$	5.29	5.49	5.75	5.99	7.71	10.62	15.79	24.97	37.94
	CLIPPING$_{0.05}$	5.78	5.92	6.08	6.21	6.98	7.86	8.77	9.76	11.04
	RANDBET$_{0.2}$ p=0.01	5.14	5.42	5.85	6.23	10.44	23.84	49.25	73.35	83.16
	RANDBET$_{0.2}$ p=0.1	4.77	5.01	5.41	5.76	8.66	16.06	32.40	56.69	75.21
	RANDBET$_{0.2}$ p=1	6.27	6.52	6.86	7.12	8.78	11.33	15.17	21.43	32.19
	RANDBET$_{0.15}$ p=0.01	4.88	5.13	5.54	5.92	8.51	14.21	26.26	46.02	66.13
	RANDBET$_{0.15}$ p=0.1	4.50	4.72	5.05	5.36	7.58	14.12	43.00	76.28	85.54
	RANDBET$_{0.15}$ p=1	5.99	6.18	6.45	6.65	8.00	9.74	12.50	16.73	24.09
	RANDBET$_{0.1}$ p=0.01	5.07	5.29	5.58	5.83	7.54	10.46	15.34	24.63	39.76
	RANDBET$_{0.1}$ p=0.1	4.82	5.04	5.32	5.53	6.82	8.85	12.48	21.36	40.03
	RANDBET$_{0.1}$ p=1	5.39	5.55	5.77	5.96	7.04	8.34	9.77	11.85	14.91
	RANDBET$_{0.05}$ p=0.1	5.14	5.26	5.46	5.61	6.38	7.19	8.06	9.16	10.46
	RANDBET$_{0.05}$ p=1	5.60	5.71	5.85	5.97	6.54	7.10	7.68	8.28	8.99
	RANDBET$_{0.05}$ p=1.5	5.51	5.64	5.77	5.87	6.38	6.98	7.51	8.10	8.72
	RANDBET$_{0.05}$ p=2	5.49	5.62	5.77	5.90	6.43	6.99	7.53	8.06	8.62

Table 19: **Overall Robustness Results on CIFAR10. Overall Robustness Results on CIFAR10.** Continued from Tab. 18; tabular results corresponding to Fig. 7 and 11 for $m = 3$ and $m = 2$ bits. We show RErr for NORMAL, CLIPPING and RANDBET with various $w_{\max}$ and p across a subset of test bit error rates.

CIFAR10: summary results for m = 3 and m = 2 bit										
	Model	Err in %	RErr in %, p in % p=0.01							
			0.01	0.05	0.1	0.5	1	1.5	2	2.5
$m = 3$ bit	RQUANT	79.59	83.95	88.57	91.07	96.15	97.81	98.20	98.60	99.07
	CLIPPING$_{0.25}$	6.89	7.34	8.00	8.65	14.46	28.70	53.64	75.51	85.13
	CLIPPING$_{0.2}$	5.82	6.21	6.79	7.30	11.90	23.31	43.00	65.68	78.79
	CLIPPING$_{0.15}$	5.84	6.16	6.60	6.95	9.95	15.92	27.84	47.54	67.08
	CLIPPING$_{0.1}$	5.71	6.01	6.39	6.73	8.99	13.06	20.88	35.13	51.76
	CLIPPING$_{0.05}$	5.61	5.78	6.01	6.19	7.07	8.13	9.34	10.95	13.16
	RANDBET$_{0.2}$ p=0.01	5.72	6.14	6.77	7.30	12.84	26.46	50.52	72.46	83.09
	RANDBET$_{0.2}$ p=0.1	6.23	6.55	7.04	7.53	11.38	21.36	41.93	65.54	79.94
	RANDBET$_{0.2}$ p=1	7.61	7.84	8.20	8.52	10.30	12.82	16.65	21.81	29.64
	RANDBET$_{0.15}$ p=0.01	5.61	5.94	6.40	6.77	9.59	15.72	28.06	46.88	64.39
	RANDBET$_{0.15}$ p=0.1	5.33	5.56	5.99	6.33	9.01	14.06	23.44	40.36	59.92
	RANDBET$_{0.15}$ p=1	7.26	7.52	7.82	8.07	9.58	11.47	13.87	17.58	23.01
	RANDBET$_{0.1}$ p=0.01	5.13	5.41	5.72	6.00	8.06	11.25	17.22	26.96	42.72
	RANDBET$_{0.1}$ p=0.1	5.69	5.96	6.26	6.51	8.04	10.81	15.51	23.88	37.52
	RANDBET$_{0.1}$ p=1	5.76	5.95	6.22	6.44	7.59	8.97	10.76	13.21	16.95
	RANDBET$_{0.05}$ p=0.01	5.50	5.62	5.83	5.99	6.83	7.79	9.05	10.48	12.32
	RANDBET$_{0.05}$ p=0.1	5.44	5.58	5.76	5.90	6.72	7.60	8.60	9.92	11.70
	RANDBET$_{0.05}$ p=1	5.57	5.69	5.87	6.01	6.68	7.38	8.08	8.96	9.96
$m = 2$ bit	RQUANT	88.68	89.53	91.62	93.23	97.74	98.40	97.85	99.20	98.74
	CLIPPING$_{0.25}$	90.14	90.54	91.13	91.82	95.96	96.90	97.21	96.66	97.12
	CLIPPING$_{0.2}$	82.00	84.86	90.79	94.17	97.25	96.69	97.16	97.73	97.01
	CLIPPING$_{0.15}$	14.62	15.29	16.30	17.16	22.88	33.18	50.86	71.17	84.30
	CLIPPING$_{0.1}$	7.87	8.29	8.93	9.57	13.95	23.65	42.43	64.65	80.89
	CLIPPING$_{0.05}$	6.59	6.78	7.05	7.26	8.55	10.26	12.73	15.99	20.51
	CLIPPING$_{0.025}$	6.94	7.06	7.23	7.34	7.96	8.57	9.16	9.77	10.47
	RANDBET$_{0.05}$ p=0.01	6.00	6.21	6.47	6.66	7.88	9.51	11.53	14.99	19.60
	RANDBET$_{0.05}$ p=0.1	5.83	6.04	6.30	6.52	7.73	9.32	11.41	14.49	19.77
	RANDBET$_{0.025}$ p=0.01	6.93	7.07	7.24	7.37	8.05	8.65	9.23	9.72	10.43
	RANDBET$_{0.025}$ p=0.1	7.02	7.13	7.31	7.41	7.98	8.48	9.00	9.65	10.32
	RANDBET$_{0.025}$ p=1	7.10	7.23	7.38	7.49	8.10	8.65	9.14	9.54	10.07

