COLA: Orchestrating Error COding and LeArning for Robust Neural Network Inference Against Hardware Defects

Anlan Yu [1] **Ning Lyu** [1] **Jieming Yin** [2] **Zhiyuan Yan** [1] **Wujie Wen** [1]

Abstract

Error correcting output codes (ECOCs) have been proposed to improve the robustness of deep neural networks (DNNs) against hardware defects of DNN hardware accelerators. Unfortunately, existing efforts suffer from drawbacks that would greatly impact their practicality: 1) robust accuracy (with defects) improvement at the cost of degraded clean accuracy (without defects); 2) no guarantee on better robust or clean accuracy using stronger ECOCs. In this paper, we first shed light on the connection between these drawbacks and error correlation, and then propose a novel comprehensive error decorrelation framework, namely *COLA*. Specifically, we propose to reduce inner layer feature error correlation by 1) adopting a separated architecture, where the last portions of the paths to all output nodes are separated, and 2) orthogonalizing weights in common DNN layers so that the intermediate features are orthogonal with each other. We also propose a regularization technique based on total correlation to mitigate overall error correlation at the outputs. The effectiveness of *COLA* is first analyzed theoretically, and then evaluated experimentally, e.g., up to 6.7% clean accuracy improvement compared with the original DNNs and up to 40% robust accuracy improvement compared to the state-of-the-art ECOC-enhanced DNNs.

1. Introduction

Due to the growing computational complexities of deep neural networks (DNNs), hardware acceleration becomes critical to their practical use (Chen et al., 2019; Hao et al., 2019; Ye et al., 2020; Zhang et al., 2020). However, DNN hardware accelerators often suffer from hardware defects, which result in DNN parameter deviations and hence performance degradation (Li et al., 2017; Zhang et al., 2018; Long et al., 2019; Mittal, 2020; Ibrahim et al., 2020; Dash et al., 2021). Therefore, it is essential to alleviate performance degradation due to hardware defects by improving the robustness of DNNs against parameter deviations.

Inspired by error correction codes (ECCs) widely used in digital communication and storage systems, recent works have adopted error correcting output codes (ECOCs) to increase the robustness of DNNs (Dietterich & Bakiri, 2018; Liu et al., 2019; Liu & Wen, 2019; Verma & Swami, 2019; Song et al., 2021). Instead of mapping sample labels to one-hot labels in the original DNNs, DNNs with ECOCs encode sample labels to codewords of a binary error correction code such as Hadamard code. The Hamming distances between codewords translate into greater inter-class distances. Additionally, DNNs with ECOCs utilize sigmoid output layer activation, which improves DNNs' fault tolerance capability (Verma & Swami, 2019). Consequently, DNNs with ECOCs can correct errors without the knowledge of the error model, thanks to their inherent error correction capability.

Just like ECCs for digital communications, we envision that an ideal ECOC solution dedicated to the state-of-the-art DNNs should satisfy the following requirements: **1) keeping model accuracy** as high as possible regardless of whether the hardware (e.g., memory to store weights) is defect-free (clean accuracy) or not (robust accuracy); **2) improving robust accuracy more prominently** when adopting stronger ECOCs with increased minimum Hamming distances (d_{min}). Unfortunately, there is no systematic study on designing ECOCs for DNNs to satisfy these two requirements. Furthermore, current solutions are far from satisfactory in terms of both aspects (Verma & Swami, 2019; Song et al., 2021). Without loss of generality, we experimentally compare the accuracy of models with and without ECOCs under different levels of weight variations using an example image classification task–AlexNet-CIFAR10 and error model (detailed settings are presented in Section 4.1).

[1]Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, USA [2]School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China. Correspondence to: Zhiyuan Yan <zhy6@lehigh.edu>, Wujie Wen <wuw219@lehigh.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

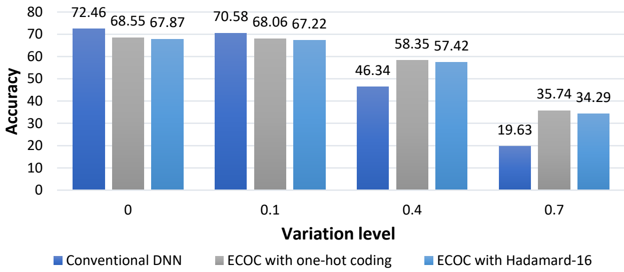


Figure 1. Performance in the presence of hardware defects of AlexNet/CIFAR10 with three network configurations: 1) using one-hot labels and softmax output layer activation; 2) using one-hot labels and sigmoid output layer activation; 3) using Hadamard-16 labels and sigmoid output layer activation. The experimental settings are described in Section 4.1.

Figure 1 shows that when the weights suffer from zero (or very limited) variations, models protected by different strengths of ECOCs (e.g., one-hot or Hadamard-16 coded label) unexpectedly deliver lower accuracy than their counterparts without ECOC. Among ECOC-protected models, the one with a stronger ECOC (Hadamard-16, $d_{min} = 8$) performs even worse than a weaker ECOC (one-hot, $d_{min} = 2$). Figure 1 also shows that while ECOCs improve the model's robust accuracy under high variation levels, their impact is still limited. Again, a stronger ECOC surprisingly offers lower robust accuracy than a weaker one.

In this work, we identify the root cause that fundamentally limits the performance of ECOCs in modern DNNs. At a high level, *our key observation is that errors in DNNs are intrinsically correlated due to the layer-wise convolution-based feature extraction or fully-connected based decision making.* Error correlation significantly diminishes the effectiveness of error correction codings for DNNs. In our work, error broadly refers to the difference between the target and the actual DNN outputs originating from both the **model approximation error** (without hardware defects) and **parameter deviation incurred model error** (with hardware defects). In other words, with or without parameter deviations, in DNNs with ECOCs, the error of each binary classifier is correlated with those of the other classifiers. An in-depth discussion is presented in Section 3.1. Inspired by this, we rethink the design of error coding for DNNs and propose a comprehensive error decorrelation framework that orchestrates the error coding and learning architecture, namely *COLA*, for much improved clean accuracy and robust accuracy when considering weight parameter errors incurred by hardware device defects for both analog and digital DNN hardware accelerators.

