On Hardening DNNs against Noisy Computations

Xiao Wang, Hendrik Borras, Bernhard Klein, Holger Fröning
HAWAII Lab, Heidelberg University, Germany
{xiao.wang, hendrik.borras, bernhard.klein, holger.froening}@ziti.uni-heidelberg.de

Abstract—**The success of deep learning has sparked significant interest in designing computer hardware optimized for the high computational demands of neural network inference. As further miniaturization of digital CMOS processors becomes increasingly challenging, alternative computing paradigms, such as analog computing, are gaining consideration. Particularly for compute-intensive tasks such as matrix multiplication, analog computing presents a promising alternative due to its potential for significantly higher energy efficiency compared to conventional digital technology. However, analog computations are inherently noisy, which makes it challenging to maintain high accuracy on deep neural networks. This work investigates the effectiveness of training neural networks with quantization to increase the robustness against noise. Experimental results across various network architectures show that quantization-aware training with constant scaling factors enhances robustness. We compare these methods with noisy training, which incorporates a noise injection during training that mimics the noise encountered during inference. While both two methods increase tolerance against noise, noisy training emerges as the superior approach for achieving robust neural network performance, especially in complex neural architectures.**

Index Terms—**Analog noise, Robustness, Quantization, Noisy training.**

I. Introduction

In the past decade, deep neural networks (DNNs) have been widely employed to address a variety of real-world problems, from fields such as image, signal and speech processing, natural language processing, and autonomous systems. Their ability to achieve remarkable performance on challenging tasks has made them a popular choice in various applications. DNNs are parametric models characterized by a vast number of weights, necessitating powerful specialized processors, such as GPUs, for training and evaluation. However, the rapid growth in the size of DNNs introduces new challenges related to memory-footprint, computation, and power consumption, particularly for deployment in resource-constrained embedded or mobile devices [28, 13]. This growing complexity contradicts the practical application of DNNs to real-world problems, as inference with these models often leads to unacceptable latency or excessive energy consumption in battery-powered devices.

In this regard, multiple recent studies have turned to alternative computing technologies, such as analog computing, for instance based on analog electrical computations [23, 19], analog optical computations [30] and similarly emerging memory technologies such as resistive RAM [6, 8]. In these applications, the computations inside DNNs are executed in the analog domain with weights being represented by analog quantities, e.g. electrical voltage [2], photons [30, 4], or conductance [14]. Analog hardware has the potential to offer substantial improvements in energy efficiency, potentially exceeding two orders of magnitude compared to digital hardware [14, 34]. However, analog quantities are subject to the noise inherently present in their physical components. Depending on the technology, the weights may be written with some error [29]. Each weight readout can be overlaid with some noise, or the arithmetic operations themselves may also be fraught with noise. In other words, analog computations are inherently noisy, and without countermeasures, this noise can negatively affect prediction accuracy.

To use such noisy hardware, methods to improve robustness against unreliable computations are essential. From an algorithmic perspective, previous empirical work has shown that noise injection during training can lead to improvements in the noise resilience of analog computing devices [34, 33, 14, 27]. For instance, Gaussian noise is widely used in the training process of DNNs to improve the robustness [14, 34, 27] during inference. Noisy Machines [34] explores the use of knowledge distillation from a digitally trained teacher network to a student network with noise injection, while BayesFT [33] proposes different types injecting noise, including Bernoulli noise, Gaussian noise, and Laplace noise, to enhance the robustness of analog neural networks. Similarly, training as a counter measure to improve robustness against noise while slowly exposing a neural network architecture to an increasing amount of noise has proven to be effective [16]. Moreover, Isik et al. [13] have explored neural network compression techniques such as pruning and knowledge distillation on noisy storage devices.

Quantization is widely recognized as an effective method for compressing models on deterministic hardware; moreover, as analog hardware also faces practical limitations regarding precision, and quantization is implicitly required. Quantization is also a good way to simulate the noise in analog-to-digital (ADC) and digital-to-analog (DAC) converters for neural network accelerators. By mapping weights and/or activations from floating-point representations into ”bins” (lower precision formats, such as 8-bit or 4-bit integers), quantized neural networks may stay sturdy when models are perturbed by noise. Similarly, is has been shown previously that that quantization can either improve or degrade adversarial robustness depending on the attack strength [11]. In general, techniques such as quantization-aware training (QAT) have demonstrated significant success in mitigating quantization errors and enhancing generalization behavior in models [26]. Given this promising body of previous work, it would be valuable to evaluate to which extend QAT improves the robustness against noise inherently present in analog computations, particularly as best to our knowledge, this question has not been explored in the literature before.

In the following, we thus explore the robustness of quantized neural networks in the presence of noisy computations. As a backdrop, we also investigate models trained with noise injection to provide a quantitative comparison.

