SNR: Squeezing Numerical Range Defuses Bit Error Vulnerability Surface in Deep Neural Networks

ELBRUZ OZEN and ALEX ORAILOGLU, University of California, San Diego, USA

As deep learning algorithms are widely adopted, an increasing number of them are positioned in embedded application domains with strict reliability constraints. The expenditure of significant resources to satisfy performance requirements in deep neural network accelerators has thinned out the margins for delivering safety in embedded deep learning applications, thus precluding the adoption of conventional fault tolerance methods. The potential of exploiting the inherent resilience characteristics of deep neural networks remains though unexplored, offering a promising low-cost path towards safety in embedded deep learning applications. This work demonstrates the possibility of such exploitation by juxtaposing the reduction of the vulnerability surface through the proper design of the quantization schemes with shaping the parameter distributions at each layer through the guidance offered by appropriate training methods, thus delivering deep neural networks of high resilience merely through algorithmic modifications. Unequaled error resilience characteristics can be thus injected into safety-critical deep learning applications to tolerate bit error rates of up to 10% at absolutely zero hardware, energy, and performance costs while improving the error-free model accuracy even further.

CCS Concepts: • **Computing methodologies** → **Neural networks**; • **Hardware** → **Fault tolerance**; **Safety critical systems**; • **Computer systems organization** → **Reliability**;

Additional Key Words and Phrases: Error resilience, neural network quantization, neural network regularization

ACM Reference format:

Elbruz Ozen and Alex Orailoglu. 2021. SNR: Squeezing Numerical Range Defuses Bit Error Vulnerability Surface in Deep Neural Networks. *ACM Trans. Embedd. Comput. Syst.* 20, 5s, Article 76 (September 2021), 25 pages.

<https://doi.org/10.1145/3477007>

1 INTRODUCTION

We have witnessed significant progress in practical machine intelligence in the last decade. Deep neural networks offer a competitive precision in computer vision [20, 25, 40], speech recognition [76], and natural language processing [2, 12] tasks while enabling novel opportunities in various embedded application domains such as autonomous driving, healthcare devices, voice-assistants, robotics, intelligent sensors, and smart-city applications.

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2021.

Authors' address: E. Ozen, and A. Orailoglu, University of California, San Diego, 9500 Gilman Dr., La Jolla, California, USA, 92093; emails: elozen@eng.ucsd.edu, alex@cs.ucsd.edu.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2021 Copyright held by the owner/author(s).

1539-9087/2021/09-ART76

<https://doi.org/10.1145/3477007>

Deep learning applications are known to be resource-intensive; delivering the desired performance requirements within the designated resource budgets (i.e., area, power) has therefore been an ongoing challenge in embedded designs. The increasing computational cost of deep neural networks has been tackled mainly through the design of more efficient deep learning models [25, 67], model compression methods such as pruning [18, 21, 54, 70] and quantization [8, 9], and domain-specific accelerator design [5, 66] in the prior literature.

Both performance and energy efficiency constitute fundamental design considerations for embedded deep learning applications; strict functional safety constraints are frequently imposed furthermore in various practical application domains such as autonomous driving, healthcare devices, robotics, and industrial control systems. While operating in the field, exposure to harsh environmental conditions (e.g., high-energy particles, high temperature) over long durations increases the likelihood of hardware errors in these systems considerably. A hardware-caused error can impact functionality with potentially catastrophic consequences; thus, additional electronics safety mechanisms are an absolute requirement in these application domains.

Fault tolerance methods for digital electronics have been a focus of intensive scrutiny since the inception of our electronic era. Well-established fault tolerance methods that are frequently employed in safety-critical electronic designs, ranging from modular redundancy techniques [1] to error correction codes [17], necessitate significant redundancy and result in noticeable area and power overheads for safety-critical systems. As a result, their applicability is often limited in resource-constrained embedded system designs.

Error tolerance is a challenging problem in the context of general-purpose computing, as imposed safety mechanisms often remain agnostic to the possible resilience characteristics of the application. As a result, a worst-case approach ascribes even a minute change in the program variables to potential data corruption, necessitating the taking of further actions, such as correction or program re-execution, to assure the correctness of system behavior. On the other hand, the inherent resilience properties of emerging deep neural network applications promise novel opportunities for tackling the functional safety problem in embedded deep learning applications at no additional cost. As long as it does not lead to a misclassification at the neural network outputs, an error in the neural network variables can be classified as non-critical. Furthermore, neural networks are known to endure mild and bounded inaccuracies gracefully, including numerical errors introduced in the quantization process. Similarly, limited noise effects are well tolerated despite the fact that nearly all neural network variables are impacted to a certain degree by such perturbation patterns [29].

The described resilience characteristics of neural networks fall unfortunately short of conclusively establishing the safety of deep learning applications. Previous studies [33, 48, 56] have repeatedly demonstrated that models often start to lose accuracy even at low bit error rates (i.e., 10^{-6}). The impact of hardware errors is bound to be substantial if a numerically significant bit position is impacted. Moreover, if the numerical range of variables in the executed model does not fully utilize the allocated numerical range in the employed number representation scheme, the numerical effect of the potential bit errors can easily exceed the expected scale of the original neural network variables by a few orders of magnitude even. As a result, even a small quantity of such large error patterns can substantially alter neural network outputs and thus critically impair system accuracy.

The large numerical range in the number representations constitutes a significant source of vulnerability against critical bit error effects in deep neural networks. First, we observe that the outlined vulnerability can be alleviated to a great extent through layer-wise quantization techniques by tightening the quantization margins to match the utilized range at each layer. Furthermore, we notice that the numerical distribution of model parameters at each layer often contains *outlier* val-

ues which can significantly exceed the rest of the distribution. The stretched quantization range imposed by such outlier variables exposes the neural network model to large magnitude errors, with deleterious consequences for model accuracy. We propose a novel regularization method, *outlier regularization*, in the training process to tighten the numerical range and cluster the parameter distributions, thus curbing the impact of potential bit errors more effectively. As a result of the combined employment of layer-wise quantization and outlier regularization techniques, deep neural networks can inherently tolerate extreme bit error rates (up to 10%) with no reliance on additional error tolerance mechanisms and at essentially no additional cost.

Experimental analysis reveals yet another surprising phenomenon in our studies, as the models trained with the proposed outlier penalty terms consistently exhibit a noticeably higher accuracy than the baseline models in both full-precision and quantized formats due to the model regularization effect in training and improved quantization quality.

This manuscript starts off with a brief survey of related studies on neural network resilience in Section 2. An introductory overview of model quantization is presented in Section 3. Section 4 discusses the design choices and our novel regularization technique to improve the error resilience of deep neural networks. A detailed experimental analysis is provided in Section 5. Finally, we discuss the future implications and directions in Section 6 prior to presenting our concluding remarks in Section 7.

2 RELATED WORK

Previous literature has witnessed a flurry of works exploring the error resilience limits of deep neural networks through error simulation [28, 32, 33, 47, 48, 56, 59, 72] and physical error injection experiments [13, 42]. Various software frameworks have also been made publicly available [7, 34, 43] for reliability analysis in neural networks. Software-level error simulation techniques have become a particularly promising path for performing quick deep neural network reliability analysis in the listed studies to avoid significant run-time and computational costs of hardware-level simulations. Alternatively, the relative vulnerability of neural network variables can be estimated through mathematical saliency models, including gradient-based sensitivity analysis techniques [10], or the LRP (layer-wise relevance propagation) metric [45, 61] discussed in the literature.

Errors in deep learning hardware can be identified through various novel approaches, including symptom-based error detector [33], algorithm-based fault tolerance techniques [19, 49, 53, 71], dedicated error detection networks [62], and learned error checking invariants in training [51]. Moreover, improved error resilience characteristics can be attained in deep learning applications through activation range restriction [6, 24], continuous error filtering [50], anomaly detection and suppression [52], fault-tolerant model training [16], and fault-tolerant neural architecture search [35, 60], as demonstrated in previous studies. Prior experimental work [23, 59] has demonstrated that compressed models, particularly binary neural networks, exhibit high error resilience characteristics.

