ERDNN: Error-Resilient Deep Neural Networks
With a New Error Correction Layer and
Piece-Wise Rectified Linear Unit

MUHAMMAD SALMAN ALI\textsuperscript{1}, MD. TAUHID BIN IQBAL\textsuperscript{1}, (Member, IEEE), KANG-HO LEE\textsuperscript{1},
ABDUL MUQEET\textsuperscript{1}, SEUNGHYUN LEE\textsuperscript{2}, (Member, IEEE), LOKWON KIM\textsuperscript{1},
AND SUNG-HO BAE\textsuperscript{1}, (Member, IEEE)

\textsuperscript{1}Department of Computer Science and Engineering, Kyung Hee University, Yongin 17104, South Korea
\textsuperscript{2}Department of Electronics Engineering, Kyung Hee University, Yongin 17104, South Korea
Corresponding author: Sung-Ho Bae (shbae@khu.ac.kr)

\section*{ABSTRACT} Deep Learning techniques have been successfully used to solve a wide range of computer vision problems. Due to their high computation complexity, specialized hardware accelerators are being proposed to achieve high performance and efficiency for deep learning-based algorithms. However, soft errors, i.e., bit flipping errors in the layer output, are often caused due to process variation and high energy particles in these hardware systems. These can significantly reduce model accuracy. To remedy this problem, we propose new algorithms that effectively reduce the impact of errors, thus keeping high accuracy. We firstly propose to incorporate an Error Correction Layer (ECL) into neural networks where convolution is performed multiple times in each layer and majority reporting is conducted for the outputs at bit level. We found that ECL can eliminate most errors while bypassing the bit-error when the bits at the same position are corrupted multiple times under the simulated condition. In order to solve this problem, we analyze the impact of errors depending on the position of bits, thus observing that errors in most significant bit (MSB) positions tend to severely corrupt the output of the network compared to the errors in the least significant bit (LSB) positions. According to this observation, we propose a new specialized activation function, called Piece-wise Rectified Linear Unit (PwReLU), which selectively suppresses errors depending on the bit positions, resulting in an increased model resistance against the errors. Compared to existing activation functions, the proposed PwReLU outperforms with large accuracy margins of up-to 20% even with very high bit error rates (BERs). Our extensive experiments show that the proposed ECL and PwReLU work in a complementary manner, achieving comparable accuracy to the error-free networks even at a severe BER of 0.1\% on CIFAR10, CIFAR100, and ImageNet.

\section*{INDEX TERMS} Deep learning, soft error, reliability, error correction layer, piece-wise ReLU, PwReLU, bit error rate.

\section*{I. INTRODUCTION} Although recent deep neural networks (DNNs) have shown promising performance in various areas, their huge computational complexity and large power consumption have impeded deployment of practical and/or real-time DNN applications. Therefore, in the pursuit of making DNNs more efficient, rigorous efforts from four different perspectives have been made, i.e., element, circuit, process (hardware implementation), and algorithm perspectives.

\subsection*{A. PREVIOUS WORK}
1) BUILDING EFFICIENT DNNs AT FOUR LEVELS
At the algorithm level, Han \textit{et al.} [18] proposed Deep Compression, containing pruning, quantization and Huffman coding for the weight values of a DNN to reduce computation complexity and memory access. Chollet \textit{et al.} [8] proposed a depth-wise separable convolution layer that decomposes 3-dimensional convolution into spatial and channel
convolutions, resulting in a significant reduction of computation complexity in the DNN architecture.

At circuit and hardware implementation levels, DeepX [28] proposed a fully pipe-lined deep learning acceleration solution, achieving 4 to 12× faster than the previous Field-programmable gate array (FPGA)-based hardware solutions. In [44] another FPGA solution was proposed which achieved 15× in energy efficiency and about 111× lesser memory transactions due to the use of the on-chip cache.

At element-level, Complementary Metal Oxide Semiconductor (CMOS) which is an industrial standard element, has been used for deep learning. In [48], an efficient CMOS-based DNN system was proposed, where 30% and 36% computation energy can be saved using 3 × 3 and 5 × 5 kernels, respectively. Although CMOS has shown very high stability, it is known that CMOS has the fundamental disadvantage of introducing higher power consumption and lower speeds [24]. Therefore, recent studies focus on using new specialized hardware accelerators containing thousands of parallel processing engines for increasing the computational efficiency of DL algorithms [4], [6], [17].

2) ERROR-PRONENESS IN ADVANCED HARDWARE SYSTEMS

Recently with the popularity of DNN algorithms and techniques, many specialized hardware accelerators have been proposed [4], [5], [17]. These hardware systems enclose several different features to process DNN algorithms fast and efficiently. However certain properties are common to all as: (1) Multiply-accumulate (MAC) operations are computed in parallel due to sparse dependencies of each feature map; and (2) data is strategically cached and reused due to spatial and temporal localities within data and across each feature map [36]. The DNN accelerators use spatial architectures [47] to leverage the first property since they contain massively parallel processing engines that are used to compute MACs.