Table 20: **Overall Robustness Results on CIFAR100.** Tabular results corresponding to Fig. 7 and 11 for $m = 8$. We show RErr for NORMAL, CLIPPING and RANDBET with various $w_{\max}$ and p across a subset of test bit error rates.

CIFAR100: summary results for m = 8 bit							
Model	Err in %	RErr in %, p in % p=0.01					
		0.005	0.01	0.05	0.1	0.5	1
NORMAL	18.21	19.84 ±0.16	20.50 ±0.25	25.05 ±0.94	32.39 ±1.89	97.49 ±0.95	99.10 ±0.19
RQUANT	18.53	19.46 ±0.13	19.95 ±0.16	22.68 ±0.63	25.90 ±1.01	87.24 ±3.99	98.77 ±0.31
CLIPPING$_{0.25}$	18.88	19.76 ±0.1	20.11 ±0.11	21.89 ±0.18	23.74 ±0.35	62.25 ±4.51	96.62 ±1.22
CLIPPING$_{0.2}$	18.64	19.36 ±0.09	19.71 ±0.1	21.33 ±0.23	23.07 ±0.38	49.79 ±4.21	94.02 ±2.38
CLIPPING$_{0.15}$	19.41	20.00 ±0.08	20.24 ±0.09	21.68 ±0.17	23.02 ±0.3	37.85 ±2.03	79.45 ±5.08
CLIPPING$_{0.1}$	20.31	20.86 ±0.07	21.09 ±0.09	22.14 ±0.17	23.10 ±0.21	31.78 ±1.15	51.71 ±3.47
CLIPPING$_{0.05}$	21.82	22.16 ±0.05	22.29 ±0.06	22.94 ±0.13	23.46 ±0.18	26.86 ±0.46	31.47 ±0.79
RANDBET$_{0.1}$ p=0.01	19.68	20.21 ±0.08	20.46 ±0.09	21.52 ±0.17	22.56 ±0.25	30.59 ±0.82	48.93 ±3.31
RANDBET$_{0.1}$ p=0.05	19.94	20.47 ±0.06	20.69 ±0.08	21.72 ±0.16	22.60 ±0.23	29.93 ±0.86	46.76 ±3.46
RANDBET$_{0.1}$ p=0.1	19.18	19.67 ±0.06	19.86 ±0.07	20.87 ±0.12	21.69 ±0.21	28.03 ±0.74	41.29 ±2.81
RANDBET$_{0.1}$ p=0.5	19.90	20.24 ±0.05	20.41 ±0.07	21.17 ±0.13	21.83 ±0.17	25.66 ±0.48	31.55 ±0.95
RANDBET$_{0.1}$ p=1	21.08	21.43 ±0.05	21.59 ±0.07	22.24 ±0.13	22.76 ±0.15	25.73 ±0.33	29.31 ±0.56
RANDBET$_{0.05}$ p=0.01	21.86	22.17 ±0.06	22.31 ±0.05	23.00 ±0.14	23.57 ±0.2	26.84 ±0.46	31.33 ±0.79
RANDBET$_{0.05}$ p=0.05	20.97	21.30 ±0.05	21.44 ±0.07	22.12 ±0.14	22.72 ±0.16	25.95 ±0.34	30.14 ±0.59
RANDBET$_{0.05}$ p=0.1	21.22	21.53 ±0.05	21.66 ±0.05	22.29 ±0.12	22.81 ±0.17	25.88 ±0.39	29.93 ±0.83
RANDBET$_{0.05}$ p=0.5	21.29	21.55 ±0.04	21.65 ±0.06	22.13 ±0.12	22.60 ±0.15	25.01 ±0.3	27.70 ±0.5
RANDBET$_{0.05}$ p=1	20.83	21.08 ±0.04	21.20 ±0.06	21.73 ±0.13	22.16 ±0.13	24.33 ±0.24	26.49 ±0.38