The improved resilience characteristics of deep neural networks can be coupled with hardware optimizations to reduce the energy consumption of deep learning accelerators through SRAM supply voltage scaling [3, 57], reliability-aware adaptive voltage swing scaling in the communication fabric [44], and reducing the operating voltage in the MAC (multiply-accumulate) units [74].

Deep learning models can further tolerate permanent hardware faults if the defective device is reconfigured and the executed model is fine-tuned with the underlying defect information [75].

ReRAM-based analog deep learning accelerators have been widely investigated to improve the accuracy of the models under device manufacturing defects, programming variations, and analog domain noise. The impact of permanent manufacturing defects and resistance variations in ReRAM devices can be tackled through adaptive training methods [4, 36]. Similarly, the noise resilience of

deep learning algorithms can be enhanced through injecting noise perturbations in the training process [22, 29, 41, 58]. Algorithm-based fault tolerance methods [37], arithmetic codes [14], and error-correcting output encoding schemes [39] have been proposed in the literature to enable online error detection/correction, and enhance resilience characteristics of the neural networks against ReRAM inaccuracies. Effective ReRAM device testing methods are presented in [38]. While the outlined studies on ReRAM primarily focus on neural network inference, ReRAM on-chip training methods can further benefit from fault tolerance techniques [69].

A comprehensive survey of the challenges and the current trends in robust machine learning systems is presented in [63].

While our approach bears similarities to activation range restriction methods [6, 24], it differs in that we embed such restrictions indirectly through the layer-wise quantization process instead of employing explicit range limitations. As a result, the restrictions are implemented implicitly without necessitating any additional operations. Moreover, we perform range restrictions on the model parameters in a similar manner through layer-wise quantization, in addition to the activations. The effectiveness of parameter range reductions is shown to be quite substantial against model parameter (i.e., weight, bias) errors, which are the primary threat to system safety due to their permanent impact on model accuracy. The remarkable benefit of this approach is also validated through experimental comparison with [24]. Yet again, our approach differs in that we actively shape the parameter distributions through dedicated regularization terms, attenuate the scale of bit error effects, and boost the error resilience of the models remarkably further.

3 OVERVIEW OF MODEL QUANTIZATION

Deep learning inference rarely requires the high-precision offered by the floating-point representations. Neural network variables can be represented in the fixed-point format and even quantized to smaller bit widths (i.e., 8-bit and often less) with small or even imperceptible impact on model accuracy. Quantized models offer a noticeable reduction in memory footprint and a significant decrease in energy consumption due to reduced data movement. Furthermore, smaller fixed-point arithmetic units that are faster and more efficient than their floating-point counterparts [66] cut down the processing cost and boost system performance further. Quantization is a commonly employed technique in deep learning inference, including embedded neural network accelerators, because of the outlined benefits.

The quantization process maps a given continuous distribution into a discrete range through a linear projection of the variables and rounding the final result to the nearest quantization point. A quantization scheme is termed *symmetric* if the representation is symmetric around zero. Namely, a symmetric quantization scheme can be considered as a function which maps a floating-point range into a fixed-point range (Equation (1)) merely through scaling and rounding (Equation (2)):

$$[-m_{float}, m_{float}] \rightarrow [-m_{fixed}, m_{fixed}] \quad (1)$$

$$x_{fixed} = \text{round} \left(\frac{m_{fixed}}{m_{float}} \times x_{float} \right) \quad (2)$$

With knowledge of the scale factor (m_{float}/m_{fixed}), the quantized variables can to a large extent be restored to their original form. A simplistic fixed-point representation may consist of a mantissa and utilize a single global scale factor. A scheme with a single global scale factor, while straightforward, may significantly exceed the minimal quantization error, failing to display a competitive model accuracy as deep neural network variables often span diverse dynamic ranges at each layer. The diversity of the dynamic ranges can be observed in an example shown in Figure 1 for the weight distributions of the ResNet-18 model [20] trained on the CIFAR10 dataset [30]. The global

Fig. 1. Weight distribution across layers (ResNet-18 model - CIFAR10 dataset).

Fig. 2. Sample weight distribution in a neural network layer (ResNet-18 model - CIFAR10 dataset).

range may not be covered with a small *quantization step size*;¹ on the other hand, employing a larger quantization step size might incur noticeable information loss and degrade model accuracy.

A quantization scheme with unique scale factors for a group of variables [11, 68], i.e., unique scale factors per layer, and distinct scale factors for weights and activations, can express values from quite diverse distributions accurately, improve quantization quality, and consequently allow model accuracy maintenance even at low bit-width representation.

Moreover, the weights and activations at each layer usually follow a distribution that embeds *outlier* values of significantly higher magnitude than the rest. An illustrative distribution pattern is shown in Figure 2 for the weights of the first convolutional layer in the ResNet-18 model.

Large-magnitude variables may necessitate an increase in the quantization step size to accommodate the entire numerical range; otherwise, such values need to be clipped. Unfortunately, both increases in quantization step size and clipping large values can each lead to a larger quantization error and consequently reduce quantization quality. Solutions such as activation clipping have been proposed [8, 9] to partially address this problem in the context of activation quantization.

¹Quantization step size refers to the numerical distance between two adjacent quantization points. This distance is constant between two adjacent points in uniform quantization schemes.

4 DESIGNING ERROR-RESILIENT DEEP NEURAL NETWORKS BY TIGHTENING NUMERICAL RANGE

This section focuses on the represented numerical range in neural network layers and its significant impact on the error resilience of deep neural networks. For starters, we expound the dependence of the numerical range at each layer on the characteristics of the employed quantization methods and the dramatic improvements in system resilience that can be garnered through proper design choices in the quantization schemes. We proceed to propose a practical approach to shape the numerical distribution of the neural network parameters with proper regularization methods so that the numerical impact of bit errors becomes less pronounced in the quantized form of the obtained parameter distributions.

4.1 Tight Quantization Bounds With Layer-wise Quantization

Deep neural network layers span diverse numerical ranges at each layer; therefore, the accuracy of the quantization scheme can be improved through layer-wise quantization, which employs unique scale factors at each layer. Although this is a common practice for improving the precision of model quantization, the error resilience and reliability implications for layer-wise quantization have not been investigated in the literature. Our analysis demonstrates that the use of unique scale factors at each layer not only yields benefits in terms of quantized model accuracy but also boosts the bit error resilience of deep neural networks rather noticeably.

While the number of allocated hardware bits is fixed on the hardware for a variable, the same binary representation can span diverse numerical ranges at each layer in a layer-wise quantization scheme because of the scale factors. The proper range adjustment with the layer-wise scale factors can guarantee that the parameter distributions at each layer fully span the provided binary representation on the hardware (e.g., from -128 to 127 for an 8-bit binary representation). As a result, the numerical impact of the hardware bit errors is guaranteed to be within the layer distribution and not significantly larger than the layer parameters.

The impact of quantization range on model resilience can be explained with an illustrative example. The maximum weight magnitude in the last convolutional layer in Figure 1 is observed to hover around 0.25 , and the global maximum weight value comes in at slightly above 4 . If the last convolutional layer is represented with a quantization scheme that covers the entire global range, the numerical impact of bit errors becomes proportional to $(4/2^{b-1}) * (2^i)$ where b is the quantization bit-width, and i is the erroneous bit position. It can be seen that the error magnitude could easily exceed the expected distribution of the weights in the last convolutional layer, even for errors manifesting at the least significant bit positions. On the other hand, the numerical error magnitude can not exceed $(0.25/2^{b-1}) * (2^i)$ for the last convolutional layer if a layer-wise quantization scheme is applied. The error magnitudes would often be smaller than the original layer weights in the layer-wise quantization scheme, leading to such muted error effects being tolerated better by deep learning models, which are known to exhibit resilience against minor and limited-magnitude perturbations.

Alternatively, the described effect of the layer-wise scaling factors can be considered as boosting the **signal-to-noise ratio (SNR)** at each layer by guiding original layer variables to span the provided fixed-point representation range fully through the proper scaling factors. As a result, the numerical impact of bit errors is diminished in comparison to the value of the original neural network variables.