Our major contributions are three-fold: 1) We propose an amplitude-adaptive weight orthogonalization (AAWO) method to orthogonalize feature errors on the early layers to prevent error correlation propagation and accumulation. Theoretical analysis shows that the feature errors are approximately independently identically distributed with weight orthogonalized. 2) We propose a regularization technique based on total correlation (TC) to reduce output error correlation rigorously. Theoretical results show that regularizing total correlation potentially lowers classification error probability. 3) We propose a holistic framework for error decorrelation tailored for DNNs, namely *COLA*, by integrating AAWO, separation architecture, and TC regularization across inner and output layers, so as to facilitate the adoption of stronger ECOCs and maximize their error correction capability of improving clean accuracy and robust accuracy.

The evaluations of *COLA* are performed on the MNIST, CIFAR10, CIFAR100, and Tiny ImageNet datasets using Lenet-5, AlexNet, and VGG-16. Experimental results show that *COLA* results in not only clean accuracy improvement compared with the original DNNs but also robust accuracy improvement compared to previous DNNs with ECOCs.

2. Background

2.1. Error Correcting Output Codes (ECOCs)

Let $\mathbf{x} \in \mathbb{R}^d$ and $C \in \mathcal{C}$ be an input to a DNN and its corresponding label, respectively, where $\mathcal{C}$ is the label sample space. The output of the DNN, parameterized by θ, is given by $f(\mathbf{x};\theta) = [f_1(\mathbf{x};\theta), f_2(\mathbf{x};\theta), ..., f_N(\mathbf{x};\theta)]$, where N is the number of outputs. For training purposes, each label C is encoded into a target through a mapping $T(C) : \mathcal{C} \to \mathcal{T}$, e.g., one-hot encoding, and supervised training of the DNN is formulated as

$$\min_{\theta} L(\theta) \triangleq \mathsf{E}_{\mathbf{x}} \left[l\left(f(\mathbf{x};\theta), T(C) \right) \right], \tag{1}$$

where $L(\theta)$ is the training loss and $l(\cdot,\cdot)$ is the sample loss. $\mathbf{x}$, modeled as a random variable, is drawn from a true distribution (infinite dataset) or empirical distribution (finite dataset).

In a classification task with DNN using ECOCs, the codebook consists of $|\mathcal{C}|$ binary codewords with code length N. With sigmoid as the output activation function, each element of $f(\mathbf{x};\theta)$ ranges from 0 to 1. A binary cross entropy (BCE) is used as the loss function. By noticing that $T(C)$ is a binary vector,

$$L_{BCE}(\theta) = \mathsf{E}_{\mathbf{x}} \left[\sum_{n=1}^{N} -\log(1 - U_n) \right], \tag{2}$$

where $T_n(C)$ is the n-th entry of the target $T(C)$ and $U_n \triangleq |f_n(\mathbf{x};\theta) - T_n(C)|$. During inference, the decision is made through the decoding process $D(f(\mathbf{x};\theta)) : [0,1]^N \to \mathcal{C}$.

The decoding process is given by

$$D(f(\mathbf{x};\theta)) = \operatorname*{argmin}_{c\in\mathcal{C}} d(f(\mathbf{x};\theta), T(c)), \tag{3}$$

where $d(\cdot,\cdot)$ is a decoding metric, such as the l_1 or l_2 norm.

We highlight several differences between DNNs with ECOCs and the original DNNs. 1) $T(C)$ maps a label to a designed binary codeword for the former, whereas $T(C)$ is the one-hot encoding for the latter. 2) While the former uses sigmoid $f_n(\mathbf{x};\theta) = \frac{1}{1+e^{-z_n}}$ as output activation function, the latter uses softmax $f_n(\mathbf{x};\theta) = \frac{e^{z_n}}{\sum_{i=1}^{N} e^{z_i}}$, where $z_n = z_n(\mathbf{x};\theta)$ is the output logit (input to activation function). 3) While DNNs with ECOCs use BCE defined in Equation (2) as the loss function, the latter uses categorical cross entropy (CCE)

$$L_{CCE}(\theta) = \mathsf{E}_{\mathbf{x}}\left[\sum_{n=1}^{N} -T_n(C)\log f_n(\mathbf{x};\theta)\right]. \tag{4}$$

2.2. Hardware Defects in DNN Accelerators

The state-of-the-art DNN accelerators are classified as digital and analog, depending on how weights are stored, and multiply-accumulate operations are implemented in circuitry. Both digital and analog DNN accelerators suffer from hardware defects (Ni et al., 2017; Kim et al., 2018).

State error in analog accelerators. Memristive DNN accelerator is one of the most popular analog DNN accelerators because of its low latency and low data movement (Shafiee et al., 2016; Ni et al., 2017). It accelerates matrix-vector multiplication by mapping operands to analog voltages, currents, and conductances on a crossbar structure. Due to the analog nature of the computation in memristive DNN accelerators, the operands, especially weights $\mathbf{W}$ that are stored as conductances on the crossbar, often deviate from their accurate values for various reasons, such as device variation, stuck-at-faults, and electrical noise (He et al., 2019; Liu et al., 2015; Chen et al., 2017). In this paper, we simulate such state error, where the perturbed weights $\tilde{\mathbf{W}}$ follow log-normal distribution: $\tilde{\mathbf{W}} = \mathbf{W} \odot e^{\mathbf{V}}$, where $\odot$ represents the Hadamard product. $\mathbf{V}$ has the same dimensions as $\mathbf{W}$, and the elements in $\mathbf{V}$ follow *Gaussian* distribution $\mathcal{N}(0,\gamma^2)$.