To quantify and compare the robustness as a key performance metric, we employ the *midpoint noise level* μ as proposed in [3]. Our contributions can be summarized as follows:

1) We investigate the *robustness by quantization* against analog noise and compare robustness across different precision formats as well as constant and dynamic scaling.
2) Similarly, we investigate noisy training methods, which simulates the conditions of a real noisy analog device, compelling the network to adapt to noise during the training process.
3) Finally, we present an overall evaluation, comparing quantization methods and noisy training techniques across different architectures.

Empirically, we evaluate the effectiveness of our methods on image classification tasks using models trained on CIFAR-10. Our evaluation shows that quantization and noisy training can all enhance the robustness to different extents.

II. Related Work

Noisy Machines [34] models generic non-volatile memory (NVM) cell noise as an additive zero-mean independent and identically distributed (*i.i.d.*) Gaussian noise term on the weights $\boldsymbol{w}_i$ of the model in each particular layer l: $\Delta\boldsymbol{w}_i \sim \mathcal{N}(\Delta\boldsymbol{w}_i; 0, \sigma_{N,l}^2\mathbf{I})$, where $\mathbf{I}$ is the identity matrix and $\sigma_{N,l}$ is the noise standard deviation of layer l. Moreover, they examine the training with injected Gaussian noise to increase robustness against such analog computations and proposed knowledge distillation as a further extension to increase robustness. BayesFT [33] adopts a memristor perturbation model, which considered multiple factors resulting in the memristance drifting. Specifically, the drifting term is applied to each neural network parameter $\boldsymbol{w}_i' \leftarrow \boldsymbol{w}_i e^{\lambda}, \lambda \sim \mathcal{N}(0, \sigma^2)$, where $\boldsymbol{w}_i'$ is the drifted neural network parameter. Compared with Noisy Machines and BayesFT, injecting noise on weights, *Walking Noise* [3] focus on injection at the output activation, addressing combined noise from weight readout and the subsequent computations such as a dot product.

Previous work on quantization have primarily focused on its effects on DNNs' robustness to adversarial examples [11, 21, 7, 5, 10] and its impact on neural networks with different architectures [32] and quantization processes [5].

Giacobbe, Henzinger, and Lechner [10] indicate that robustness against adversarial attacks is non-monotonic in the number of bits. Duncan et al. [7] find that quantization can improve a network's resilience to adversarial attacks overall whilst causing negligible loss of precision. Gorsline, Smith, and Merkel [11] suggest that there is a critical adversarial attack strength at which quantization has little-to-no effect on accuracy. For attack strengths less than this critical strength, increasing weight precision improves accuracy by enabling more complex decision boundaries. At attack strengths greater than the critical strength, increasing precision causes a drop in accuracy stemming from decision boundaries being closer to data points. Defensive Quantization [21] highlights that vanilla quantization suffers more from adversarial attacks due to the error amplification effect, where the quantization operation further enlarges the distance caused by amplified noise. They propose to control the Lipschitz constant of the network during quantization, such that the magnitude of the adversarial noise remains non-expansive during inference.

Additionally, Sung, Shin, and Hwang [32] analyse the effects of quantization on feedforward deep neural networks and convolutional neural networks as their complexity varies. This study also show that highly complex DNNs have the capability of absorbing the effects of severe weight quantization through retraining, but connection-limited networks are less resilient. To provide intrinsic robustness to the model against a broad range of quantization processes, Robust Quantization [5] introduces a kurtosis regularization term, which is added to the model loss function.

III. Methodology

While analog accelerators promise to be orders of magnitude more energy efficient than their digital counterparts, they are inevitably fraught with noise and non-linearities in their computations. In this section, we describe the metric utilized to evaluate the robustness of DNNs under noisy computations, as well as the methods employed as counter measures to losses in robustness due to noise.

A. Robustness: midpoint noise level

To quantify robustness of a DNN against injected noise, we measure the *midpoint noise level* μ; a metric developed in a previous work by Borras, Klein, and Fröning [3]. It is defined as the injected noise at which the network achieves half of its maximum accuracy, precisely $\delta a = \frac{a_{max} - a_{min}}{2}$, with a_{max} and a_{min} being the maximally and minimally achieved accuracy, respectively. Under normal circumstances, a_{min} equates to the random prediction accuracy of a given dataset. Equation (1) describes a scaled and shifted logistic function which is fitted to the observed data,

$$F(\sigma; \mu, s, \delta a, a_{min}) = \frac{2}{1 + e^{(\sigma-\mu)/s}} \cdot \delta a + a_{min} \quad (1)$$

with μ as specified before, s the curve's slope, and δa and a_{min} being the curve's scale and shift factor. Fitting a function to the data additionally gives us the ability to link data points with uncertainty information, which are a natural result of fluctuations during training. Thus the error estimate returned by the fit can be used to assess the reliability of the obtained result:

$$\mu = \arg\min_{\mu, s, \delta a, a_{min}} \left\| \frac{F(\sigma; \mu, s, \delta a, a_{min}) - y(\sigma)}{\Delta y(\sigma)} \right\|^2, \forall\sigma \quad (2)$$

where $y(\sigma)$ and $\Delta y(\sigma)$ refer to the observed accuracy and uncertainty, respectively.