The described scheme can be applied to both model weights and activations with dedicated scale factors at each layer to improve the resilience of the model against the bit errors in the weights and the activations.

4.2 Squeezing Layer-wise Bounds With Outlier Regularization

Layer-wise quantization improves model resilience by precluding error effects that dwarf the actual variable distribution at each layer. However, we observe that the distribution of the variables at deep neural network layers frequently contains large magnitude variables, *outliers*, that increase the numerical range, thus exposing the model to error effects that can dwarf a large fraction of the distribution.

When the quantization step size is increased to accommodate such large magnitude variables, the ratio of a non-outlier variable magnitude compared to the quantization step greatly diminishes. We have previously concluded that the numerical impact of bit errors is proportional to the quantization step size ($2^i \times \text{step_size}$) for impacted bit position i ; therefore, the bit error effects can grow disproportionately when compared to the magnitude distribution of non-outlier variables in the corresponding layer.

The described problem can be alleviated by constricting the quantization range. A primitive solution could involve the setting of a smaller quantization range and the clipping of the large magnitude variables after training. However, this approach might result in an increase in the quantization error for the large magnitude variables and consequently impact model accuracy. An alternative approach might involve clipping the range during training to preclude the formation of a distribution with large magnitude variables; however, the optimal selection of the clipping threshold is rather difficult to establish when performed manually at each layer.

We utilize the flexibility of the training process to form clustered variable distributions at each layer and allow the target model to adjust the required numerical range at each layer automatically by discouraging significantly large magnitude parameters. Such modifications can be encouraged by designing additional goals for training and optimizing these goals in the learning process.

Deep neural network training is carried out by minimizing a *loss function*, which measures the similarity between the produced neural network outputs and the expected classification. For instance, a widely employed loss function, *categorical cross-entropy*, is formulated as in Equation (3) for a batch of training examples:

$$-\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(p_{i,c}) \quad (3)$$

In Equation (3), N and C denote the training batch size and the number of classification categories, respectively. The quantities, $y_{i,c}$ and $p_{i,c}$, represent the correct binary label and the produced Softmax layer output for the training example i and prediction class c . Deep neural network training is carried out by performing a forward pass to compute the loss function, then calculating the gradient of the loss function with respect to the model parameters in the backward pass, and finally updating the parameters in the direction opposite to the gradients at each step to minimize the loss function.

An additional regularization term in the cost function can penalize the outlier variables with large magnitude and shrink the numerical range of the distribution effectively at each layer. L1 or L2 norm regularization can be utilized for this purpose, but such regularization schemes are not selective and penalize all weight values. Such an approach can fail to yield the expected benefits as the resultant scale changes may retain the large scale magnitude disparateness across the variable space. Instead, we introduce a novel regularization term, *outlier regularization*, which focuses the penalty term on the large magnitude parameters so as to penalize large magnitude variables while keeping the rest of the distribution intact.

We consider two different regularization terms to encourage the desired behavior in model training. The first alternative, *max-magnitude regularization*, computes the penalty term by

accumulating the maximum magnitude values from each parameter group as in Equation (4). We treat the weight and bias values of a layer as separate groups in our implementation. In Equation (4), L denotes the total number of layers, and W_l and B_l refer to the weights and biases at layer l . The magnitude is computed element-wise, and the maximum function returns the maximum entry at each computed magnitude tensor.

$$\sum_{l=1}^L (\max(|W_l|) + \max(|B_l|)) \quad (4)$$

Max-squared regularization has a similar implementation except for the fact that we accumulate the maximum square values from each parameter group as in Equation (5) to construct the penalty term. The square function in Equation (5) is element-wise, similar to the magnitude function in Equation (4).

$$\sum_{l=1}^L (\max(W_l^2) + \max(B_l^2)) \quad (5)$$

We include the computed penalty term in the loss function, as an example demonstrates for maximum magnitude regularization in Equation (6):

$$-\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(p_{i,c}) + \alpha \sum_{l=1}^L (\max(|W_l|) + \max(|B_l|)) \quad (6)$$

We further employ in the loss function a regularization coefficient (α) to exert precise control over the penalty term. The proper selection of the regularization coefficient empowers the deep learning model to penalize the largest magnitude parameter at each group sufficiently while incurring no adverse effect on the learning process so that competitive model accuracy can be attained. The parameter distributions are expected to show a stronger clustering as a result of regularization; when such parameter distributions are quantized with a proper scale factor to cover the squeezed bounds, the magnitude of the bit errors that can be suffered would diminish vis-à-vis the value of the quantized parameters.

The reader will note that the described regularization scheme targets the model parameters (i.e., weight, bias) in the given description. Model parameters are leaf nodes in the gradient back-propagation tree for the convolutional and fully connected layers; thus, the maximum magnitude entries can be penalized with the specified regularization methods with no impact on the other weights, even those that lie within the same layer. The described independence property across the weights underpins our technique as it enables the precise manipulation of the desired parameters with no unintended consequences in the parameter distributions.

The application of regularization methods to the activations, while feasible, does introduce far more complex interactions in model training. First, activations are derived through the multiply-accumulate operations that involve the layer weights; thus, the gradients accumulated on the activations eventually end up being back-propagated and used to update the weight entries. Such indirect effects on the model weights might introduce interference in training, reduce training efficacy in forming desired parameter distributions in the target model, and diminish system reliability against weight errors. Furthermore, penalizing a particular activation at a layer inevitably affects the other activations in the same layer in case of weight sharing across the units, particularly for the convolutional layers where the filter weights are shared to construct an entire channel of the output feature map. As a result, the direct manipulation of the activations through regularization necessitates a more profound theoretical analysis. As the immediate vulnerability surface in neural

networks mainly originates from the weight errors due to their repeated use in the inference operations and their consequent long-term impact on the model accuracy, our work puts particular emphasis on squeezing the error margins for model parameters, with the regularization analysis for the activations reserved for future work.

5 EXPERIMENTAL ANALYSIS

5.1 Experimental Setup

We implement the experimental setup in PyTorch [55]. The LeNet-5 model with the MNIST dataset [31], the VGG-16 [64] and the ResNet-18 [20] models with the CIFAR10 dataset [30], and the SqueezeNet model [27] with the GTSRB dataset [65] are used for the experiments. We utilize the following hyper-parameter configurations for training the target models: the LeNet-5 model is trained for 100 epochs with the Adam optimizer starting at a learning rate of 10^{-3} , and reducing the learning rate tenfold at epochs 33 and 66. The VGG-16 and the ResNet-18 models are trained for 200 epochs with the SGD (Stochastic Gradient Descent) optimizer (momentum = 0.9) starting at a learning rate of 10^{-2} and reducing the learning rate tenfold at epoch 100. Finally, the SqueezeNet model is trained for 100 epochs with the SGD optimizer starting at a learning rate of 10^{-2} , and reducing the learning rate tenfold at epoch 50. The baseline and regularized models are trained with the same hyper-parameter configurations except for the usage of the regularization term. The value of proper regularization coefficients (α) depends on the target model and the regularization type. The utilized values are indicated in Table 1. The training outcome is not highly sensitive to minor changes in the regularization coefficients, and proper selection of these coefficients can be accomplished by starting with a relatively large value and reducing the value tenfold at every step until it is possible to train the model with a competitive accuracy. It should be noted that the utilized coefficient values for the proposed outlier regularization methods are noticeably larger than the standard regularization terms (e.g., L1 or L2) in Table 1 since only outliers are penalized with the proposed regularization terms instead of all parameters.

We employ a 2's complement fixed-point representation and a symmetric quantization scheme in the experiments. We consider the widely-adopted 8-bit data types for the weights and activations in the context of the error injection experiments and the presented reliability analysis. More aggressive quantization schemes are employed in Section 5.3.4 to demonstrate the superior accuracy retention of the regularized models in the post-training quantization experiments.