Bit-flip error in digital accelerators. Typical digital DNN accelerators like GPUs have dedicated hardware memory, such as DRAM, to store model parameters. To improve energy efficiency, recently DNN accelerators also decrease the memory supply voltage (Reagen et al., 2016; Kim et al., 2018; Chandramoorthy et al., 2019; Stutz et al., 2021). However, bit error probability increases exponentially as memory supply voltage scales down. In this work, we assume bit error happens independently and randomly on each bit position with a bit flip rate α.

3. Our Approach-*COLA*

3.1. Design Motivation and Overview of *COLA*

To further understand why existing ECOCs achieve both undesirable clean accuracy and robust accuracy for DNNs with or without hardware-incurred weight errors, we observe that error correlation is intrinsically rooted in DNNs for two reasons. **First**, with or without ECOCs, a convolutional or fully-connected layer of a DNN leverages the shared information (e.g., the same input features or neurons) to compute each output feature map or neuron. If there exist errors in the shared information, the errors appearing across different outputs are structurally correlated. When such correlated errors are not decoupled at earlier layers, these errors will propagate and accumulate layer by layer and eventually translate into incorrect decisions at the output layer. **Second**, from the perspective of learning, modern DNNs need to collaboratively train output classifiers via softmax activation to achieve high accuracy (see Equation (4)), while ECOC-enhanced DNNs require the independence of output classifier learning (see Equation (2)) to maximize the error toleration capability with increased minimum Hamming distance. For models with zero or very limited weight variations, the correlated intrinsic model approximation error dominates. While original DNNs penalize such an error by softmax-based collaborative classifier learning, DNNs with ECOCs cannot. Hence, original DNNs without ECOC outperform DNNs with ECOCs in clean accuracy. With more hardware defects, the correlated errors, which are now dominated by errors due to parameter deviation, also increase significantly and quickly exceed the toleration limit of original DNNs. In this case, DNNs with ECOCs with a larger error tolerance margin due to larger d_{min} and the usage of sigmoid output (Verma & Swami, 2019), perform better. However, the improvement is still limited even with stronger ECOCs, since the inflated output error correlation is not explicitly handled in learning.

To reduce error correlation in DNNs, one intuitive way is to physically separate the path from an intermediate layer to the output nodes as proposed in (Verma & Swami, 2019). However, as we shall show in Sec. 4.2, its effectiveness is limited due to the high error correlation on the common layers. Therefore, we believe effectively decorrelating errors before and after the structurally separated intermediate layer is essential. Inspired by this, we propose a comprehensive error decorrelation framework *COLA*. The driving vision of *COLA* is to mitigate the feature error correlation due to model approximation error or model parameter deviations from the inner layers to the output layer through a fine-grained manner so as to improve the clean accuracy and robust accuracy of ECOC-based DNN inference simultaneously. Figure 2 depicts an overview of *COLA*, which mainly consists of inner feature error decorrelation and out-

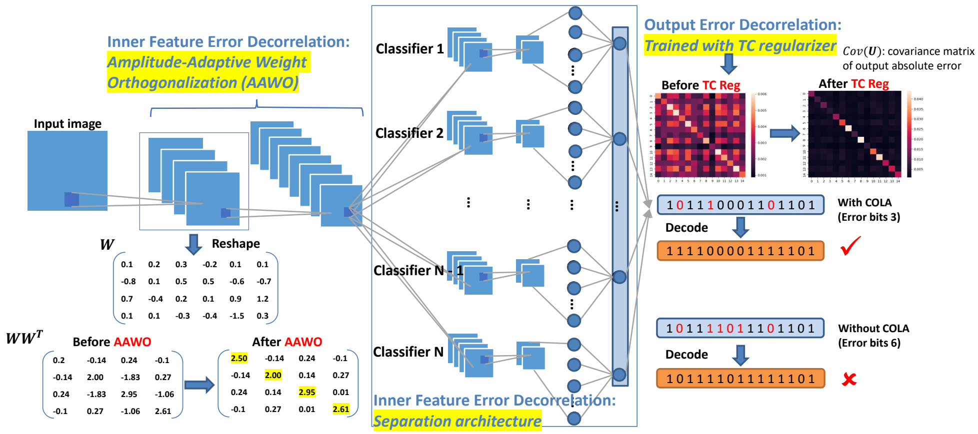


Figure 2. An overview of our proposed *COLA*, including inner feature error decorrelation and output error decorrelation. Inner feature error decorrelation further includes the adoption of separation architecture and our proposed amplitude-adaptive weight orthogonalization (AAWO). By using AAWO, $\mathbf{W}\mathbf{W}^T$ becomes a diagonal matrix. The proposed TC regularizer results in diagonal dominant $Cov(\mathbf{U})$.

put error decorrelation. To decorrelate inner feature errors, we adopt separation architecture to reduce error correlation on the latter layers and propose amplitude-adaptive weight orthogonalization (AAWO) to reduce error correlation on the common layers. Moreover, we propose a regularization technique based on total correlation (TC) to reduce output error correlation. Detailed designs and theoretical analysis are presented below.

3.2. Inner Layer Feature Error Decorrelation

Feature error decorrelation in the early shared layers is vital to the performance of separation architectures. In this subsection, we first analyze how the existing weight orthogonalization (WO), originally proposed to speed up DNN convergence (Huang et al., 2018; 2020), helps decorrelate inner feature errors. We then propose an amplitude-adaptive weight orthogonalization (AAWO) to better suit the needs of ECOCs for separation architectures.