This metric follows the intuition that a larger μ corresponds to higher robustness against noise. And is roughly equivalent to simply finding the data point closest to the threshold, where the investigated network achieves half of the maximum possible accuracy. However, the metric is more stable against fluctuations by using the entire data and incorporating statistical errors and additionally allowing to quantify how certain the resulting observation is.

By injecting noise globally with the same intensity at all layers of the network, the *midpoint noise level* μ can reflect the sensitivity of a network to noise. Furthermore, noise can also be injected exclusively at a single layer to probe how the internals of a network react to the noise.

B. Quantization

When applying quanitzation to DNNs, the goal is to reduce the precision of both the parameters $\boldsymbol{w}_i$, as well as the intermediate activation maps $\boldsymbol{a}_i$ to a lower precision, with minimal impact on the generalization power, in other terms accuracy of the model. To do this, we first provide a quantization scheme that maps a floating point value to a quantized one, such as an integer, and then introduce how we adapt this scheme for training with backpropagation.

In this work we focus on the common quantization method, *uniform quantization*, where the distance s between quantization intervals is identical across all quantization levels. This distance s is also called the *scaling factor*. Then, the quantization function can be defined as follows:

$$Q(x) = \lfloor \frac{x}{s} \rceil + z, \tag{3}$$

where Q is the quantization operator, x is a real-valued input, $\lfloor \cdot \rceil$ is the rounding operation (e.g. round to nearest or round to floor) and z is the *zero point*. The scaling factor and the zero point are used to map a floating point value to the integer grid, whose size depends on the bit-width b. The scaling factor can be defined manually as a hyperparameter, but it can also be defined according to the desired range of real values:

$$s = \frac{\beta - \alpha}{2^b - 1}, \tag{4}$$

where $[\alpha, \beta]$ denotes the bounded range that real values are clipped with.

Symmetric and asymmetric quantization. *Symmetric quantization* is a simplified version of the general asymmetric case. Symmetric quantization restricts the zero point to 0, i.e. $-\alpha = \beta$. This reduces the computational overhead of dealing with the zero point offset. However, *asymmetric quantization* often results in a tighter clipping range and thus better representation of the actual value space, since the clipping range can be chosen to use the min/max of the input, i.e. $\alpha = x_{min}, \beta = x_{max}$, which results in the zero point non-zero. This is especially important when the target weights or activations are imbalanced, e.g., the activations after a ReLU, which are always of positive value.

After the representation range $[\alpha, \beta]$ is determined, any values of x that lie outside of this range will be clamped to its limit:

$$q_l = \text{clamp}\Big(\lfloor \frac{x}{s} \rceil + z, q_{min}, q_{max}\Big), \tag{5}$$

where q_l is the quantization level , $[q_{min}, q_{max}]$ denotes the range of the integer grid, i.e. $\{0, \ldots, 2^b - 1\}$ for *unsigned* integers and $\{-2^{b-1}, \ldots, 2^{b-1} - 1\}$ for *signed* integers. Here, $[q_{min}, q_{max}]$ are equivalent to the quantized values of $[\alpha, \beta]$.

Notably, the clipping procedure may incur a *clipping error* [24]. To reduce the clipping error one can expand the quantization range by increasing the scaling factor. However, increasing the scaling factor leads to increased *rounding errors* [24] as the rounding error is dependent on the range $\left[-\frac{1}{2}s, \frac{1}{2}s\right]$. As such choosing the scaling factor s becomes a trade-off.

Static and dynamic quantization. Except for the methods for determining the clipping range $[\alpha, \beta]$, another important differentiator of quantization methods is *when* the clipping range is determined. This range can be computed statically for weights, as the parameters are fixed during inference. However, the activation maps differ for each input sample. So, there are two approaches to quantizing activations: *dynamic quantization*, and *static quantization*.

In dynamic quantization, this range is dynamically calculated for each activation map during runtime. This approach requires real-time computation of the signal statistics (min, max, percentile, etc.) which can have a significant overhead. However, dynamic quantization often results in higher accuracy as the signal range is exactly calculated for each activation.

Another quantization approach is static quantization, in which the clipping range is pre-calculated and static during inference. This approach does not add any computational overhead, but it typically results in lower accuracy compared to dynamic quantization. In this work, we mainly focus on static quantization. Specifically, we investigate *constant scaling* and *dynamic scaling*. In constant scaling, a single scaling factor is applied to all activations, whereas in dynamic scaling, each activation (or groups thereof) has its own scaling factor. We use the term *dynamic scaling* to differentiate it from *dynamic quantization*.