Soft errors can arise in these specialized hardware systems which cause them to malfunction (application failure, violations of safety and reliability standards). The errors are mainly due to the high energy particles striking the electronic devices [36]. Depending upon the hardware technology and operating conditions, the error rate may vary from 1 × 10⁻⁴ to 1 × 10⁻¹, usually and can go up to 1 × 10⁰ under extreme conditions [19].

Soft errors in DNN algorithms can be disastrous as the majority of them are deployed in safety-critical applications [3], [13]. Therefore error correction is required to maintain the system performance and reliability. For example, in a self-driving car, a soft error can result in misclassification of objects, introducing an erroneous action by the car. Such events result in the violation of standards such as ISO 26262 [11], therefore it should be considered as an important concern.

There has been a huge effort to remedy errors occurring in advanced hardware systems. Hasan et al. [20] proposed a system that works for feature extraction and hand-written digit recognition tasks, where they proposed to inject errors in DNNs during training to learn the error patterns. Feinberg et al. [14] proposed an error-resilient DNN, where arithmetic linear codes and Hamming codes are used as error correction mechanism [25], [46]. However, their proposed error correction methods are complicated in practice and showed effectiveness on a limited dataset (MNIST).

Unlike the above methods being specific to a particular task or dataset, our proposed method offers a generic approach that can be adapted to any dataset and any model even in the presence of very high bit error rates (BERs). To the best of our knowledge, our work to study and remove the bit errors occurring in the deep learning algorithms, is the first comprehensive work on this topic. For reproducibility and use in practice, we resorted to the simplest error correction mechanism, i.e., n-level repetition code, and checked its effectiveness with our proposed activation function.

3) ACTIVATION FUNCTIONS

Neural networks are universal functional approximators and can learn and represent any arbitrary complex function [39].

Initially, a sigmoid function was commonly used as an activation function. However, due to slow convergence rates, vanishing gradient problem [2] and zero centered output the use of sigmoid is limited these days. The sigmoid function is expressed by

\[ \text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)} \]

where \( x \) is a real number. The hyperbolic tangent function (Tanh) was another popular activation function. The hyperbolic tangent function ranges between \([-1, 1]\), hence its output is zero-centered. Similar to the sigmoid function, the hyperbolic tangent function also suffers from vanishing gradient problem [29]. The hyperbolic tangent function is expressed as

\[ \text{Tanh}(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \]

Recently, ReLU (Rectified Linear Unit) has become very popular, since it contains non-linearity as well as reduces the vanishing gradient problem [15]. Therefore, increased use of this activation function is observed in almost every deep learning model. Moreover, in recent research, it has been proved that the convergence rate of ReLU is 6× higher than hyperbolic tangent [33]. The ReLU is expressed as

\[ \text{ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \]

Since there are no arithmetic operations in ReLU, it is computationally very fast and efficient than other functions. Nevertheless, recently proposed new activation functions improve the training time, convergence rates and provide slightly higher accuracy as compared to ReLU such as LReLU [21].

However, these activation functions cease to perform in an error-prone environment. Slight injection of error considerably reduces the accuracy of the model and renders...
it useless. The use of an error correction method may reduce the impact of errors but still, the accuracy of the models using these activation functions remains low, as we observe in our extensive experiments (Section III). These problems lead us to the development of a new activation function that exhibits all features of the ReLU, and at the same time shows its efficacy in an error-prone environment.

B. OUR CONTRIBUTIONS

In this paper, we propose a new error correction framework from the perspective of a robust layer and activation function, easily applicable to standard DNN architectures having soft errors.

Our main contributions in this paper are as follow:

- We propose to incorporate a generalized error correction layer (ECL) into neural networks for error-prone specialized hardware that can detect and remove errors and considerably increases accuracy of the models.
- We propose a new activation function, called PwReLU, to increase the model deterrence against the errors where its learnable parameters are adjusted to maximize error suppression performance independent of error rates and layer position.
- Our comprehensive experiments revealed that ECL and PwReLU yield the complementary effect to achieve high error-resiliency, resulting in only 2-3% accuracy drops compared to the error-free baseline models even in complex image classification datasets (i.e., CIFAR10, CIFAR100, and ImageNet) under the high error rate conditions.

II. METHODOLOGY

A. ERROR INJECTION MODEL

In this work, we focus on the soft errors that occur in the form of bit-flips in the data path of a DNN-accelerator. Our error model is in line with the work done in [7], [16], [42], [52]. One of the major reason of soft errors in these modern hardware systems is due to the striking of high energy particles which cause the hardware to malfunction (for example bit flip).