We utilize two machines for model training and error injection experiments, each with 2× Intel Xeon E5-2630 v4 CPUs, NVIDIA GeForce GTX 1080Ti GPU, and 16GB RAM.

5.2 Error Model

We design our in-house error injection framework to simulate the impact of bit errors on neural network variables. Bit errors are injected into weight and activation tensors prior to each convolutional and fully connected layer operation. Error injection is performed through a dedicated PyTorch function that receives the input tensor (weight or activation), the error probability, the data bit-width, and the maximum quantization magnitude as inputs. We indicate the portion of the erroneous bits in the weight/activation tensors with the term *bit error rate* to conform to the terminology used in previous studies [56]. The proposed framework is utilized to measure and report the classification accuracy at different bit error rates.

The bit error generation process is outlined in detail in Algorithm 1. The error injection function first generates the bit error mask through a Bernoulli distribution with the given error probability. A pseudo-quantization scheme allows us to determine the total numerical perturbation at each variable as a result of injected bit errors. Finally, the target tensor is perturbed by adding the

ALGORITHM 1: Error injection method

Input: Input tensor (I), error probability (p), data bit-width (b), maximum quantization magnitude (m)

Output: Output tensor (O)

1. Generate bit error locations through a Bernoulli distribution with probability p
 2. Perform pseudo-quantization on the input tensor (I) by using b and m
 3. Identify the original bit values in the quantized representation
 4. Compute the numerical perturbation value for each bit error location
 5. Compute overall numerical perturbations for each variable by summing numerical perturbations caused by individual bit errors
 6. De-quantize overall numerical perturbation values that are computed for each variable
 7. Add de-quantized perturbation values to the input tensor (I) to generate output tensor (O)
-

generated numerical errors to simulate the bit error effects on the variables. The reader will note that the bit error rate in the perturbed tensors will approximately equal the provided bit error probability after error injection.

While our simulation tool does not allow modeling error accumulation effects over time in the neural network tensors, model resilience against the worst-case accumulated error effects can be indirectly assessed by determining the maximum lifetime of the tensors (i.e., time duration until they are discarded or refreshed), multiplying it with the expected error probability for a unit amount of time, and finally utilizing the obtained value as the error probability in the designed error simulation framework.

Our analysis and the resilience methods focus solely on the errors in the data path. We particularly focus on the bit errors on the parameters due to their permanent effect on the model behavior and the classification accuracy. Experimental analysis for the activation errors is also provided. Protection of the control circuitry can be delivered economically with conventional hardware fault-tolerance methods, as deep learning accelerators usually necessitate minimal control circuitry due to the inherent regularity of the micro-architecture and neural network computations. In addition, our analysis omits consideration of the errors in the quantization scale factors. As the adopted layer-wise quantization schemes incur a negligible number of scale factors when compared to the model parameters, i.e., at double the number of layers if distinct scale factors are utilized for the weights and activations at each layer, their resilience can be rendered by conventional hardware fault-tolerance methods at negligible overheads as well.

5.3 Experimental Results

We investigate the effectiveness of our approach by analyzing the impact of the proposed regularization terms through the measurement of full-precision accuracy values and observing the particularities of the parameter distributions at each layer. Moreover, bit error resilience characteristics are evaluated by measuring the classification accuracy at various bit error rates. Finally, we perform post-training quantization experiments at lower bit widths to demonstrate the improvements in accuracy retention for the proposed regularization techniques as a result of the obtained parameter distribution characteristics. We compare the proposed regularization methods with the baseline case with no regularization and standard L1 and L2 regularization methods in the experiments.

5.3.1 Impact of Regularization Terms on Training and Full-precision (Non-Quantized) Model Accuracy. The training efficacy and the model accuracy are essential considerations that need to

Table 1. The Impact of Regularization on the Full-precision (non-quantized) Model Accuracy

	Regularization Coeff. (α)	Train Acc. (%)	Test Acc. (%)
LeNet-5 Baseline	-	99.71	99.24
LeNet-5 Regularized (L1)	10^{-4}	99.86	99.27
LeNet-5 Regularized (L2)	10^{-4}	100.00	99.41
LeNet-5 Regularized (Max-Magnitude)	1	99.31	99.07
LeNet-5 Regularized (Max-Squared)	1	99.97	99.30
VGG-16 Baseline	-	99.71	91.55
VGG-16 Regularized (L1)	10^{-5}	99.65	92.24
VGG-16 Regularized (L2)	10^{-5}	99.80	91.93
VGG-16 Regularized (Max-Magnitude)	10^{-1}	99.76	92.21
VGG-16 Regularized (Max-Squared)	10^{-1}	99.90	91.89
ResNet-18 Baseline	-	99.37	92.22
ResNet-18 Regularized (L1)	10^{-5}	99.86	93.57
ResNet-18 Regularized (L2)	10^{-5}	99.82	92.85
ResNet-18 Regularized (Max-Magnitude)	10^{-1}	99.76	92.90
ResNet-18 Regularized (Max-Squared)	10^{-1}	99.86	92.97
SqueezeNet Baseline	-	99.83	94.36
SqueezeNet Regularized (L1)	5×10^{-4}	99.20	94.20
SqueezeNet Regularized (L2)	10^{-3}	99.83	93.94
SqueezeNet Regularized (Max-Magnitude)	5×10^{-1}	99.79	94.47
SqueezeNet Regularized (Max-Squared)	1	99.82	94.42

be addressed for the proposed regularization techniques. Table 1 summarizes the regularization coefficients used for training (α) and the final accuracy values of the baseline and regularized models. We observe that the target deep neural network models can be trained effectively with the outlier regularization techniques and outperform the baseline model accuracy values by a clear margin, particularly in the test/validation examples. We hypothesize that the described phenomenon in the test/validation examples occurs because of reduced over-fitting and improved generalization. Table 1 demonstrates that the obtained accuracy values through the outlier regularization methods are comparable and even sometimes superior to the standard L1/L2 weight regularizers that are widely used in practice to reduce over-fitting. The slight accuracy improvement in the training examples is a more interesting phenomenon, necessitating a more comprehensive theoretical analysis for a more definitive answer.

5.3.2 Impact of Regularization Term on Parameter Distributions. The outlier regularization leads to a significant reduction in the dynamic range of the parameters at each layer and to parameter distributions that display a stronger clustering. Figures 3 and 4 demonstrate the impact of the regularization term on the model parameter distributions at each layer for the LeNet-5 and ResNet-18 models. The colored portions of the violin plots demonstrate the distribution density at a particular weight value (i.e., wider regions indicate a higher density). The minimum and maximum values at each layer are indicated through the horizontal lines attached to both ends of the distribution curves.

The influence of the proposed regularization terms (i.e., *Max-Magnitude* and *Max-Squared*) on the parameter distributions is quite noticeable in both neural network models demonstrated in Figures 3 and 4. We observe that the large magnitude outliers seen in the baseline models (*No Regularization*) disappear entirely as a result of the introduced regularization terms, as clearly

Fig. 3. Regularization effect on the numerical range of the LeNet-5 parameters.

exemplified in Figure 4. As a result, the dynamic range of parameters reduces significantly at each layer.² The boundaries of the clustered parameter distributions are learned automatically at each layer while attempting to squeeze the numerical range and minimize the cross-entropy loss at the same time. Furthermore, zero-centered and Gaussian-like shapes of non-regularized layer parameter distributions transform into well-clustered uniform and even bimodal distributions that are concentrated around the limit values.

Our experimental results in Figures 3 and 4 with the standard L1 and L2 regularizers indicate an interesting phenomenon. While L1 and L2 regularizers can reduce the numerical range, the obtained distributions are quite distinct from those obtained through the outlier regularization methods. L1 and L2 terms penalize all parameters instead of only outliers; thus, the large magnitude discrepancies between the layer parameters still remain in the final distributions. Moreover, L1 regularization can even amplify such discrepancies by penalizing the small variables and inducing sparsity in the model while failing to shrink outlier parameters that are significantly higher than the rest of the distribution. In other words, we observe that standard regularizers do not deliver

²Please note the scale difference in Figures 3 and 4 for baseline and various regularized distributions.