Quantization-aware training. In this paper, we focus on quantization compatible with backpropagation, generally called quantization-aware training (QAT), which evaluates the impact of parameter quantization during training and adjusts parameters using training data to mitigate accuracy degradation, thus achieving higher accuracies than direct post-training quantization. A common method for implementing QAT is to insert fake quantization [9], where values are stored in full precision but discretized during computation. Additionally, the Straight Through Estimator (STE) [1] is used to address the non-differentiable quantization operator in backpropagation by approximating the rounding operation with an identity function.

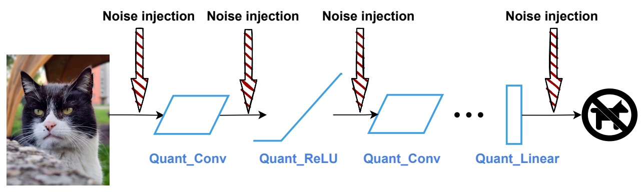


Fig. 1. Global noise injection in a quantized neural network.

A quantized convolutional neural network with global noise injection is depicted in Fig. 1.

IV. Experiments

All experiments are conducted using the CIFAR-10 dataset [18]. We focus on three representative convolutional neural network architectures, LeNet-5 [20], VGG [31] and ResNet [12]. The simplicity of LeNet-5 allows us to isolate the effects of quantization without the added complexity of more modern, deeper networks. VGG and ResNet networks, in contrast, have significantly deeper architectures. This depth allows for the study of quantization effects across many layers, providing insights into how quantization impacts deeper networks.

To quantize a neural network, we employ QAT, using the open-source framework *Brevitas* [25] and follow the findings from Krishnamoorthi [17]. Specifically, we apply per-channel quantization on convolutional layers and per-tensor quantization on fully connected layers and activations. Uniform quantization is used for both weight tensors and activations. For the activations, we explore both constant and dynamic scaling. The bit widths for weight tensors and activations are uniform.

As additive noise is often the primary type of noise in accelerators, we employ the quantified robustness metric *midpoint noise level* μ [3] to asses the robustness of an architecture against noisy computation. To inject noise at the activations of a given neural network, a custom noise module is added after each layer. Similar to STE [1], the noise is only injected in the forward path, not the backward path. For all experiments, we inject Gaussian additive noise globally with the same intensity at all layers of the network.

A. Robustness of QAT without noisy training

1) Results with LeNet-5: In order to compare the effectiveness of quantization as a robustness method, all experiments are trained on the CIFAR-10 dataset without noise injection. While the baseline is trained with floating-point precision. Both the baseline and quantized models are trained using Adam [15] with cosine learning rate decay [22], and an initial learning rate of 0.001 (with batch size 128) for 500 epochs. In order to isolate the effects of quantization, models are trained without batch normalization layers and regularization methods[1]. To observe the uncertainty of data points, each inference is repeated 10 times on different networks with different random weight initialization. The accuracies obtained for different bit widths and different scaling methods are shown in Fig. 2. As expected the model accuracy degrades with increasing noise for the baseline LeNet-5 and its quantized variants.

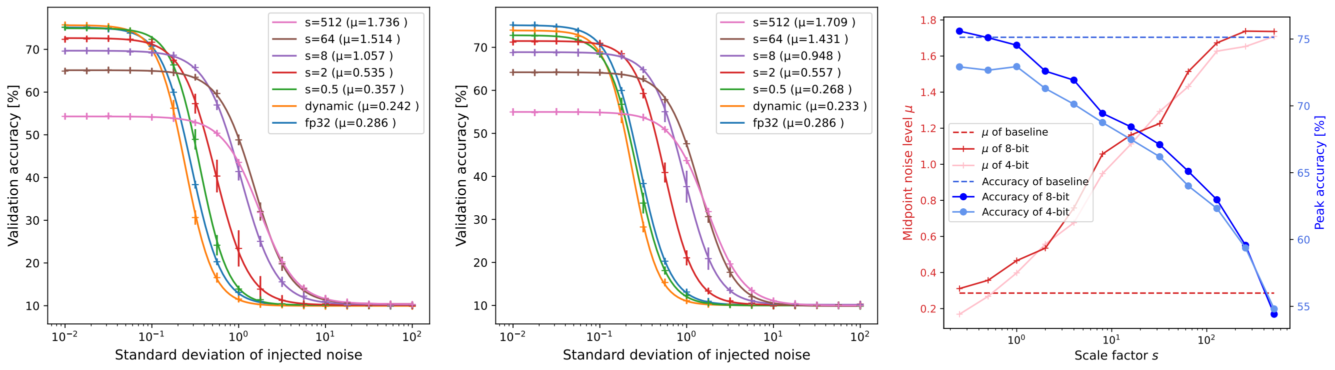
The clipping range is computed statically for weights, as the parameters are fixed during inference. For the activations, we compare dynamic and constant scaling. As can be seen, the dynamic scaling (orange curves in Fig. 2 (a) and (b)) can achieve almost the same peak accuracy as the non-quantized baseline model for both 4-bit and 8-bit. However, the curves lie left to the baseline curve, which means that the robustness of the dynamic scaling is slightly worse than the baseline.