Since the source of these errors is random, therefore the errors can occur at any bit state independent of bit order and state. Hence they tend to have i.i.d. condition. Therefore, the bit-error behavior is modeled by a Bernoulli distribution as

$$f(k; p) = \begin{cases} q, & \text{if } k = 1 \\ p - 1 - q, & \text{if } k = 0 \end{cases}$$

where $p$ is the Bit Error Rate (BER). Figure 1 shows an example of errors occurring in one variable with 8 bit-depth, where input (a single variable) can be represented as a sequence of target bits and, the input can be distorted at each bit level in a stochastic process. In our experiments, errors are injected in the output of every convolution layer in a DNN similar to [27], [42], [43].

![FIGURE 1. Example of error injection on 8-bits input data. For BER = 0.25, we randomly flip the 2 bits and show the two possibilities of outputs.](image)

B. ERROR CORRECTION LAYER

To build robust DNNs, we propose two complementary methods, i.e., bit-error detection and removal mechanism within a DNN architecture so that the effect of the error is minimized while preserving the overall accuracy. Specifically, we present an ECL that is applied after every convolution layer. That is, convolution operation for every layer is performed multiple times to get different outputs for the same input, where outputs tend to be different due to the presence of soft error in them. These multiple outputs are then fed as an input to ECL for correcting errors.

Figure 2 shows a simple example of the proposed ECL. As shown in figure 2, for the same input, the convolution is performed multiple times (3 times in Figure 2), and the majority voting is performed at each bit-level. The MSB and LSB indicate most significant bit (the highest ordered bit) and least significant bit (the lowest ordered bit) respectively.

Since there is a clear trade-off between the error correction ratio and computation complexity by performing multiple repetitions of convolution, it is important to choose an optimal repetition number. Based on comprehensive experiments, we adopted the 3-level repetition code in our model as a standard, these experiments are discussed in ablation studies in section III. Nevertheless, an increasing number of repetitions will further reduce the error rate at the expense of computational complexity. Error Probability of repetition codes is derived by binomial distribution which is given by

$$P_x(x) = \binom{n}{x} p^x (1 - p)^{n-x}.$$  \hspace{1cm} (2)

With the bit-error model in Eq. (1) and binomial distribution in Eq. (2), the error probability (EP) for the n-level repetition code can be computed as

$$P_x(x) = \sum_{i=(n/2)}^{n} \binom{n}{x} p^x (1 - p)^{n-x},$$  \hspace{1cm} (3)

where $P_x(x)$ represents the total probability of error, $n$ denotes the total number of bits in the sequence, $x$ represents the total number of erred bits, and $p$ represents the BER.

In the 3-level repetition code, a single bit is coded thrice. For example, 0 is coded as 000. When we receive a code as 001 and the probability of our error is less than 1/2, then it is more probable that the transmitted code was 000 instead of 001. Three-level repetition code fails when there are two-bit errors or three-bit errors. For example, if the correct sequence was 000, where the first and last bit are...
flipped i.e. 101 (due to error), the output of such a code would be 1, resulting in a decoding error. Therefore, we can say that error will occur in a case when more than half of the bits are corrupted. From Eq. (3), we can compute $\text{EP}$ for 3-level repetition code as: $P_e(x) = \left(\frac{3}{8}\right) (\text{BER})^2 (1 - \text{BER}) + \left(\frac{1}{8}\right) (\text{BER})^3$, where $\left(\frac{3}{8}\right) (\text{BER})^2 (1 - \text{BER})$ represents the error when an error is occurred in two bits out of three bits, while $\left(\frac{1}{8}\right) (\text{BER})^3$ shows when the error occurred in all the three bits. The reduced error rate is the sum of error probability of the two failed conditions. The proposed ECL reduces the errors significantly while ensuring an overall accuracy boost at the expense of computation overhead which can be compensated with the highly efficient specialized hardware.

These errors may occur at both the MSB and LSB locations. Errors occurring at LSBs may not have much impact on the accuracy of the model, but the errors at MSBs affect the performance significantly. Therefore, along with the ECL, we further devise a new safety mechanism that keeps the errors in check occurring mainly in the most significant bits.

**C. PIECEWISE-ReLU**

Since activation functions enable the model to learn a complex arbitrary mapping from input to output, they have crucially been used in deep neural networks [23], [33], [41], [45], [51]. In this paper, we propose to employ an activation function to suppress errors in error-prone neural networks. We devise a new activation function to serve as selectively reducing errors in MSB. To the best of our knowledge, we are the first to employ an activation function to reduce errors in neural networks.

We propose a new activation function, called Piece-wise Rectified Linear Unit (PwReLU) which contains learnable thresholds in PwReLU in order to suppress high-ordered bit errors. Specifically, we decrease the slope of our proposed activation function by a pre-defined hyper parameter after each threshold, leading to a decreased slope of an activation function gradually. As a result, the reduced feature value minimizes the impact of error due to the bit-flips in higher ordered bits.