Fig. 4. Regularization effect on the numerical range of the ResNet-18 parameters.

the desired outcome, or worse, would result in the exact opposite of the desired effect in the layer parameter distributions.

The obtained parameter distributions attained through the proposed outlier regularization methods offer two fundamental advantages over the baseline parameter distributions. First, we expect the numerical impact of the bit errors to be relatively muted compared to an average parameter value due to the well-clustered and better-utilized quantization range. Second, the regularized models can be quantized more accurately because of the clustering behavior and the reduction in the quantization step size. As a result, improvements in both bit error resilience and quantization accuracy are anticipated in the experiments.

5.3.3 Bit Error Resilience Analysis. We investigate the bit error resilience of the target models in the quantized format by injecting errors into model parameters and measuring the test set

Fig. 5. Weight error rate vs. LeNet-5 test set classification accuracy on MNIST.

Fig. 6. Weight error rate vs. VGG-16 test set classification accuracy on CIFAR10.

classification accuracy at various bit error rates. The accuracy curves are shown in Figures 5, 6, 7, and 8.

We experiment with various numerical range allocation methods to demonstrate the effectiveness of the proper quantization schemes in delivering high error resilience in neural networks and show further resilience improvements obtained through the described regularization techniques. We first consider a worst-case scenario where the utilized numerical representation can cover numbers that are 10 times greater than the numerical bounds necessary to cover the maximum global magnitude of the model parameters ($10 * Global$). Such a scenario could occur in general-purpose programmable hardware (i.e., large width integer formats in commodity CPU architectures) where the offered range delivered through the number of bits and the number representation fails to be utilized by the quantized model parameters. An improved quantization scheme, *Global*, designates a numerical representation range just wide enough to cover the maximum global magnitude of the model parameters, with a single scale factor to quantize the neural network parameters. The *layer-wise* scheme employs unique scale factors at each layer to improve the quantization

Fig. 7. Weight error rate vs. ResNet-18 test set classification accuracy on CIFAR10.

Fig. 8. Weight error rate vs. SqueezeNet test set classification accuracy on GTSRB.

precision further. The parameter distribution at each layer is quantized individually in this scheme to obtain tighter quantization bounds. Finally, *L1*, *L2*, *Max-Magnitude*, and *Max-Squared* schemes perform layer-wise quantization on the regularized models that are obtained with the corresponding penalty terms.

We reach the following conclusions as a result of the weight error injection experiments performed on the target models. First, the non-utilized numerical range in the number representations constitutes a significant safety vulnerability in the presence of bit errors. For instance, the VGG-16 and the ResNet-18 models experience noticeable accuracy drops in the *10 * Global* scenario at bit error rates even smaller than 10^{-6} in the parameters. The described vulnerability can be alleviated with a single scaling factor change when the variables are quantized to cover solely the global parameter range (*Global*). Consequently, the bit error resilience of the models can be improved to tolerate bit error rates up to two orders of magnitude larger. Furthermore, if the dynamic range of parameters is different across the layers as in VGG-16 and ResNet-18, the resilience characteristics can be boosted further by up to 1-2 orders of magnitude through the *Layer-wise* quantization

Table 2. Tolerated BER (Bit Error Rate) for Various Deep Neural Network Resilience Methods

		Tolerated Weight BER
LeNet-5 (MNIST)	Binary Neural Nets [59]	10^{-2}
	Median Feature Selection [50]	10^{-4}
	Anomaly Suppression [52]	10^{-3}
	This Work	5×10^{-2}
VGG-16 (CIFAR10)	Binary Neural Nets [59]	10^{-3}
	FT-ClipAct [24]	10^{-5}
	This Work	10^{-2}
ResNet-18 (CIFAR10)	Median Feature Selection [50]	10^{-4}
	Anomaly Suppression [52]	10^{-4}
	This Work	10^{-2}
SqueezeNet (GTSRB)	Median Feature Selection [50]	5×10^{-5}
	Anomaly Suppression [52]	10^{-4}
	This Work	10^{-2}

schemes and the use of unique scale factors at each layer. Finally, the regularized models offer resilience to bit error rates an order of magnitude higher when they are layer-wise quantized. The resilient deep neural network models obtained through the combined usage of layer-wise quantization and outlier regularization methods (*Max-Magnitude* and *Max-Squared*) can withstand extreme error rates, namely, bit error rates in the parameters as high as 5% for LeNet-5 and 1% for the VGG-16, ResNet-18, and SqueezeNet models while reining in the accuracy loss at around 1%. The LeNet-5 model can even tolerate parameter bit error rates as high as 10% with classification accuracy losses confined to around 3%.

We observe that standard L2 regularization has at best marginal and often no positive effect on error resilience over the non-regularized models when the parameter distributions are layer-wise quantized. Furthermore, L1 regularization degrades the bit error resilience of the models noticeably in the quantized form, in accord with the observed large numerical disparity across the layer parameters in Section 5.3.2. As a result, L1 regularized models with layer-wise quantization often exhibit weaker error resilience than the non-regularized models with global quantization. The experimental results clearly indicate that standard regularization techniques are inefficient for our purposes, and the proposed outlier regularization terms are essential for boosting parameter error resilience of deep neural networks.

We compare the parameter bit error resilience of the proposed regularized models (i.e., *Max-Magnitude* and *Max-Squared*) with the previously reported resilience values in prior literature in Table 2. Previous work includes neural network resilience methods such as FT-ClipAct [24], Median Feature Selection [50], and Anomaly Suppression [52]. We further compare our results with Binary Neural Networks [26], which are shown to be extremely resilient against bit errors in the parameters [59].

A tolerated weight bit error rate comparison at the same accuracy loss points demonstrates the noticeable superiority of our approach in enduring high bit error rates over the other active error resilience techniques [24, 50, 52]. For instance, our method delivers resilient deep learning models that can tolerate 50, 1000, 100, and 100 times larger weight bit error rates for the LeNet-5, VGG-16, ResNet-18, and SqueezeNet benchmarks, respectively, when compared to the highest limits achieved by the others for accuracy loss restrictions of no higher than 1%.

The compared active error resilience methods [24, 50, 52] mainly operate on the activations and tackle the impact of parameter (weight) errors indirectly on the produced layer activations.

Fig. 9. Activation error rate vs. LeNet-5 test set classification accuracy on MNIST.

We believe the effectiveness of our approach against parameter errors stems from the fact that we attenuate the magnitude of the parameter bit errors directly by regularizing and restricting the parameter quantization range, thus largely preventing error diffusion into the activations in the first place. We believe regularization plays a particularly important role in muting the error effects through numerical range utilization improvements and boosts in the magnitude ratio of parameters to errors, in a manner analogous to improving the signal-to-noise ratio in communication systems. Moreover, the reader will note that our approach necessitates neither additional operations nor any hardware extensions, unlike [50, 52]. As a result, our method can be deployed seamlessly in existing deep neural network accelerators with no additional overheads.

The obtained models are observed in Table 2 to be even more resilient than Binary Neural Networks [59], offering resilience for up to 5-10 times larger bit error rates in the parameters. We conjecture that while aggressive quantization schemes such as binarization deliver extensive error resilience by forestalling large error magnitudes (i.e., a parameter can only alternate between +1 and -1), nevertheless, each error transition in a compact binary representation changes the direction of information fully, thus exhibiting noticeably inferior resilience compared to our approach.

We perform similar error injection experiments on the activations by utilizing the same deep learning models. The results for activation error injection experiments are demonstrated in Figures 9, 10, 11, and 12. The *10 * Global*, *Global*, and *Layer-wise* activation quantization schemes are analogous to the descriptions provided for the weight quantization methods in the previous experiments. *L1*, *L2*, *Max-Magnitude*, and *Max-Squared* refer to the layer-wise activation quantization performed on the weight regularized models. The reader will note that we forgo the enforcement of any direct regularization on the activations due to the challenges outlined in Section 4.2.