For a shared fully connected layer, indexed m, in a separation architecture, its input-output relation can be

$$\mathbf{o}^m = \sigma(\mathbf{W}^m\mathbf{o}^{m-1} + \mathbf{b}^m), \tag{5}$$

where $\mathbf{o}^m$, σ, $\mathbf{W}^m$, $\mathbf{o}^{m-1}$, and $\mathbf{b}^m$ are the output, activation function, weight matrix, input, and bias of the layer m, respectively. With parameter deviation $\Delta\mathbf{W}^m$ and layer input error $\Delta\mathbf{o}^{m-1}$, the corrupted layer output $\tilde{\mathbf{o}}^m$ is expressed as

$$\begin{aligned}\tilde{\mathbf{o}}^m &= \sigma((\mathbf{W}^m + \Delta\mathbf{W}^m)(\mathbf{o}^{m-1} + \Delta\mathbf{o}^{m-1}) + \mathbf{b}^m) \\ &= \sigma(\mathbf{W}^m\mathbf{o}^{m-1} + \mathbf{W}^m\Delta\mathbf{o}^{m-1} + \Delta\mathbf{W}^m\mathbf{o}^{m-1} \\ &+ \Delta\mathbf{W}^m\Delta\mathbf{o}^{m-1} + \mathbf{b}^m).\end{aligned} \tag{6}$$

In general, the entries of the error term $(\mathbf{W}^m\Delta\mathbf{o}^{m-1} + \Delta\mathbf{W}^m\mathbf{o}^{m-1} + \Delta\mathbf{W}^m\Delta\mathbf{o}^{m-1})$ can be highly correlated. We assume $\Delta\mathbf{W}^m\Delta\mathbf{o}^{m-1}$ is negligible and focus on $\mathbf{W}^m\Delta\mathbf{o}^{m-1}$ and $\Delta\mathbf{W}^m\mathbf{o}^{m-1}$. Assume $\Delta\mathbf{W}^m$ consists of i.i.d. *Gaussian* entries, then $\Delta\mathbf{W}^m\mathbf{o}^{m-1}$ consists of i.i.d. *Gaussian* entries. If the entries of $\Delta\mathbf{o}^{m-1}$ are i.i.d. *Gaussian*, and WO is applied, i.e., $\mathbf{W}^m(\mathbf{W}^m)^T = A_w\mathbf{I}$ with constant A_w, then $\mathbf{W}^m\Delta\mathbf{o}^{m-1}$ consists of i.i.d. *Gaussian* entries. Compared with the desired output, the error term can be viewed as small value, so the activation function can be approximated as linear locally at $(\mathbf{W}^m\mathbf{o}^{m-1} + \mathbf{b}^m)$, which finally makes the whole $(\mathbf{W}^m\Delta\mathbf{o}^{m-1} + \Delta\mathbf{W}^m\mathbf{o}^{m-1} + \Delta\mathbf{W}^m\Delta\mathbf{o}^{m-1})$ approximately independent *Gaussian*. Though the entries of the error term are not identically distributed for a single DNN input across the whole DNN input space, the error can be viewed as i.i.d. *Gaussian* approximately.

Note that both the correlation and the variance of the entries of final output error $\Delta\mathbf{o}$ contribute to the accuracy drop of DNNs with ECOCs. Though WO reduces correlations, the strong constraint $\mathbf{W}^m(\mathbf{W}^m)^T = A_w\mathbf{I}$ makes DNN converge to a point with larger gradient $\nabla_{\mathbf{W}^m}L$ (as it needs to satisfy $\nabla_{\mathbf{W}^m}L + \boldsymbol{\mu}^T\mathbf{W}^m = 0$ instead of $\nabla_{\mathbf{W}^m}L = 0$, where $\boldsymbol{\mu}$ is the *Lagrangian* multiplier). This results in larger absolute value of $\frac{\partial L}{\partial \mathbf{W}^m}$, larger variance of the loss variation $\Delta L \approx \frac{\partial L}{\partial \mathbf{W}^m}\Delta\mathbf{W}^m$ and therefore larger variance of the entries of $\Delta\mathbf{o}$. To balance the correlation and variance, we relax WO and propose AAWO as follows

$$\min_{\theta} L(\theta), \quad s.t. \quad \mathbf{W}^m(\mathbf{W}^m)^T = \mathbf{D}_m \tag{7}$$

where $\mathbf{D}_m$ is a diagonal matrix with trainable diagonal elements. At the price of a little weaker feature error decorrelation, we reduce the variance of the error entries to obtain better robustness as we will show in Section 4.2.3.

For a convolutional layer, indexed m_c, in the shared layers of the separation architecture, we have

$$\begin{aligned}\tilde{\mathbf{o}}^{m_c} = \sigma(&\mathbf{W}^{m_c} \circledast \mathbf{o}^{m_c-1} + \mathbf{W}^{m_c} \circledast \Delta\mathbf{o}^{m_c-1} \\ &+ \Delta\mathbf{W}^{m_c} \circledast \mathbf{o}^{m_c-1} + \Delta\mathbf{W}^{m_c} \circledast \Delta\mathbf{o}^{m_c-1} + \mathbf{b}^{m_c}),\end{aligned} \tag{8}$$

where $\circledast$ is the convolution operation. As a linear operation, convolution can be represented by matrix multiplication