The above strategy provides an increased model-deterrence against the error while preserving the accuracy even in the presence of very high BERs. We present $n$-variants of PwReLU depending upon the number of thresholds, as denoted by PwReLU$_n$. PwReLU with single bend is denoted as PwReLU$_1$ while the one with double bends is denoted as PwReLU$_2$ and so on up to PwReLU$_n$. Formally, PwReLU$_n$ is defined as,

$$\text{PwReLU}_n(x | \theta) = \begin{cases} (T_n + x) \times a_n, & \text{if } x \geq T_n \\ \vdots & \\ (T_2 + x) \times a_2, & \text{if } T_2 \leq x \leq T_3 \\ (T_1 + x) \times a_1, & \text{if } T_1 \leq x \leq T_2 \\ x, & \text{if } 0 \leq x \leq T_1 \\ 0, & \text{if } x < 0 \end{cases} \quad (4)$$

where $\{T_1, T_2, \ldots, T_n\}$ is the set of thresholds which are learnable during training, $n$ is the number of bends, and $\{a_1, a_2, \ldots, a_n\}$ is the set of gradients in each bend. Based on empirical results we set the value of $a$ as $[a_1 = 0.5, a_2 = 0.4, a_3 = 0.3, a_4 = 0.2, a_5 = 0.1]$ which are fixed in all experiments.

Figure 3 shows an example of PwReLU with three thresholds, after each threshold, there is a slight decrease in the slope of the function. The bending nature of PwReLU constrains the activation values from exploding and hence keeps the errors, in check. The PwReLU is trained along with the standard back-propagation [49] as

$$\frac{\partial \xi}{\partial T_n} = \sum_{y_i} \frac{\partial \xi}{\partial y_i} \frac{\partial f(y_i)}{\partial T_n}, \quad (5)$$

where $\xi$ is the objective function, and $\frac{\partial \xi}{\partial f(y_i)}$ is the gradient propagated from deeper layers similar to the activation function [21]. Through the back-propagation, the gradient of
TABLE 1. Performance of the proposed method on different Bit Error Rates compared to the baseline accuracy on MNIST, CIFAR10, CIFAR100 and ImageNet datasets. Accuracies are given in percentages (%). Top-5 accuracies for CIFAR100 and ImageNet are given in brackets.

| Model   | Activation | Without Error | 0.05 | 0.1  | 0.2   | 0.05 | 0.1  | 0.2   |
|---------|------------|---------------|------|------|-------|------|------|-------|
| BNN     | Tanh       | 97.00         | 76.00| 59.66| 43.33 | 95.00| 86.00| 58.00 |
| LeNet   | ReLU       | 99.00         | 82.00| 61.66| 25.00 | 97.07| 85.33| 54.00 |
|         | PwReLU     | 99.00         | 78.00| 51.66| 30.00 | 98.00| 91.33| 49.33 |

CIFAR10

| Model   | Activation | Without Error | 0.02 | 0.04  | 0.06  | 0.02 | 0.04  | 0.06  |
|---------|------------|---------------|------|-------|-------|------|-------|-------|
| VGG16   | ReLU       | 93.75         | 14.26| 10.91| 10.28 | 88.74| 58.98| 25.72 |
|         | PwReLU_n   | 93.00 ±0.5    |      |      |       | 76.85| 47.55| 26.03 |
|         |            |               |      |      |       | 91.74| 85.39| 78.94 |
| Alexnet | ReLU       | 86.88         | 45.39| 37.33| 34.04 | 81.01| 68.43| 56.03 |
|         | PwReLU_n   | 86.31 ±0.5    | 72.01| 62.86| 55.75 | 84.63| 80.83| 77.91 |
| Resnet18| ReLU       | 94.36         | 11.25| 10.13| 10.08 | 86.08| 33.48| 13.82 |
|         | PwReLU_n   | 94.56 ±0.5    | 72.04| 27.15| 16.47 | 93.65| 88.57| 77.19 |