We observe that a large numerical range in the activations can lead to safety vulnerabilities, similar to the neural network parameters. Squeezing the numerical range to the utilized global maximums and further to the unique layer-wise range at each layer can improve resilience noticeably (i.e., resilience up to 1000 times error rates for *Layer-wise* quantization when compared to a loose numerical range in *10 * Global*). We observe that the regularization effect on the parameters can often improve the activation error resilience marginally because regularized weights can indirectly eliminate the large outliers in the activations, reduce the activation quantization range, and consequently, mute the impact of large activation errors. The resilience to excessive activation error rates is often not as critical of a concern as parameter errors because the activations

Fig. 10. Activation error rate vs. VGG-16 test set classification accuracy on CIFAR10.

Fig. 11. Activation error rate vs. ResNet-18 test set classification accuracy on CIFAR10.

are generated from scratch at each forward inference pass with no long-term impact on model accuracy and thus less likely to exhibit an accumulation effect. However, the additional resilience properties obtained through proper quantization schemes are still desirable and provide additional safety against large impact errors in the activations.

5.3.4 Impact of Regularization Term on Quantization Accuracy. While we demonstrate significant improvements in bit error resilience and non-quantized model accuracy after training, the proposed outlier regularization techniques deliver additional benefits for post-training quantization methods as well. The variable distributions obtained through the proposed regularization methods are well clustered into a small dynamic range with no outliers; therefore, such distributions can be quantized to smaller bit widths while maintaining accuracy more gracefully than the non-regularized counterparts.

We conduct post-training quantization experiments on the model weights and activations to demonstrate the outlined advantage of the regularized models. We employ a layer-wise

Fig. 12. Activation error rate vs. SqueezeNet test set classification accuracy on GTSRB.

Table 3. The Impact of Regularization on the Quantized Model Accuracy

	Quantization Width (Weight, Activation)	Train Acc. (Change) (%)	Test Acc. (Change) (%)
LeNet-5 Baseline		39.38 (−60.33)	40.21 (−59.03)
LeNet-5 Regularized (L1)	(2, 4)	24.21 (−75.65)	24.21 (−75.06)
LeNet-5 Regularized (L2)		82.99 (−17.01)	82.17 (−17.24)
LeNet-5 Regularized (Max-Magnitude)		98.65 (−0.66)	98.41 (−0.66)
LeNet-5 Regularized (Max-Squared)		99.16 (−0.81)	98.59 (−0.71)
VGG-16 Baseline		98.63 (−1.08)	90.11 (−1.44)
VGG-16 Regularized (L1)	(4, 8)	97.64 (−2.01)	89.86 (−2.38)
VGG-16 Regularized (L2)		98.92 (−0.88)	90.42 (−1.51)
VGG-16 Regularized (Max-Magnitude)		99.71 (−0.05)	92.02 (−0.19)
VGG-16 Regularized (Max-Squared)		99.88 (−0.02)	91.93 (+0.04)
ResNet-18 Baseline		98.05 (−1.32)	90.39 (−1.83)
ResNet-18 Regularized (L1)	(4, 8)	98.17 (−1.69)	91.36 (−2.21)
ResNet-18 Regularized (L2)		99.48 (−0.34)	91.97 (−0.88)
ResNet-18 Regularized (Max-Magnitude)		99.77 (+0.01)	92.89 (−0.01)
ResNet-18 Regularized (Max-Squared)		99.90 (+0.04)	92.81 (−0.16)
SqueezeNet Baseline		99.18 (−0.65)	93.01 (−1.35)
SqueezeNet Regularized (L1)	(4, 8)	96.39 (−2.81)	90.43 (−3.77)
SqueezeNet Regularized (L2)		99.21 (−0.62)	92.20 (−1.74)
SqueezeNet Regularized (Max-Magnitude)		99.55 (−0.24)	93.84 (−0.63)
SqueezeNet Regularized (Max-Squared)		99.40 (−0.42)	93.57 (−0.85)

quantization scheme in these experiments by allocating a unique scaling factor for the weights and activations at each layer. The accuracy results are generated by measuring the classification accuracy of the post-training quantized models on the training and test datasets. We demonstrate the quantized model accuracy results in Table 3.

Experimental results indicate that LeNet-5 models with outlier regularization can be quantized into 2-bit weights and 4-bit activations directly, and accuracy losses confined to no more than 1% with neither fine-tuning nor compensation steps employed. On the other hand, the non-regularized LeNet-5 model results in around 60% accuracy loss when quantized directly after training. LeNet-5 models with standard regularization techniques experience a noticeable accuracy loss, which is even more severe than the non-regularized LeNet-5 model for the L1 regularization case. We

observe similar trends in other neural network benchmarks. For instance, VGG-16 and ResNet-18 models with outlier regularization do not experience a noticeable loss, and the accuracy is often slightly improved when quantized to 4-bit weights and 8-bit activations. However, the non-regularized VGG-16 and ResNet-18 models incur 1.44% and 1.83% accuracy loss due to quantization, respectively. The accuracy loss values for the L1 and L2 regularized models are observed to be noticeably larger than the models with outlier regularization and even inferior to the non-regularized models in the case of L1 regularization.

While improving the quantization process is not the primary focus of our work, and various advanced techniques do exist to obtain accurate and compact quantized models [15], the outlined observation aims solely to showcase the positive impact of regularization on the quantized model accuracy under post-training scenarios with no fine-tuning.

5.3.5 Observed Differences between Max-magnitude and Max-squared Regularization Terms. The proposed max-magnitude and max-squared regularization terms penalize the outlier parameters at each layer. We observe that both regularization terms can squeeze parameter distributions effectively and deliver significant error resilience improvements. Max-magnitude regularization usually incurs a slightly larger penalty value than the max-squared term at the same regularization coefficient (α) since neural network parameters are often observed to be small values between $[-1, 1]$, where the variable magnitude is numerically larger than its squared version. However, the difference between the obtained parameter distributions through these regularization terms is observed to be minimal in practice.

6 DISCUSSION

The resilience of deep neural networks is commonly observed in previous studies. Neural networks can maintain accuracy gracefully, even under widespread noise and error conditions, as long as the captured information in the model variables is not overwhelmed by noise and error incidents. For instance, neural networks are known to maintain accuracy gracefully during quantization, which is a process of widespread error introduction. From this perspective, error tolerance in neural networks bears significant similarities to the problem of improving the signal-to-noise ratio in digital signal processing and telecommunication systems.

While the outlined observations draw a hopeful conclusion regarding the resilience of a neural network, bit error effects whose numerical impact stretches well beyond the described noise levels remain its Achilles' heel. The resilience properties of the neural networks can begin to shatter rapidly under large magnitude error incidents. The proper design choices that preclude such scenarios not only improve error resilience dramatically but may furthermore boost the accuracy of the system noticeably under error-free conditions. For instance, the proper utilization and the constriction of the numerical range in the quantization schemes improve both the quantization quality and tighten the potential error margins effectively.

We further demonstrate how the distribution of the model parameters can be formed in the desired manner through the proper training goals to obtain deep learning models that are highly error-resilient in the quantized form. Surprisingly, such restricted models can be even more accurate than their unrestricted counterparts in both full-precision and quantized formats.

In summary, we illustrate that deep learning hardware, when properly designed, can tolerate excessive bit errors in the datapath at levels even up to 10%, offering resilience shields unmatched by much costlier traditional fault-tolerant systems. Moreover, such resilient systems are constructed at no additional cost enjoying several auxiliary benefits such as improved model accuracy. Overall, delivering datapath safety in deep learning algorithms can indeed be a rather satisfying free lunch, maybe even with a few complimentary low-calorie free desserts thrown in.

The ability to endure such severe error conditions for free is not only an invaluable characteristic for functional safety but also opens up a slew of opportunities for the next generation of embedded designs in deep neural network processing. The relaxation in the rigidity of the individual correctness requirements in the hardware can be exploited to reduce the resource footprint through design optimizations or utilizing alternative manufacturing technologies that suffer from inherent variability and may be prone to high defect rates.