$$\begin{aligned}&Vec(\tilde{\mathbf{o}}^{m_c}) = \sigma(\mathbf{C}_{\mathbf{W}^{m_c}} Vec(\mathbf{o}^{m_c-1}) + \\ &\mathbf{C}_{\mathbf{W}^{m_c}} Vec(\Delta\mathbf{o}^{m_c-1}) + \mathbf{C}_{\mathbf{o}^{m_c-1}} Vec(\Delta\mathbf{W}^{m_c}) + \\ &\mathbf{C}_{\Delta\mathbf{W}^{m_c}} Vec(\Delta\mathbf{o}^{m-1}) + \mathbf{b}^{m_c}),\end{aligned} \tag{9}$$

where $\mathbf{C}_{\mathbf{W}^{m_c}}$ and $\mathbf{C}_{\mathbf{o}^{m_c-1}}$ are the corresponding circular shift matrices so that their matrix multiplications are equivalent to the convolutions, and $Vec(\cdot)$ is vectorization operation. Notice that large random or orthogonal circular shift matrices have good isometric properties (Wright & Ma, 2022), i.e., $\mathbf{C}_{\mathbf{o}^{m_c-1}}\mathbf{C}^T_{\mathbf{o}^{m_c-1}}$ and $\mathbf{C}_{\mathbf{W}^{m_c}}\mathbf{C}^T_{\mathbf{W}^{m_c}}$ are close to multiple of identity matrices. The remaining arguments for i.i.d. error on convolutional layers are the same with fully connected layers. AAWO for convolutional layers can be obtained by first converting $\mathbf{W}$ into a two-dimensional matrix and then optimizing Equation (7).

3.3. Output Error Decorrelation

In this subsection, a regularization technique is proposed to directly penalize output error correlation. We first introduce the concept of total correlation as a measure of error correlation.

Definition 3.1. Let $Z_1, Z_2, \ldots, Z_K$ be random variables, their total correlation $TC(Z_1, Z_2, \ldots, Z_K)$ is defined as

$$TC(Z_1, Z_2, \ldots, Z_K) = \left[\sum_{i=1}^{K} H(Z_i)\right] - H(Z_1, Z_2, \ldots, Z_K), \tag{10}$$

where $H(Z_i)$ is the entropy of the random variable Z_i and $H(Z_1, Z_2, \ldots, Z_K)$ is their joint entropy.

As the distribution of $U_1, U_2, \ldots, U_N$ are intractable in DNNs with ECOCs, we made approximations of their total correlation by viewing them as *Gaussian*.

Lemma 3.2. *Let $Z_1, Z_2, \ldots, Z_K$ be Gaussian distributed random variables with covariance matrix $\mathbf{\Sigma}$, then their total correlation*

$$TC_G(Z_1, Z_2, \ldots, Z_K) = \frac{1}{2}\mathrm{Tr}(\log \mathbf{\Sigma}) - \frac{1}{2}\log|\mathbf{\Sigma}|. \tag{11}$$

By using TC_G as a proxy of output error total correlation, DNNs with ECOCs are trained to minimize

$$L(\theta) = L_{BCE}(\theta) + \lambda TC_G(U_1, U_2, \ldots, U_N). \tag{12}$$

To show how the total correlation influences the performance of DNN with ECOC, we derive the following theorem.

Theorem 3.3. *Let ϵ be the upper bound of the total correlation, i.e., $TC(U_1, U_2, \ldots, U_N) < \epsilon$. Let d_{min} be the minimum Hamming distance of the code, and denote the set*

$$\mathcal{A} = \left\{(u_1, u_2, \ldots, u_N) : \sum_{n=1}^{N} u_n > d_{min}/2\right\}. \tag{13}$$

Let $P_U = P_{U_1,U_2,\ldots,U_N}$ and $\bar{P}_U = P_{U_1}P_{U_2}\ldots P_{U_N}$, the classification error probability P_e is then upper bounded as

$$P_e \le \bar{P}_U(\mathcal{A}) + \sqrt{1 - e^{-\epsilon}}. \tag{14}$$

In Theorem 3.3, $\bar{P}_U$ is a constructed distribution, such that $U_1, U_2, ..., U_N$ are independent and the marginal distribution for each U_n is the same as that for P_U. Theorem 3.3 suggests that the difference between classification error probability under correlated and independent output errors is smaller than a quantity $\sqrt{1 - e^{-\epsilon}}$ determined by output error total correlation. According to Equation (14), as the code length of ECOC goes to infinity, $\bar{P}_U(\mathcal{A})$ vanishes, and the term $\sqrt{1 - e^{-\epsilon}}$ dominates. This explains why merely increasing the code length results in limited improvement in classification accuracy, and why error decorrelation is important for ECOCs. Without any further assumption, the influence of code length and Hamming distance on classification performance is described in the following corollary:

Corollary 3.4. *Let β be a constant such that the BCE loss $L_{BCE} \le N\beta$. Suppose $\beta < \frac{d_{min}}{2N}$, then $\bar{P}_U(\mathcal{A}) \le \exp\left(-\frac{2}{N}\left(\frac{d_{min}}{2} - N\beta\right)^2\right)$, and*

$$P_e \le \exp\left(-\frac{2}{N}\left(\frac{d_{min}}{2} - N\beta\right)^2\right) + \sqrt{1 - e^{-\epsilon}}. \tag{15}$$

Apparently, the bound in Equation (15) becomes smaller when d_{min} increases. However, a larger d_{min} is usually accompanied by larger code length N which tends to increase the value of the bound. To analyze the effects of d_{min} and N together, we define $\nu = \inf_N \frac{d_{min}(N)}{N}$ for any family of codes, then Equation (15) can be written as

$$P_e \le \exp\left(-2N\left(\nu/2 - \beta\right)^2\right) + \sqrt{1 - e^{-\epsilon}}. \tag{16}$$

When β, ν, and ϵ are fixed, increasing N leads to a smaller upper bound for P_e and potentially higher accuracy. Similarly, when N and ϵ are fixed, increasing ν (i.e., increasing minimum Hamming distance d_{min}) or decreasing β (i.e., smaller loss) also results in higher accuracy. Moreover, using the proposed TC regularization leads to smaller ϵ and smaller upper bound for P_e, which further proves the effectiveness of the proposed output error decorrelation method. For the proofs of the results in this subsection, please refer to Appendix A.