CIFAR100

| Model   | Activation | Without Error | 0.01 | 0.02 | 0.03 | 0.01 | 0.02 | 0.03 |
|---------|------------|---------------|------|------|------|------|------|------|
| VGG16   | ReLU       | 68.16 (88.31) | 7.39 (20.21)| 1.87 (7.73)| 1.27 (6.03) | 66.24 (86.77)| 59.53 (82.03)| 46.76 (70.76) |
|         | PwReLU_n   | 70.19 (88.87) ±0.5 | 26.79 (51.10)| 8.65 (22.63)| 3.78 (12.55) | 68.10 (88.28)| 63.78 (85.76)| 57.68 (80.60) |
| Alexnet | ReLU       | 58.69 (84.12) | 9.12 (24.02)| 4.42 (14.30)| 2.88 (11.29) | 54.77 (81.14)| 43.81 (71.10)| 25.99 (55.59) |
|         | PwReLU_n   | 59.95 (84.91) ±0.5 | 18.58 (41.31)| 8.48 (23.49)| 5.50 (16.85) | 58.15 (83.74)| 53.41 (80.21)| 45.25 (73.13) |
| Resnet18| ReLU       | 74.14 (91.79) | 5.71 (17.86)| 1.70 (7.33)| 1.29 (6.22) | 72.73 (91.37)| 67.66 (88.50)| 53.87 (78.29) |
|         | PwReLU_n   | 73.92 (92.05) ±0.5 | 7.26 (22.31)| 1.43 (6.42)| 1.23 (5.33) | 73.20 (91.86)| 69.30 (88.79)| 59.34 (83.03) |

ImageNet

| Model   | Activation | Without Error | 0.001 | 0.005 | 0.01 | 0.001 | 0.005 | 0.01 |
|---------|------------|---------------|-------|-------|------|-------|-------|------|
| VGG16   | ReLU       | 71.63 (90.42) | 9.06 (20.49)| 0.14 (0.69)| 0.09 (0.34) | 71.58 (90.40)| 68.31 (88.33)| 53.72 (77.25) |
|         | PwReLU_n   | 71.56 (90.34) ±0.5 | 60.83 (83.20)| 3.69 (9.08)| 0.51 (2.02) | 71.57 (90.33)| 71.21 (89.99)| 69.47 (89.05) |
| Alexnet | ReLU       | 56.83 (79.90) | 0.38 (1.24)| 0.15 (0.54)| 0.12 (0.51) | 56.25 (79.43)| 35.10 (57.26)| 4.678 (10.09) |
|         | PwReLU_n   | 56.46 (79.76) ±0.5 | 28.63 (49.90)| 1.76 (5.06)| 0.51 (1.70) | 56.36 (79.67)| 54.19 (78.08)| 48.90 (73.34) |
| Resnet18| ReLU       | 67.46 (88.08) | 1.62 (5.55)| 0.14 (0.68)| 0.15 (0.54) | 67.16 (87.90)| 60.70 (83.24)| 26.67 (45.71) |
|         | PwReLU_n   | 67.31 (88.00) ±0.5 |       |       |       |       |       |       |

FIGURE 3. An example of our proposed PwReLU with three thresholds. These thresholds are optimized during training. Note that after each threshold, there is a slight decrease in the slope of the function, helping to minimize the wide spread of feature-value.

$PwReLU_n$ is calculated as

$$
\frac{\partial f(y_i)}{\partial T_n} = \begin{cases} 
  a_n, & \text{if } x \geq T_n \\
  a_2, & \text{if } T_2 \leq x < T_3 \\
  a_1, & \text{if } T_1 \leq x < T_2 \\
  0, & \text{if } 0 \leq x < T_1 \\
  0, & \text{if } x < 0
\end{cases}
$$

To verify the effectiveness of the proposed PwReLU, we conduct comprehensive experiments with five variants of the proposed PwReLU in the next Section.

III. EXPERIMENTAL RESULTS

In this section, we thoroughly explore various aspects of PwReLU and ECL along with their comparative performance on different datasets after varying the error-rates. All the experiments are performed in Pytorch [40] framework. The training process of all the models was independent of errors. Errors were induced during the inference process at bit-level after every convolution operation, as similar to [27], [42], [43]. For experiments, we used MNIST [35], CIFAR10 [32], CIFAR100 [30] and ImageNet [12] dataset. For MNIST we used LeNet [34] and Binarized Neural Network (BNN) [10] while for CIFAR10, CIFAR100 and ImageNet, we used VGG16 [45], Alexnet [33] and Resnet18 [22] architectures. We also compared the performance of different popular activation functions against PwReLU.

A. EXPERIMENTS WITH ERROR CORRECTION LAYER (ECL)

It is known that the accuracy of a fully-trained model drops significantly when an error is injected. The phenomenon can also be understood by looking at the results at Table 1, where we artificially inject errors in the model to observe the variations in performance. However, in this work, we address...
this issue with the proposed Error Correction Layer (ECL), and we observe its efficacy from the elaborate experiments with and without ECL on different models and datasets.

1) EXPERIMENTS ON MNIST
At first, we conduct experiments on MNIST dataset with LeNet and BNN architectures. As we observe, compared to LeNet, BNN has a stronger defense against bit errors. On the contrary, LeNet with a 20% error preserved only one-fourth of its accuracy without ECL. However, with ECL the accuracy got significantly improved, and both the models achieve more than 85% accuracy on 10% bit corruption in the presence of ECL while their accuracy without ECL was notably low. Note that in our experiments with PwReLU (at following subsection), we use five different variants of it; nonetheless, in our tables, we use the best value of PwReLU. Note also that since PwReLU compliments the effect of ECL, we observe from Table 1 that the performance of PwReLU is always better than ReLU.