The aggressive hardware optimizations can deliver remarkable energy efficiency improvements in memory, communication, and computation circuitry in CMOS (complementary metal-oxide-semiconductor) devices when combined with the superior error resilience characteristics of the proposed deep learning algorithms. First, we estimate an up to 60% energy reduction in the SRAM on-chip buffers through supply voltage scaling when the consequent bit cell failure rates characterized in [57] are inherently tolerated by deep neural networks. Second, data communication constitutes a significant portion of the energy consumption in modern deep learning accelerators [46]. Innovative energy efficiency techniques such as reduced voltage swing [44] can cut down the interconnect energy consumption by 40% at the cost of increased bit error rates in data communication. While specific applications such as multimedia benchmarks tolerate consequent bit error rates in the range of 10^{-5} – 10^{-3} inherently [44], our presented analysis indicates even greater potential for the popular deep learning algorithms when the outlined design methods are followed to boost error resilience. Third, the supply voltage of the multiply-accumulate units can be scaled to obtain a quadratic reduction in dynamic power of up to 50% [74] if the accuracy impact of the elevated number of timing errors can be minimized through the proposed resilience techniques. We believe comprehensive hardware studies, as well as the implementation of automated design tools, are necessary for the precise identification of the voltage vs. frequency operating points for embedded deep learning accelerators so that utmost energy efficiency goals can be attained with a guaranteed system accuracy. Our future work will focus on various hardware case studies to achieve the outlined energy efficiency goals.

Moreover, the error resilience of deep neural networks can facilitate novel computation paradigms [73], and novel communication fabrics through photonics, wireless, or 3D interconnects [46] when the inherent inaccuracy and manufacturing issues for such emerging technologies can be endured through the algorithmic resilience of deep neural networks.

Finally, we anticipate that the lack of sensitivity to aging effects due to wide error tolerance bounds can extend the operational lifetime of the system noticeably without sacrificing accuracy, reliability, or quality of service in embedded application domains.

7 CONCLUSIONS

The usage of deep neural networks has permeated across various embedded application domains that are governed by strict safety constraints, from autonomous driving to industrial control systems. As the computational burden of deep neural networks challenges the delivery of functional safety in embedded deep learning applications through traditional fault tolerance techniques, we investigate an alternative path to embed unparalleled reliability characteristics into deep neural networks by utilizing their inherent resilience. Our results indicate that properly designed quantization schemes can reduce error margins and improve system resilience noticeably against bit errors in the neural network variables. We propose a novel regularization term to manipulate the parameter distributions at each layer and tighten error margins further. Neural network models obtained through the described regularization process are shown to be highly accurate after training, lend themselves to accurate quantization at low bit widths with minimal accuracy loss, and exhibit extreme levels of resilience to hardware bit errors in the quantized format. Our work presents strong evidence that highly accurate and error-resilient deep learning systems can

be built merely through algorithmic modifications and proper design techniques while imposing essentially zero hardware, energy, and performance overheads.

REFERENCES

- [1] Massimo Baleani, Alberto Ferrari, Leonardo Mangeruca, Alberto Sangiovanni-Vincentelli, Maurizio Peri, and Saverio Pezzini. 2003. Fault-tolerant platforms for automotive safety-critical applications. In *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. ACM, 170–177.
- [2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).
- [3] Nandhini Chandramoorthy, Karthik Swaminathan, Martin Cochet, Arun Paidimarri, Schuyler Eldridge, Rajiv V. Joshi, Matthew M. Ziegler, Alper Buyuktosunoglu, and Pradip Bose. 2019. Resilient low voltage accelerators for high energy efficiency. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 147–158.
- [4] Lerong Chen, Jiawen Li, Yiran Chen, Qiuping Deng, Jiyuan Shen, Xiaoyao Liang, and Li Jiang. 2017. Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 19–24.
- [5] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2019. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 2 (2019), 292–308.
- [6] Zitao Chen, Guanpeng Li, and Karthik Pattabiraman. 2021. A low-cost fault corrector for deep neural networks through range restriction. *arXiv preprint arXiv:2003.13874* (2021).
- [7] Zitao Chen, Guanpeng Li, Karthik Pattabiraman, and Nathan DeBardeleben. 2019. BinFI: An efficient fault injector for safety-critical machine learning systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–23.
- [8] Jungwook Choi, Swagath Venkataramani, Vijayalakshmi Srinivasan, Kailash Gopalakrishnan, Zhuo Wang, and Pierce Chuang. 2019. Accurate and efficient 2-bit quantized neural networks. In *Proceedings of the 2nd SysML Conference*.
- [9] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. PACT: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085* (2018).
- [10] Wonseok Choi, Dongyeob Shin, Jongsun Park, and Swaroop Ghosh. 2019. Sensitivity based error resilient techniques for energy efficient deep neural network accelerators. In *Proceedings of the 56th Design Automation Conference*. 1–6.
- [11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2014. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024* (2014).
- [12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [13] Fernando Fernandes dos Santos, Pedro Foletto Pimenta, Caio Lunardi, Lucas Draghetti, Luigi Carro, David Kaeli, and Paolo Rech. 2018. Analyzing and increasing the reliability of convolutional neural networks on GPUs. *IEEE Transactions on Reliability* 68, 2 (2018), 663–677.
- [14] Ben Feinberg, Shibo Wang, and Engin Ipek. 2018. Making memristive neural network accelerators reliable. In *International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 52–65.
- [15] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630* (2021).
- [16] Ghouthi Boukli Hacene, François Leduc-Primeau, Amal Ben Soussia, Vincent Gripon, and François Gagnon. 2019. Training modern deep neural networks for memory-fault robustness. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
- [17] Richard W. Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal* 29, 2 (1950), 147–160.
- [18] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems* 28 (2015), 1135–1143.
- [19] Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, and Stephen W. Keckler. 2021. Making convolutions resilient via algorithm-based error detection techniques. *IEEE Transactions on Dependable and Secure Computing* (2021).
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

- [21] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4340–4349.
- [22] Zhezhi He, Jie Lin, Rickard Ewetz, Jiann-Shiun Yuan, and Deliang Fan. 2019. Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping. In *Proceedings of the 56th Annual Design Automation Conference*. 1–6.
- [23] Tifenn Hirtzlin, Marc Bocquet, J.-O. Klein, Etienne Nowak, Elisa Vianello, J.-M. Portal, and Damien Querlioz. 2019. Outstanding bit error tolerance of resistive RAM-based binarized neural networks. In *International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 288–292.
- [24] Le-Ha Hoang, Muhammad Abdullah Hanif, and Muhammad Shafique. 2020. FT-ClipAct: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1241–1246.
- [25] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).
- [26] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. *Advances in Neural Information Processing Systems* 29 (2016).
- [27] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).
- [28] Xun Jiao, Mulong Luo, Jeng-Hau Lin, and Rajesh K. Gupta. 2017. An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations. In *International Conference on Computer-Aided Design (ICCAD)*. IEEE/ACM, 945–950.
- [29] Michael Klachko, Mohammad Reza Mahmoodi, and Dmitri Strukov. 2019. Improving noise tolerance of mixed-signal neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
- [30] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.222.9220>. (2009).
- [31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
- [32] Minjae Lee, Kyuyeon Hwang, and Wonyong Sung. 2014. Fault tolerance analysis of digital feed-forward deep neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5031–5035.
- [33] Guanpeng Li, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, and Stephen W. Keckler. 2017. Understanding error propagation in deep learning neural network (DNN) accelerators and applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.
- [34] Guanpeng Li, Karthik Pattabiraman, and Nathan DeBardleben. 2018. TensorFI: A configurable fault injector for TensorFlow applications. In *International Symposium on Software Reliability Engineering Workshops*. IEEE, 313–320.
- [35] Wenshuo Li, Xuefei Ning, Guangjun Ge, Xiaoming Chen, Yu Wang, and Huazhong Yang. 2020. FTT-NAS: Discovering fault-tolerant neural architecture. In *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 211–216.
- [36] Chenchen Liu, Miao Hu, John Paul Strachan, and Hai Li. 2017. Rescuing memristor-based neuromorphic design with high defects. In *54th Design Automation Conference (DAC)*. ACM/EDAC/IEEE, 1–6.
- [37] Mengyun Liu, Lixue Xia, Yu Wang, and Krishnendu Chakrabarty. 2018. Fault tolerance for RRAM-based matrix operations. In *International Test Conference (ITC)*. IEEE, 1–10.
- [38] Mengyun Liu, Lixue Xia, Yu Wang, and Krishnendu Chakrabarty. 2019. Fault tolerance in neuromorphic computing systems. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 216–223.
- [39] Tao Liu, Wujie Wen, Lei Jiang, Yanzhi Wang, Chengmo Yang, and Gang Quan. 2019. A fault-tolerant neural network architecture. In *56th Design Automation Conference (DAC)*. ACM/IEEE, 1–6.
- [40] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single shot multibox detector. In *European Conference on Computer Vision*. Springer, 21–37.
- [41] Yun Long, Xueyuan She, and Saibal Mukhopadhyay. 2019. Design of reliable DNN accelerator with un-reliable ReRAM. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1769–1774.
- [42] Atieh Lotfi, Saurabh Hukerikar, Keshav Balasubramanian, Paul Racunas, Nirmal Saxena, Richard Bramley, and Yanxiang Huang. 2019. Resiliency of automotive object detection networks on GPU architectures. In *International Test Conference (ITC)*. IEEE, 1–9.
- [43] Abdulrahman Mahmoud, Neeraj Aggarwal, Alex Nobbe, Jose Rodrigo Sanchez Vicarte, Sarita V. Adve, Christopher W. Fletcher, Iuri Frosio, and Siva Kumar Sastry Hari. 2020. PyTorchFI: A runtime perturbation tool for DNNs. In *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 25–31.