Table 1. Accuracy (% in the format of average ± standard deviation) with state error in analog accelerators. γ defines the variation level as introduced in Section 2.2. *Original*, *ECOC* and *ECOC+Sep* (Verma & Swami, 2019) are the benchmarks. *ECOC+Sep+orth*, *ECOC+TC* and *ECOC+TC+Sep+orth* are different combinations of techniques in *COLA*.

	γ	Original	ECOC	ECOC+Sep	*COLA* (Ours)		
					ECOC+Sep+orth	ECOC+TC	ECOC+TC+Sep+orth
LeNet-5 MNIST	0	98.87 ± 0.08	98.82 ± 0.04	98.86 ± 0.05	98.75 ± 0.08	**98.92** ± 0.09	98.79 ± 0.09
	0.1	98.43 ± 0.15	98.59 ± 0.07	98.66 ± 0.04	98.60 ± 0.08	98.68 ± 0.08	**98.71** ± 0.08
	0.4	70.10 ± 2.41	81.09 ± 1.92	82.85 ± 0.93	91.54 ± 0.56	85.98 ± 1.41	**93.12** ± 0.47
	0.7	23.63 ± 1.58	36.10 ± 1.94	36.94 ± 2.19	45.80 ± 1.26	40.90 ± 1.50	**49.50** ± 1.69
AlexNet CIFAR10	0	72.33 ± 0.19	68.04 ± 0.50	71.67 ± 0.71	77.08 ± 0.38	69.74 ± 0.39	**79.04** ± 0.20
	0.1	70.65 ± 0.11	67.45 ± 0.35	71.07 ± 0.63	76.89 ± 0.37	69.41 ± 0.36	**78.88** ± 0.15
	0.3	57.84 ± 0.43	62.61 ± 0.58	65.67 ± 0.33	75.04 ± 0.19	65.69 ± 0.23	**76.53** ± 0.17
	0.5	36.11 ± 0.77	51.99 ± 0.61	54.48 ± 0.83	68.49 ± 0.69	56.34 ± 0.54	**68.55** ± 0.20
VGG-16 CIFAR100	0	68.16 ± 0.52	49.06 ± 0.55	68.84 ± 0.41	68.82 ± 0.12	71.19 ± 0.15	**71.30** ± 0.29
	0.1	64.74 ± 0.95	48.07 ± 0.49	66.92 ± 0.45	67.74 ± 0.37	68.89 ± 0.39	**70.16** ± 0.35
	0.2	51.38 ± 1.56	44.00 ± 0.42	60.44 ± 0.34	66.12 ± 0.31	61.19 ± 0.58	**68.60** ± 0.37
	0.3	25.11 ± 1.18	33.06 ± 0.79	43.29 ± 0.89	62.93 ± 0.23	39.52 ± 0.55	**64.25** ± 0.13

4. Evaluation

4.1. Experimental Settings

We use Tensorflow as our implementation framework. All simulations are conducted in a workstation with one AMD Ryzen Thread ripper 2990WX 32-core processor and four NVIDIA GeForce RTX 2080Ti GPUs.

Datasets We evaluate and compare the performance of *COLA* with various baselines on four datasets: MNIST (LeCun, 1998), CIFAR10/CIFAR100 (Krizhevsky et al., 2009), and Tiny ImageNet (Russakovsky et al., 2015).

Models We apply *COLA* across different DNN models. Specifically, LeNet-5 and AlexNet are used to evaluate MNIST and CIFAR10, respectively. To evaluate the scalability and sensitivity of *COLA* in complex tasks which often suffer from more prominent learning errors and error correlations at a reasonable fault injection simulation cost, we also extend our evaluation to CIFAR100 and TinyImageNet using VGG-16. Following the separation architectures used in (Verma & Swami, 2019) and (Song et al., 2021), detailed architectures of *COLA* used in the experiment for LeNet-5, AlexNet, and VGG-16 are shown in Figure 5. The code lengths and the number of parameters used for different models and datasets are given in Table 2. For a fair comparison, schemes with ECOCs are designed without significantly increasing the complexity of the original model, measured by the total number of trainable parameters. Model complexities for different configurations are listed in Table 2. To verify the scalability of our proposed *COLA*, we also extend our evaluation to large models such as ResNet-34 and ResNet-50. These additional experimental results can be found in Appendix C.

Codebook selection Besides using one-hot code, we select the codebook $\mathcal{T}$ from Hadamard codes, of which the code length is 2^x and the minimum Hamming distance is 2^{x-1}. We exclude all-zero columns from the code generator matrix.

Table 2. Network Complexity

Network	Dataset	Models	#Parameters
LeNet-5	MNIST	Original	$44,426$
		ECOC-Had15	$44,851$
		Sep-Had15	$51,727$
AlexNet	CIFAR10	Original	$2,472,266$
		ECOC-Had15	$2,473,551$
		ECOC-Had63	$2,485,887$
		Sep-Had15	$876,703$
		Sep-Had63	$2,623,951$
		Sep-orth-Had15	$876,959$
		Sep-orth-Had63	$2,624,207$
VGG-16	CIFAR100	Original	$34,040,228$
		ECOC-Had127	$34,150,847$
		Sep-Had127	$32,994,311$
		Sep-orth-Had127	$32,997,447$
VGG-16	Tiny ImageNet	Original	$40,741,384$
		ECOC-Had255	$40,966,719$
		Sep-Had255	$41,284,989$
		Sep-orth-Had255	$41,288,125$

The code length N is 15, 63, 127 and 255 for MNIST, CIFAR10, CIFAR100 and Tiny ImageNet, respectively.