2) EXPERIMENTS ON CIFAR10
We conduct experiments on CIFAR10 dataset with VGG16, Alexnet, and Resnet18 networks. These models are deep and comparatively very complex, hence even a small error can propagate from initial layers to final layers, resulting in an incorrect prediction. In accordance with the depth of the model, there is high chance for the additive errors to influence the model accuracy. Alexnet is less deeper, and hence the accuracy of Alexnet without ECL is significantly higher compared to VGG16 and Resnet18, as evident from the CIFAR10 results in Table 1. Similar to the previous experiment, PwReLU significantly compliments the performance of ECL, providing comprehensively high accuracy even without ECL for CIFAR10 dataset. At a high BER of 0.06, PwReLU with ECL shows an accuracy boost of 53%, 21%, and 64% in contrast to ReLU on VGG16, Alexnet, and Resnet18, respectively.

3) EXPERIMENTS ON CIFAR100
For CIFAR100 dataset, we observe that the top-5 accuracy of VGG16 using PwReLU with BER 0.01 without ECL is 51.10%, which is about 30% better than ReLU. Without ECL, accuracy is considerably low. ECL significantly increases the accuracy while achieving near baseline accuracy on BER of 0.01. PwReLU consistently shows better results in all error conditions. At a high BER of 0.03, PwReLU with ECL shows a top-1 accuracy boost of 11%, 16%, and 6%, while on top-5 shows an improvement of 10%, 18%, and 5% over ReLU on VGG16, Alexnet, and Resnet18, respectively.

4) EXPERIMENTS ON ImageNet
We also conduct experiments on ImageNet dataset to observe the performance of ECL. Considering the prior experiments, a similar result pattern is observed with ImageNet experiments, where the PwReLU performs consistently better over ReLU. VGG16 without ECL at BER of 0.001 on ImageNet shows an accuracy boost of 50% approximately against ReLU. With ECL, all the models show near baseline accuracy at BER of 0.001. However, with higher BERs, PwReLU consistently performs better. With BER of 0.005, PwReLU shows a top-1 accuracy boost of 3%, 19%, and 3% compared to ReLU, while on a very high BER of 0.01 it shows an improvement of 16%, 44%, and 14% on VGG16, Alexnet, and Resnet18, respectively.

In overall, we observe that ECL with PwReLU achieves near-baseline accuracy on high BERs with deeper models and complex datasets. Moreover, we observe the impact of error in the model is suppressed at a considerable rate in the presence of ECL and PwReLU.

B. EXPERIMENTS WITH PIECEWISE RECTIFIED LINEAR UNIT (PwReLU)
PwReLU suppresses the impact of error utilizing its learned thresholds and decreasing slope, which, in fact, reduces the impact of errors occurring in higher ordered bits. We used five variants of PwReLU in our experiments, and then the results were compared with ReLU. Figure 4 shows the performance of PwReLU, as compared to ReLU on the MNIST dataset, where it is evident that PwReLU2 significantly improves the accuracy. For CIFAR10 dataset, as shown in Figure 5, PwReLU5 significantly outperforms ReLU with a huge margin for all the models. The only downside of PwReLU5 is its baseline accuracy which is slightly less as compared to other activation functions. For CIFAR100 dataset (Figure 6), PwReLU3 for Alexnet and Resnet18 is notably better than the remaining activations. PwReLU1 performs the best for VGG16 with an accuracy that is almost double of ReLU.

From our extensive experiments, we observe that PwReLU successfully suppresses the errors while offering better accuracies as compared to ReLU in all the error conditions. Based on our current experiments and results, PwReLU produces better results than other variants in the case of n=5, and hence, we consider it as the default value for PwReLU.
C. ABLATION STUDIES

1) COMPARISON OF PwReLU

To further validate the performance of PwReLU, we compare its results with other popular activation functions, such as CELU [1], ELU [9], LReLU [21], PReLU [37], ReLU6 [31], RReLU [50], PLU [38] and SReLU [26]. The comparative experiments are done at different error-rates on CIFAR 10 dataset with VGG16 architecture. Results are shown in Figure 7, where we observe that proposed activation function performs the best in presence of soft errors.

Among the other functions, PReLU and PLU are the most error-prone functions. The performance of SReLU is better than all the activation functions except PwReLU, while the remaining activation functions perform similar to ReLU.