- [44] Andrea Mineo, Maurizio Palesi, Giuseppe Ascia, Partha Pratim Pande, and Vincenzo Catania. 2016. On-chip communication energy reduction through reliability aware adaptive voltage swing scaling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 11 (2016), 1769–1782.
- [45] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. 2017. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition* 65 (2017), 211–222.
- [46] Seyed Morteza Nabavinejad, Mohammad Baharloo, Kun-Chih Chen, Maurizio Palesi, Tim Kogel, and Masoumeh Ebrahimi. 2020. An overview of efficient interconnection networks for deep neural network accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10, 3 (2020), 268–282.
- [47] Mohamed A. Neggaz, Ihlen Alouani, Pablo R. Lorenzo, and Smail Niar. 2018. A reliability study on CNNs for critical embedded systems. In *36th International Conference on Computer Design (ICCD)*. IEEE, 476–479.
- [48] Mohamed A. Neggaz, Ihlen Alouani, Smail Niar, and Fadi Kurdahi. 2019. Are CNNs reliable enough for critical applications? an exploratory study. *IEEE Design & Test* 37, 2 (2019), 76–83.
- [49] Elbruz Ozen and Alex Orailoglu. 2019. Sanity-Check: Boosting the reliability of safety-critical deep neural network applications. In *28th Asian Test Symposium (ATS)*. IEEE, 7–12.
- [50] Elbruz Ozen and Alex Orailoglu. 2020. Boosting bit-error resilience of DNN accelerators through median feature selection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3250–3262.
- [51] Elbruz Ozen and Alex Orailoglu. 2020. Concurrent monitoring of operational health in neural networks through balanced output partitions. In *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 169–174.
- [52] Elbruz Ozen and Alex Orailoglu. 2020. Just Say Zero: Containing critical bit-error propagation in deep neural networks with anomalous feature suppression. In *International Conference On Computer Aided Design (ICCAD)*. IEEE/ACM, 1–9.
- [53] Elbruz Ozen and Alex Orailoglu. 2020. Low-cost error detection in deep neural network accelerators with linear algorithmic checksums. *Journal of Electronic Testing* 36, 6 (2020), 703–718.
- [54] Elbruz Ozen and Alex Orailoglu. 2021. Squeezing correlated neurons for resource-efficient deep neural networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part II*. Springer International Publishing, 52–68.
- [55] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019), 8026–8037.
- [56] Brandon Reagan, Udit Gupta, Lillian Pentecost, Paul Whatmough, Sae Kyu Lee, Niamh Mulholland, David Brooks, and Gu-Yeon Wei. 2018. Ares: A framework for quantifying the resilience of deep neural networks. In *55th Design Automation Conference (DAC)*. ACM/ESDA/IEEE, 1–6.
- [57] Brandon Reagan, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. 2016. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *43rd Annual International Symposium on Computer Architecture (ISCA)*. ACM/IEEE, 267–278.
- [58] Angad S. Rekhi, Brian Zimmer, Nikola Nedovic, Ningxi Liu, Rangharajan Venkatesan, Miaorong Wang, Brucek Khailany, William J. Dally, and C. Thomas Gray. 2019. Analog/mixed-signal hardware error modeling for deep learning inference. In *56th Annual Design Automation Conference*. 1–6.
- [59] Majid Sabbagh, Cheng Gongye, Yunsu Fei, and Yanzhi Wang. 2019. Evaluating fault resiliency of compressed deep neural networks. In *International Conference on Embedded Software and Systems (ICCESS)*. IEEE, 1–7.
- [60] Christoph Schorn, Thomas Elskens, Sebastian Vogel, Armin Runge, Andre Guntoro, and Gerd Ascheid. 2020. Automated design of error-resilient and hardware-efficient deep neural networks. *Neural Computing and Applications* 32, 24 (2020), 18327–18345.
- [61] Christoph Schorn, Andre Guntoro, and Gerd Ascheid. 2018. Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 979–984.
- [62] Christoph Schorn, Andre Guntoro, and Gerd Ascheid. 2018. Efficient on-line error detection and mitigation for deep neural network accelerators. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 205–219.
- [63] Muhammad Shafique, Mahum Naseer, Theodoris Theodorides, Christos Kyrkou, Onur Mutlu, Lois Orosa, and Jungwook Choi. 2020. Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead. *IEEE Design & Test* 37, 2 (2020), 30–57.
- [64] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [65] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks* 32 (2012), 323–332.

- [66] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. 2020. Efficient processing of deep neural networks. *Synthesis Lectures on Computer Architecture* 15, 2 (2020), 1–341.
- [67] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*. PMLR, 6105–6114.
- [68] Darrell Williamson. 1991. Dynamically scaled fixed point arithmetic. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing Conference Proceedings*. 315–318.
- [69] Lixue Xia, Mengyun Liu, Xuefei Ning, Krishnendu Chakrabarty, and Yu Wang. 2018. Fault-tolerant training enabled by on-line fault detection for RRAM-based neural computing systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 9 (2018), 1611–1624.
- [70] Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. 2019. AutoPrune: Automatic network pruning by regularizing auxiliary parameters. In *Advances in Neural Information Processing Systems*. 13681–13691.
- [71] Zheng Xu and Jacob Abraham. 2019. Safety design of a convolutional neural network accelerator with error localization and correction. In *International Test Conference (ITC)*. IEEE, 1–10.
- [72] Zheyu Yan, Yiyu Shi, Wang Liao, Masanori Hashimoto, Xichuan Zhou, and Cheng Zhuo. 2020. When single event upset meets deep neural networks: Observations, explorations, and remedies. In *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 163–168.
- [73] Shimeng Yu. 2018. Neuro-inspired computing with emerging nonvolatile memory. *Proc. IEEE* 106, 2 (2018), 260–285.
- [74] Jeff Zhang, Kartheek Rangineni, Zahra Ghodsi, and Siddharth Garg. 2018. ThunderVolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.
- [75] Jeff Jun Zhang, Tianyu Gu, Kanad Basu, and Siddharth Garg. 2018. Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator. In *36th VLSI Test Symposium (VTS)*. IEEE, 1–6.
- [76] Zixing Zhang, Jürgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Björn Schuller. 2018. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, 5 (2018), 1–28.

Received April 2021; revised June 2021; accepted July 2021