For constant scaling factors, we identify a trade-off between the peak accuracy and noise robustness of a model for both 4-bit and 8-bit. Larger scaling factors result in larger midpoint levels and lower peak accuracies. In the low noise regime, dynamic scaling factors and small constant scaling factors can preserve the peak accuracies, but when the constant scaling factors are larger than 1, the accuracies drop significantly. In the high noise regime, larger constant scaling factors can achieve significantly better accuracies than dynamic scaling factors up to a scaling factor of 512. As introduced in Section III-B, the rounding error then lies in the range of $[-\frac{1}{2}s, \frac{1}{2}s]$, thus increasing the scaling factor leads to increased rounding error and reduced top accuracy. However, with constant scaling, the rounding error can be potentially mitigated by the weights during training.

To better evaluate the trade-off between different quantization granularities, Fig. 2(c) shows the trade-off between the peak accuracy and midpoint noise level of 4-bit model and the 8-bit model. The blue solid curves show that the peak accuracy is partially linearly correlated with the scaling factor. Experiments with models quantized to 16-bit did not yield better results than 8-bit. Surprisingly, the 8-bit model outperforms the 4-bit model on both accuracy performance and robustness performance. A possible explanation is that the 8-bit model can represent wider clipping ranges, resulting in smaller clipping errors and consequently improved fitting capabilities.

2) Results with VGG and ResNet: For experiments on VGG, we choose the achitecture VGG-11 using 50% dropout on fully connected layers. We train both the full precision model (fp32) and quantized models for 500 epochs. The full precision model is trained with learning rate $\eta = 1 \times 10^{-3}$, whereas the quantized models require a smaller learning rate of $\eta = 1 \times 10^{-4}$. For experiments on ResNet, we choose the architecture ResNet-18, trained for 500 epochs with learning

[1]While both batch normalization and regularization are important for generalization and thus maximizing test error, they are not used here to avoid interference with noise experiments. Future work will revisit noise and regularization in combination. A discussion on the impact of batch normalization on midpoint noise level can be found in [3].



(a) 8-bit weights and activations

(b) 4-bit weights and activations

(c) Trade-off between accuracy and midpoint noise level μ

Fig. 2. Robustness of LeNet-5 on CIFAR-10, quantized with different bit widths, using either constant scaling (scaling factors s) or dynamic scaling on activations. Note that the x-axis is of logarithmic scale.

TABLE I
ROBUSTNESS OF VGG-11 AND RESNET-18 ON CIFAR-10, QUANTIZED WITH DIFFERENT BIT WIDTHS AND SCALING FACTORS.

Model	Bitwidths	Scaling factors	Peak accuracy (%)	Midpoint noise level μ
VGG-11	fp32	-	**87.7**	0.154 (±0.5%)
	8-bit	dynamic	87.2	0.024 (±0.1%)
		0.5	84.3	0.145 (±0.2%)
		1	82.2	0.2 (±0.2%)
		2	76.8	0.222 (±0.3%)
		3	10.0	0.013 (±0.0%)
	4-bit	dynamic	86.5	0.031 (±0.1%)
		0.5	84.5	0.12 (±0.2%)
		1	82.6	0.177 (±0.2%)
		2	78.3	**0.23 (±0.3%)**
		3	10	0.010 (±0.1%)
ResNet-18	fp32	-	87.0	0.495 (±0.2%)
	8-bit	dynamic	**87.3**	0.452 (±0.3%)
		0.5	87.1	0.526 (±0.3%)
		1	87.0	0. 557 (±0.3%)
		4	86.5	0.577 (±0.3%)
		8	85.7	0.625 (±0.4%)
		10	10	0.05 (±2.5%)
	4-bit	dynamic	86.8	0.281 (±0.3%)
		0.5	86.5	0.492 (±0.4%)
		1	86.7	0.605 (±0.3%)
		4	86.5	0.657 (±0.5%)
		8	86.5	**0.665 (±0.3%)**
		10	10	0.005 (±1.3%)