2) MODIFYING PwReLU

The proposed PwReLU suppresses errors by decreasing the activation function’s slope, and we verify our hypothesis by comparing the proposed function against ReLU. To further test the efficacy of PwReLU, we evaluate it with another variant of PwReLU, named as Weighted Piecewise ReLU (WPwReLU). For this, we make the parameter α in Equation 4 a trainable parameter to make our PwReLU more generalized. In other words, we tune the point of threshold along with the value of gradient during training. The goal is to check the model’s behavior in the presence of an error when the slope is trained during the training time. We found out empirically that the performance of WPwReLU is relatively low as compared to the standard PwReLU. We believe that by training the gradient, we made PwReLU more complex as the slope can increase in some cases, which may further intensify the error, and hence its performance degrades in the presence of high error. Our experimental results showing comparison of PwReLU5 and WPwReLU5 are given in Figure 9.

However, the right variant of PwReLU for the respective model requires a hyperparameter search which will provide the optimum performance in the presence of soft errors.
3) EFFECTIVENESS OF ECL
We also evaluate the utility of ECL by increasing the number of repetitions in ECL. In theory, if we increase the number of repetitions, the error-rate is supposed to reduce at a considerable rate. If the number of repetitions is infinite, the error will approach to zero. We observed, by increasing the number of repetitions from 3 to 5 and 7 in the repetition-code algorithm, the model can still achieve close to baseline accuracy with very high error rates. Our results are given in Figure 8. Increasing the number of repetitions is a trade-off between accuracy and computation complexity. Based on the operating conditions and resources available, an optimum number of repetitions can be used to achieve the desired accuracy.

4) COMPUTATION COMPLEXITY
The proposed ECL considerably increases the computation complexity of the model. The redundancy in convolution operation increases the number of FLOPS and time complexity of the model. Due to the repetition of convolution operations in the proposed ECL module, the number of MAC operations is increased by more than $2 \times$. The computation complexity in terms of GMACs is given in Table 2. We also calculated the running time of the model on the CIFAR10 test set under the standard conditions with bit error and compared it with the running time of the model in the presence of ECL. The results are given in Table 3. All the calculations were performed on Inteli9-7900X with GTX 1080 Ti GPU.

IV. CONCLUSION
Recent specialized hardware systems are highly parallel architectures that are ideal for the implementation of DNNs; nonetheless, the soft errors occurring due to their process variation and complex circuitry render them useless. In this paper, we propose a trainable activation function PwReLU along with ECL to provide deterrence against soft errors occurring in these systems. DNNs equipped with our proposed activation function offer higher accuracy due to the suppression of high-valued errors that were not detected by ECL. With ECL, the proposed activation unit can yield considerably better accuracy compared to other activation
functions. As there are trainable parameters and stochastic processes in PwReLU and ECL, our proposed framework is generalized for any error types, making it deployable in various practical applications of DNNs.

REFERENCES

[1] J. T. Barron, “Continuously differentiable exponential linear units,” 2017, arXiv:1704.07483. [Online]. Available: http://arxiv.org/abs/1704.07483

[2] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 157–166, Mar. 1994.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016, arXiv:1604.07316. [Online]. Available: http://arxiv.org/abs/1604.07316

[4] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, “DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning.” ACM SIGPLAN Notices, vol. 49, no. 4, pp. 269–284, Apr. 2014.

[5] Y.-H. Chen, J. Emer, and V. Sze, “Eyresis: A spatial architecture for energy-efficient dataflow for convolutional neural networks,” ACM SIGARCH Comput. Archit. News, vol. 44, no. 3, pp. 367–379, 2016.

[6] Y. Chen, T. Luo, S. Liu, S. Zhang, H. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, “DuDianNao: A machine-learning supercomputer,” in Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture, Dec. 2014, pp. 609–622.

[7] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, “BinFI: An efficient fault injector for safety-critical machine learning systems,” in Proc. Int. Conf. High Perf. Comput., Netw., Storage Anal., Nov. 2019, pp. 1–23.

[8] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1251–1258.

[9] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units,” 2015, arXiv:1511.07289. [Online]. Available: http://arxiv.org/abs/1511.07289

[10] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1,” 2016, arXiv:1602.02830. [Online]. Available: http://arxiv.org/abs/1602.02830

[11] B. J. Czerny, J. D’Ambrosio, and R. Debouk, “ISCM-PLA: A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2001, pp. 73–74.

[12] A. Krizhevsky and G. Hinton, “Convolutional deep belief networks on CIFAR-10,” Unpublished Manuscript, vol. 40, no. 7, pp. 1–9, 2010.

[13] A. Krizhevsky, V. Nair, and G. Hinton. (2014). The Cifar-10 Dataset. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., pp. 1097–1105, 2012.

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[16] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” [Online]. Available: http://yann.lecun.com/exdb/mnist/

[17] G. Li, S. K.-S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, “Understanding error propagation in deep learning neural network (DNN) accelerators and applications,” in Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal., Nov. 2017, pp. 1–12.

[18] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve deep network learning by exponential linear units (ELUs),” 2015, arXiv:1511.07289. [Online]. Available: http://arxiv.org/abs/1511.07289

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in Proc. NIPS Autodiff Workshop, 2017, pp. 1–4.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1510–1520.