Table 21: **Overall Robustness Results on MNIST.** Tabular results corresponding to Fig. 7 and 11 for $m = 8, 4, 2$ bits. We show RErr for NORMAL, CLIPPING and RANDBET with various $w_{\max}$ and p across a subset of test bit error rates.

MNIST: summary results for $\mathbf{m = 8, 4, 3}$ bit									
	Model	Err	RErr in %, p in % p=0.01						
		in %	1	5	10	12.5	15	17.5	20
$m = 8$ bit	NORMAL	0.39	0.77	86.37	89.92	89.82	89.81	90.09	90.03
	RQUANT	0.40	0.69	85.96	90.20	89.86	90.10	89.72	89.83
	CLIPPING$_{0.1}$	0.39	0.48	18.21	88.93	90.35	90.06	90.56	90.18
	CLIPPING$_{0.05}$	0.42	0.47	0.63	8.67	51.38	80.64	87.79	89.57
	CLIPPING$_{0.025}$	0.43	0.47	0.56	0.71	0.95	1.81	7.22	32.68
	RANDBET$_{0.1}$ p=1	0.36	0.44	3.41	86.29	89.05	89.85	90.10	89.93
	RANDBET$_{0.05}$ p=1	0.34	0.39	0.59	8.92	51.32	79.35	87.63	89.15
	RANDBET$_{0.05}$ p=5	0.34	0.38	0.50	1.02	5.12	41.31	79.19	87.88
	RANDBET$_{0.05}$ p=10	0.40	0.43	0.51	0.67	0.86	1.74	9.77	47.58
	RANDBET$_{0.05}$ p=15	0.39	0.40	0.45	0.56	0.64	0.78	1.10	2.72
	RANDBET$_{0.05}$ p=20	0.39	0.42	0.48	0.53	0.57	0.63	0.74	0.94
$m = 4$ bit	RQUANT	0.36	0.72	87.21	90.23	90.01	89.88	89.97	89.67
	CLIPPING$_{0.1}$	0.38	0.51	38.75	88.33	89.47	89.57	90.10	89.67
	CLIPPING$_{0.05}$	0.31	0.39	0.78	44.15	78.64	87.32	89.03	89.71
	CLIPPING$_{0.025}$	0.37	0.41	0.50	0.67	0.99	4.63	29.46	67.21
	RANDBET$_{0.1}$ p=1	0.38	0.48	13.29	87.43	89.70	89.63	89.41	90.02
	RANDBET$_{0.1}$ p=5	0.38	0.48	0.78	24.73	74.88	87.04	88.72	89.55
	RANDBET$_{0.1}$ p=10	0.40	0.47	0.64	1.22	2.62	16.72	64.33	83.80
	RANDBET$_{0.1}$ p=15	0.56	0.59	0.73	1.03	1.28	1.87	3.71	14.39
	RANDBET$_{0.1}$ p=20	0.56	9.48	14.29	7.39	6.07	5.80	6.10	8.12
	RANDBET$_{0.05}$ p=1	0.37	0.43	0.67	36.99	77.12	85.97	88.62	89.94
	RANDBET$_{0.05}$ p=5	0.38	0.42	0.53	1.38	12.90	60.73	83.69	88.75
	RANDBET$_{0.05}$ p=10	0.34	0.39	0.47	0.65	0.91	2.11	19.15	71.25
	RANDBET$_{0.05}$ p=15	0.37	0.39	0.43	0.52	0.63	0.79	1.17	3.16
	RANDBET$_{0.05}$ p=20	0.44	0.48	0.53	0.60	0.65	0.72	0.86	1.04
$m = 2$ bit	CLIPPING$_{0.1}$	0.47	3.82	89.19	89.92	90.22	90.14		
	CLIPPING$_{0.05}$	0.41	0.62	77.19	89.47	90.40	90.06		
	RANDBET$_{0.05}$ p=3	0.47	0.53	1.36	82.71	88.66	90.28		
	RANDBET$_{0.05}$ p=5	0.40	0.49	0.77	25.72	78.71	88.22		
	RANDBET$_{0.05}$ p=10	0.40	0.45	0.58	0.94	1.82	15.70		
	RANDBET$_{0.05}$ p=15	0.46	0.51	0.60	0.77	0.91	1.21		