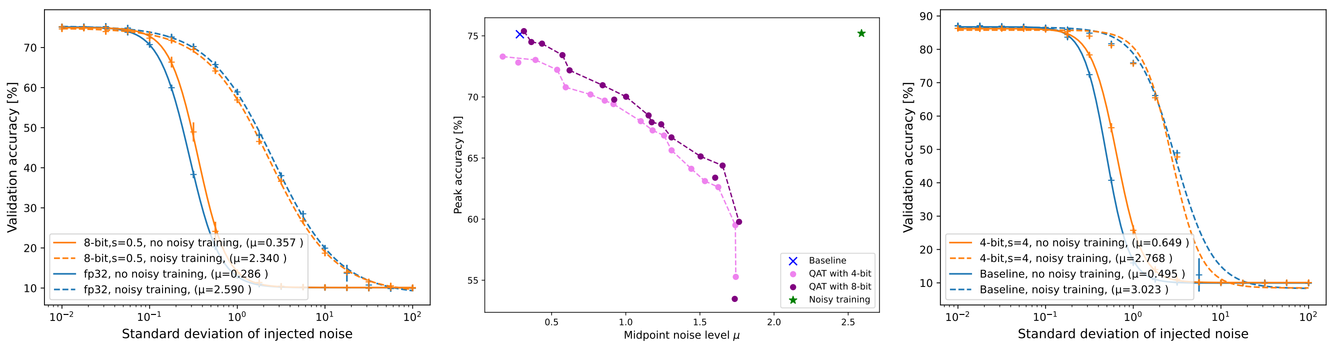
rate $\eta = 0.01$ for both the full precision model and quantized models. The results are shown in Table I.

By comparing dynamic scaling with constant scaling, the advantages of constant scaling become evident. For both 8-bit and 4-bit of VGG-11 and ResNet-18, the midpoint noise levels with dynamic scaling are significantly smaller than those with constant scaling factors, up until the model completely collapses when the constant scaling factor reaches 3 for VGG-11 and 10 for ResNet-18, respectively. When comparing models using dynamic scaling factors to the original floating-point model, it is clear that the dynamic scaling can nearly retain the original peak accuracy, while the midpoint noise levels μ are significantly lower. Overall the conclusions here are identical to LeNet-5, previously.

B. Experiments with Noisy Training.

Noise injection during training is a method used to expose the network to more realistic loss scenarios by randomly perturbing weights and activations. This simulates the conditions of a real noisy analog device, compelling the network to adapt to the noise during the training process. Ideally, training within the noisy analog systems leads to better empirical results, yet understanding the noise dynamics in analog hardware remains challenging.

To compare with noisy training techniques, we train the network while simultaneously injecting noise. The noise intensity



(a) LeNet-5, noisy training with static quantization

(b) Pareto analysis of LeNet-5 for different quantization levels.

(c) ResNet-18, noisy training with static quantization.

Fig. 3. Robustness of LeNet-5 and ResNet-18 on CIFAR-10 with noisy training, combined with quantization.

is matched precisely to that encountered during inference (i.e., $\sigma_{\text{training}} = \sigma_{\text{inference}}$). Results of LeNet-5 and ResNet-18 are shown in Fig. 3, (a) and (c).

From the results in Fig. 3, we observe that the dashed lines in each plot, representing noisy training combined with quantization, are closely matched, almost overlapping. This indicates that, regardless of whether the model is simple or complex, quantization methods can not significantly improve the robustness of models trained with noise injection.

The likely explanation for this inconsistency with the performance observed when models are trained without noise injection is that low precision quantization can degrade the fitting capability of models. This degradation helps explain why quantization does not improve the baseline performance of noisy training with floating-point 32-bit precision. Previous studies support these findings. For instance, Sung, Shin, and Hwang [32] discovered that highly complex DNNs can absorb the effects of severe weight quantization through retraining, while connection-limited networks exhibit less resilience.

Although quantization methods do not improve the robustness of the baseline performance of noisy training, there is a positive aspect: quantized models trained with noise can achieve the same level of robustness as the floating point models while maintaining their accuracy. This finding suggests that it is feasible to deploy quantized models, which have smaller sizes, without sacrificing robustness.

C. Comparison among Architectures

The experiments above reveal significant differences in the performance of various models. An overall comparison is provided in Table II. In general, we make the following observations:

1) **More complex models trade accuracy against robustness.** Without noise injection and quantization, VGG-11 and ResNet-18 achieve higher accuracies (87.7% and 87.0%, respectively) compared to LeNet-5 (75%). However, the midpoint noise level μ of VGG-11 (0.154) is significantly lower than LeNet-5 (0.286). In general, complex models can achieve better accuracy, however, they also involve more multiplications and accumulations, overall leading to more noise involved and, consequently, less robustness. From Table II, it is evident that VGG-11 has more noisy layers than LeNet-5. However, with noisy training, complex models, which typically have a greater capacity to fit data, can learn to become more resilient to noise, thereby achieving higher robustness.
2) **Skip connections reduce the amount of error accumulation, thereby improving robustness.** In contrast, ResNet-18 exhibits significantly more robustness, likely due to the used *skip connections*. These connections allow the original stacked layers to focus on optimizing the residual mapping while the skip connections preserve the primary identity information, which is less exposed to noise. Specifically, for one building block, three noisy layers are present in the original stacked layers, while only one or zero noisy layer is present in the skip connection, depending on whether downsampling is applied or not.
3) **Constant scaling is more robust than dynamic scaling.** For all three architectures, dynamic scaling is worse for robustness than constant scaling, and even worse than non-quantized *fp32* values. To gain a more detailed understanding of this an in-depth analysis of activation distributions and effects related to training dynamics would be required, which is out-of-scope for this work.
4) **Larger scaling factors yield diminishing returns in terms of robustness with increasing model complexity.** On VGG-11 and ResNet-18, larger constant scaling factors result in only slight improvements in midpoint noise levels. For instance, for VGG-11 with 8-bit quantization and a constant scaling factor of 2, the midpoint noise level increases by 44.2%, but accuracy drops from 87.7% to 76.8%, for ResNet-18 with 4-bit quantization and a constant scaling factor of 8, the midpoint noise level increases by 34.3%. In contrast, for LeNet-5 with 8-bit quantization and a constant scaling factor of 8, the midpoint noise level rises by over 200%, while accuracy