Error models We evaluate the performance of *COLA* under two scenarios: 1) state errors in analog DNN accelerators, and 2) bit flip errors in digital DNN accelerators as described in Section 2.2. The errors are injected into the models during inference. The variation levels in analog DNN accelerators align with (Liu & Wen, 2019) and the bit-flip rates are chosen according to (Stutz et al., 2021; 2022). Model parameters are uniformly quantized into 8 bits–a common setting in most DNN accelerators (Stutz et al., 2021). Mean accuracy over 100 fault injection simulations is reported.

Evaluation benchmarks We compare the inference accuracy with and without hardware defects between the baseline schemes and *COLA*. The baseline schemes are: 1) the original model with one-hot labels and softmax output activation (referred to as *Original* henceforth for brevity), 2) DNNs with the conventional ECOCs (*ECOC*), and 3) A recent ECOC solution built upon the same separation architecture

Table 3. Performance of VGG-16/Tiny ImageNet with different levels of state errors (γ) and bit-flip rates (α).

Type	Level	Original	ECOC	ECOC+Sep	ECOC+TC+Sep+orth
γ	0	51.73 $\pm$ 0.67	50.53 $\pm$ 1.03	52.04 $\pm$ 0.78	**54.72** $\pm$ 0.19
	0.1	28.09 $\pm$ 1.07	45.42 $\pm$ 0.82	47.43 $\pm$ 0.92	**52.37** $\pm$ 1.01
	0.2	08.19 $\pm$ 1.78	29.76 $\pm$ 2.49	30.86 $\pm$ 1.98	**46.82** $\pm$ 1.87
	0.3	02.94 $\pm$ 1.10	08.75 $\pm$ 1.07	10.23 $\pm$ 1.46	**34.85** $\pm$ 1.66
α	0	50.35 $\pm$ 1.32	48.73 $\pm$ 1.78	51.21 $\pm$ 1.01	**54.59** $\pm$ 0.35
	0.001	45.01 $\pm$ 1.52	48.09 $\pm$ 1.76	50.17 $\pm$ 0.88	**53.74** $\pm$ 0.49
	0.01	02.33 $\pm$ 0.68	28.65 $\pm$ 2.57	30.64 $\pm$ 2.65	**42.54** $\pm$ 2.35
	0.1	00.50 $\pm$ 0.00	00.52 $\pm$ 0.02	00.51 $\pm$ 0.01	**04.57** $\pm$ 0.94

of *COLA* for a fair comparison – (*ECOC+Sep*) (Verma & Swami, 2019). For *COLA*, three different settings are 1) the proposed intermediate layer feature error decorrelation scheme in addition to Sep (*ECOC+Sep+orth*), 2) the proposed output error decorrelation method in combination with ECOCs (*ECOC+TC*), and 3) the combination of these two methods (*ECOC+TC+Sep+orth*).

4.2. Results and Analysis

4.2.1. Performance with Hardware Defects

In this subsection, we evaluate the performance of *COLA* in the presence of analog state errors (γ) and bit-flip errors (α) that occur in weight parameters. The results are summarized in Table 1, Table 3 and Table 4.

Baseline comparison (*Original*, *ECOC*, and *ECOC+Sep*). According to the tables, we observe that standard *ECOC* degrades the clean accuracy (i.e., $\gamma = 0$ or $\alpha = 0$) compared with *Original*. This can be explained with Theorem 3.3 by arguing that the output error here originates from inaccurate model approximation and the loss residual. The magnitude of accuracy drop varies among different datasets. In general, for simple tasks like MNIST, the accuracy drop is trivial. For difficult tasks like Tiny ImageNet (see Table 3), the accuracy drop is also not significant due to the inherent high model approximation error. On the other hand, for CIFAR10 and CIFAR100, we observe a large accuracy drop after using ECOC because the impact of error correlation is more pronounced. Considering hardware defects (i.e., $\gamma > 0$ and $\alpha > 0$), *ECOC* improves the accuracy only slightly in most cases. The effectiveness of *ECOC* has been limited primarily because of error correlation. After introducing *ECOC+Sep* (Verma & Swami, 2019) to reduce error correlation, both clean accuracy and robust accuracy improve but the performance is still not prominent since the error decorrelation by separation only is not comprehensive.

Ablation study and comparison with the state-of-the-art. We observe that all the techniques in *COLA* outperform *ECOC* with no doubts. Specifically, *ECOC+TC* achieves better accuracy than *ECOC*, which can also be supported by the conclusion made in Theorem 3.3. *ECOC+Sep+orth* can

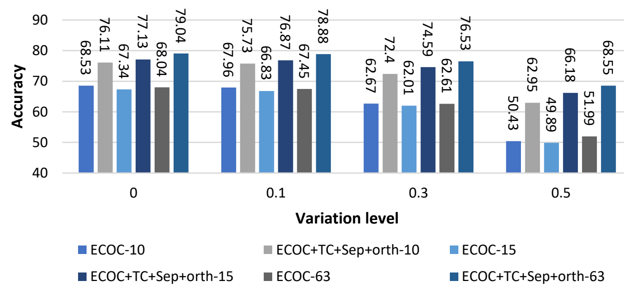


Figure 3. Influence of code length and d_{min}: AlexNet-CIFAR10 under state errors for 6 network configurations: with and without *COLA* when the codebook is one-hot code, Hadamard-15 (Had15), and Hadamard-63 (Had63).

achieve similar or better accuracy than *ECOC+Sep* (Verma & Swami, 2019), because using the additional amplitude-adaptive weight orthogonalization (AAWO) technique results in feature error independence on the early layers, which further leads to error decorrelation. Combining the inner layer feature error decorrelation and output error decorrelation, *ECOC+TC+Sep+orth* achieves the best performance among all, e.g., up to 6.7% improvement on clean accuracy for AlexNet-CIFAR10 and up to 53% improvement on robust accuracy ($\alpha = 0.01$) for VGG-16-CIFAR100 compared with *Original*. Compared with *ECOC*, *ECOC+TC+Sep+orth* achieves up to 40% improvement on robust accuracy ($\alpha = 0.05$) for VGG-16-CIFAR100.