[21] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1026–1034.

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861

[24] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, “Memristor crossbar-based neuromorphic computing system: A case study,” IEEE Trans. Neural Netw. Learn. Syst., vol. 25, no. 10, pp. 1864–1878, Dec. 2014.

[25] M. Ishizaka, M. Shintani, and M. Inoue, “Area-efficient and reliable hybrid CMOS/Memristor ECC circuit for ReRAM storage,” in Proc. IEEE 27th Asian Test Syst. (ATS), Oct. 2018, pp. 167–172.

[26] J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016, pp. 1737–1743.

[27] J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016, pp. 1737–1743.

[28] J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016, pp. 1737–1743.

[29] J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016, pp. 1737–1743.
including expression recognition, combined age, and gender recognition.

Interpretation, efficient deep learning architecture design, and facial analysis,

X. Sun, T. Zhang, C. Cheng, X. Yan, Y. Cai, Y. Yang, and R. Huang, “A memristor-based in-memory computing network for Hamming code error correction,” IEEE Electron Device Lett., vol. 40, no. 7, pp. 1080–1083, Jul. 2019.

J. J. Tihhi, N. C. Crago, and J. S. Emer, “Exploiting spatial architectures for edit distance algorithms,” in Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS), Mar. 2014, pp. 23–34.

J. Wang, J. Lin, and Z. Wang, “Efficient convolution architectures for convolutional neural network,” in Proc. 5th Int. Conf. Wireless Commun. Signal Process. (WCSP), Oct. 2016, pp. 1–5.

P. J. Werbos, “Backpropagation through time: What it does and how to do it,” Proc. IEEE, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” 2015, arXiv:1505.00853. [Online]. Available: http://arxiv.org/abs/1505.00853

X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An extremely efficient convolutional network for mobile devices,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6848–6856.

Z. Zhang, L. Huang, R. Huang, W. Xu, and D. S. Katz. “Quantifying the impact of memory errors in deep learning,” in Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER), Sep. 2019, pp. 1–12.

MUHAMMAD SALMAN ALI received the bachelor’s degree in computer science from the National University of Sciences and Technology (NUST), Islamabad, Pakistan, in 2018. He is currently pursuing the joint M.S. and Ph.D. degree with Kyung Hee University, South Korea. His research interests include deep learning interpretation and efficient deep learning models.

MD. TAHID BIN IQBAL (Member, IEEE) received the bachelor’s degree in information technology from the University of Dhaka, in December 2012, and the Ph.D. degree from the Department of Computer Science and Engineering, Kyung Hee University, South Korea, in February 2019. He is currently working as a Postdoctoral Fellow of the Machine Learning and Visual Computing Laboratory, Kyung Hee University. His research interests include deep learning interpretation, efficient deep learning architecture design, and facial analysis, including expression recognition, combined age, and gender recognition.

KANG-HO LEE received the bachelor’s degree from the Department of Electronics Engineering and the Department of Computer Science and Engineering, Kyung Hee University, South Korea, in February 2019, where he is currently pursuing the master’s degree with the Department of Computer Science and Engineering. His research interests include deep learning model compression and efficient deep learning architecture design.

ABDUL MUQET received the bachelor’s degree from the University of Karachi, Karachi, Pakistan, in 2014. He is currently pursuing the M.S. degree with Kyung Hee University, South Korea. His research interests include deep learning interpretation and efficient deep learning models.

SEUNGHYUN LEE (Member, IEEE) received the B.S. degree from Seoul National University, South Korea, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 2013. He conducted his postdoctoral research at Stanford University, until 2016. He joined the faculty of Kyung Hee University, South Korea, in 2016. His research interests include novel nano-electronic devices and neuromorphic components based on novel nanomaterials and memristors. He was a recipient of the Gold Award from the Material Research Society (MRS) in 2012 and the Ph.D. degree, that is, postdoctoral works were introduced in Michigan and Stanford News.

LOKWON KIM received the Ph.D. degree from the Department of Electrical Engineering, UCLA. He has conducted research on neuromorphic computing for deep learning, system-on-chip architecture, automated optimization methodology for power, area, and the performance of VLSI designs, and reconfigurable computing with the UCLA. He has authored or coauthored ten journal articles and nine conference papers, and holds nine patents.

SUNG-HO BAE (Member, IEEE) received the B.S. degree from Kyung Hee University, South Korea, in 2011, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2012 and 2016, respectively. From 2016 to 2017, he was a Postdoctoral Associate with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), MA, USA. Since 2017, he has been an Assistant Professor with the Department of Computer Science and Engineering, Kyung Hee University. He has been involved in model compression/interpretation for deep neural networks, and inverse problems in image processing and computer vision.

* * *