TABLE II
OVERALL COMPARISON AMONG ARCHITECTURES ON CIFAR-10.

Model	FLOPS (M)	Param (M)	Noisy layers	Peak accuracy (%)	Midpoint noise level μ	
					w/o Noisy Training	with Noisy Training
LeNet-5	0.66	0.06	12	75	0.286	2.59
VGG-11	276.56	132.86	27	87.7	0.154	2.957
ResNet-18	37.53	11.69	33	86.9	0.494	3.023

only decreases from 75% to 70%. VGG-11 collapses when the scaling factor reaches 3, whereas LeNet-5 can function properly even with an extremely large scaling factors, such as 512. Larger scaling factors come with larger quantization errors. The inferior robustness of VGG-11 can be attributed to the error amplification effect, where quantization errors are amplified through the layers of deep neural networks [21].

V. Conclusion

Analog hardware holds the potential to significantly reduce the latency and energy consumption of neural network inference, however, at the same time is imprecise and introduces noise within computations that limits accuracy in practice. In this work, we investigate the robustness of DNNs affected by analog noise during inference. We employ quantization-aware training and noisy training techniques as robustness enhancing methods.

Our experimental results, conducted on the CIFAR-10 dataset with various models, including LeNet-5, VGG-11 and ResNet-18, indicate that quantization with constant scaling factors can significantly improve the robustness. Large scaling factors, however, lead to accuracy loss. Additionally, by comparing different architectures, we find that deeper network architectures are likely to suffer from error amplification, making them more sensitive to large scaling factors. Our results also show that when models are trained in an environment identical to the noisy conditions experienced during inference, robustness is significantly improved. Despite potential benefits of quantization, it does not improve the robustness of models when noise is injected during training. Overall, our findings highlight the importance of aligning training conditions with anticipated inference noise and suggest that targeted robustness strategies are essential to fully realize the benefits of analog hardware in practical, real-world applications.

Outlook

This study underscores both the potential and the challenges of leveraging analog hardware to improve the speed and energy efficiency of neural network inference, despite the inherent computational noise it introduces. Our findings suggest several promising directions for future work to further enhance robustness in noisy environments. A key question is whether robust performance requires detailed knowledge of system noise or if approximate estimates of noise type and strength could suffice, simplifying practical implementations. Given the observed limitations of quantization alone, evaluating perturbation-based methods as complementary strategies may address weaknesses seen when training does not fully align with noisy inference conditions. Examining the combination of various robustness techniques could reveal potential synergies, strengthening noise resilience without significantly impacting performance.

Additionally, further exploration of architectural factors such as network depth, width, residual connections, and attention mechanisms could clarify their influence on robustness, particularly since deeper architectures are prone to error amplification in noisy contexts. Improving robustness may also benefit from advanced sensitivity evaluation methods that go beyond techniques like walking noise, offering faster and more precise insights into the layer-specific impact of noise. Exploring robustness techniques adapted to each layer's unique sensitivity profile within a model may further enhance resilience. Advancing these research directions holds promise for developing robust, energy-efficient models optimized for noise-prone, analog hardware.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. “Estimating or propagating gradients through stochastic neurons for conditional computation”. In: *ArXiv* abs/1308.3432 (2013).

[2] Jonathan Binas et al. “Precise deep neural network computation on imprecise low-power analog hardware”. In: *ArXiv* abs/1606.07786 (2016).

[3] Hendrik Borras, Bernhard Klein, and Holger Fröning. “Walking Noise: On Layer-Specific Robustness of Neural Architectures against Noisy Computations and Associated Characteristic Learning Dynamics”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. ECML-PKDD. 2024.

[4] Frank Brückerhoff-Plückelmann et al. “Probabilistic Photonic Computing with Chaotic Light”. In: *CoRR* abs/2401.17915 (2024). DOI: 10.48550/arXiv.2401.17915.

[5] Brian Chmiel et al. “Robust quantization: One model to rule them all”. In: *Advances in Neural Information Processing Systems*. Vol. 33. NeurIPS. 2020. DOI: 10.5555/3495724.3496170.

[6] Yide Du et al. “Exploring the impact of random telegraph noise-induced accuracy loss on resistive RAM-based deep neural network”. In: *IEEE Transactions on Electron Devices* 67 (2020).

[7] Kirsty Duncan et al. “Relative robustness of quantized neural networks against adversarial attacks”. In: *2020 International Joint Conference on Neural Networks*. IJCNN. IEEE. 2020.

[8] Yannick Emonds, Kai Xi, and Holger Fröning. “Implications of Noise in Resistive Memory on Deep Neural Networks for Image Classification”. In: *CoRR* abs/2401.05820 (2024). DOI: 10.48550/ARXIV.2401.05820.

[9] Amir Gholami et al. “A survey of quantization methods for efficient neural network inference”. In: *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022.

[10] Mirco Giacobbe, Thomas A Henzinger, and Mathias Lechner. “How many bits does it take to quantize your neural network?” In: *Tools and Algorithms for the Construction and Analysis of Systems*. TACAS. Springer. 2020. DOI: 10.1007/978-3-030-45237-7_5.

[11] Micah Gorsline, James Smith, and Cory Merkel. “On the adversarial robustness of quantized neural networks”. In: *Great Lakes Symposium on VLSI*. 2021.

[12] Kaiming He et al. “Deep residual learning for image recognition”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR. 2016.

[13] Berivan Isik et al. “Neural network compression for noisy storage devices”. In: *ACM Transactions on Embedded Computing Systems* 22 (2023).

[14] Vinay Joshi et al. “Accurate deep neural network inference using computational phase-change memory”. In: *Nature Communications* 11 (2020).

[15] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations*. ICLR (2014).

[16] Bernhard Klein et al. “Towards Addressing Noise and Static Variations of Analog Computations Using Efficient Retraining”. In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases - International Workshops of ECML PKDD 2021, Proceedings Part I*. Vol. 1524. Communications in Computer and Information Science. Springer, 2021, pp. 409–420. DOI: 10.1007/978-3-030-93736-2_32.

[17] Raghuraman Krishnamoorthi. “Quantizing deep convolutional networks for efficient inference: A whitepaper”. In: *ArXiv* abs/1806.08342 (2018).

[18] Alex Krizhevsky and Geoffrey E. Hinton. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto, 2009.

[19] Lisa Kuhn, Bernhard Klein, and Holger Fröning. “On the Non-Associativity of Analog Computations”. In: *CoRR* abs/2309.14292 (2023). DOI: 10.48550/ARXIV.2309.14292.

[20] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86 (1998).

[21] Ji Lin, Chuang Gan, and Song Han. “Defensive Quantization: When Efficiency Meets Robustness”. In: *International Conference on Learning Representations*. ICLR. 2019.

[22] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *International Conference on Learning Representations*. ICLR (2017).

[23] Boris Murmann. “Mixed-signal computing for deep neural network inference”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29 (2020).

[24] Markus Nagel et al. “A White Paper on Neural Network Quantization”. In: *ArXiv* abs/2106.08295 (2021).

[25] Alessandro Pappalardo. *XILINX/brevitas*. 2023. DOI: 10.5281/zenodo.3333552.

[26] Antonio Polino, Razvan Pascanu, and Dan Alistarh. “Model compression via distillation and quantization”. In: *ArXiv* abs/1802.05668 (2018).

[27] Angad S Rekhi et al. “Analog/mixed-signal hardware error modeling for deep learning inference”. In: *56th Annual Design Automation Conference 2019*.

[28] Wolfgang Roth et al. “Resource-Efficient Neural Networks for Embedded Systems”. In: *Journal of Machine Learning Research* 25.50 (2024), pp. 1–51. URL: http://jmlr.org/papers/v25/18-566.html.

[29] Yulin Shao, Soung Chang Liew, and Deniz Gündüz. “Denoising noisy neural networks: A bayesian approach with compensation”. In: *IEEE Transactions on Signal Processing* (2023).

[30] Yichen Shen et al. “Deep learning with coherent nanophotonic circuits”. In: *Nature Photonics* 11 (2017).

[31] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *International Conference on Learning Representations*. ICLR (2014).

[32] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. “Resiliency of deep neural networks under quantization”. In: *ArXiv* abs/1511.06488 (2015).

[33] Nanyang Ye et al. “Improving the robustness of analog deep neural networks through a Bayes-optimized noise injection approach”. In: *Communications Engineering* 2 (2023).

[34] Chuteng Zhou et al. “Noisy Machines: Understanding Noisy Neural Networks and Enhancing Robustness to Analog Hardware Errors Using Distillation”. In: *ArXiv* abs/2001.04974 (2020).