Among different model-dataset combinations, *COLA* works better for more complex datasets. The reason is that complex datasets usually have more output nodes and hence larger error correlation. With the help of *COLA*, error correlation is reduced significantly and the advantage of using ECOCs with a larger code length is further enhanced. In summary, *COLA* can simultaneously improve clean accuracy and robust accuracy for different tasks.

4.2.2. The Influence of Code Length and d_{min}

We use AlexNet-CIFAR10 as an example to demonstrate the influence of code length N and d_{min} on model accuracy. Figure 3 compares the performance of AlexNet under one-hot code, Hadamard-15 (Had15), and Hadamard-63 (Had63) before and after applying *COLA*. As Figure 3 shows, *ECOC*-15 (with $N = 15$ and $d_{min} = 8$) performs worse than *ECOC*-10 (with $N = 10$ and $d_{min} = 2$) at all variation levels. This is counter-intuitive since the code with larger N and d_{min} are expected to provide a larger error margin and better robustness. As illustrated in Equation (16), the accuracy is determined by N, ν, and the error correlation ϵ if the average output error $\frac{1}{N}\sum_n U_n$ is fixed as β. *ECOC*-15 performs worse than *ECOC*-10 because *ECOC*-15 has a stronger error correlation. To further verify this, we then

Table 4. Accuracy (% in the format of average ± standard deviation) with bit-flip error in digital accelerators. α defines the bit-flip rate as introduced in Section 2.2. *Original*, *ECOC* and *ECOC+Sep* (Verma & Swami, 2019) are the benchmarks. *ECOC+Sep+orth*, *ECOC+TC* and *ECOC+TC+Sep+orth* are different combinations of techniques in *COLA*.

	α	Original	ECOC	ECOC+Sep	*COLA* (Ours)		
					ECOC+Sep+orth	ECOC+TC	ECOC+TC+Sep+orth
LeNet-5 MNIST	0	98.82 ± 0.08	98.80 ± 0.10	98.73 ± 0.08	98.61 ± 0.05	**98.86** ± 0.10	98.69 ± 0.10
	0.01	86.45 ± 2.13	89.68 ± 1.83	93.96 ± 0.58	97.29 ± 0.19	95.29 ± 1.51	**97.58** ± 0.25
	0.05	26.48 ± 1.96	33.20 ± 2.40	48.60 ± 2.24	72.31 ± 1.58	57.62 ± 2.42	**73.54** ± 1.00
	0.10	13.95 ± 0.51	14.55 ± 0.68	31.02 ± 1.38	35.05 ± 0.99	26.58 ± 0.78	**38.53** ± 1.17
AlexNet CIFAR10	0	71.69 ± 0.47	67.25 ± 0.39	71.04 ± 0.76	76.91 ± 0.52	69.51 ± 0.51	**78.35** ± 0.45
	0.01	58.56 ± 0.50	60.83 ± 0.67	62.59 ± 0.33	74.53 ± 0.15	64.54 ± 0.37	**76.02** ± 0.26
	0.05	25.64 ± 0.13	31.58 ± 1.31	33.51 ± 1.26	59.02 ± 0.92	38.74 ± 1.66	**62.12** ± 0.58
	0.10	13.72 ± 0.25	15.68 ± 1.52	16.35 ± 1.49	37.66 ± 0.73	16.78 ± 1.17	**41.50** ± 0.79
VGG-16 CIFAR100	0	66.12 ± 0.56	47.97 ± 1.08	68.55 ± 0.48	68.68 ± 0.28	70.55 ± 0.32	**71.13** ± 0.21
	0.001	59.01 ± 0.93	46.15 ± 1.19	68.13 ± 0.63	68.19 ± 0.35	70.32 ± 0.35	**70.57** ± 0.54
	0.010	09.67 ± 1.83	31.62 ± 1.70	60.29 ± 1.45	61.85 ± 1.52	57.99 ± 1.75	**63.24** ± 1.21
	0.050	01.03 ± 0.05	01.47 ± 0.16	04.96 ± 0.25	34.67 ± 1.70	02.58 ± 0.46	**41.51** ± 1.58

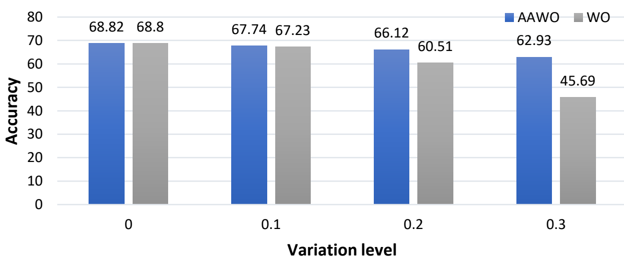


Figure 4. Performance comparison between the proposed amplitude-adaptive weight orthogonalization (AAWO) and original weight orthogonalization (WO) for VGG-16-CIFAR100 under state error in analog accelerators.

apply *COLA* to both *ECOC*-10 and *ECOC*-15 to mitigate error correlation. As expected, *ECOC+TC+Sep+orth*-15 outperforms *ECOC+TC+Sep+orth*-10 because reducing error correlation via *COLA* enables the DNN to truly take advantage of the enhanced error correcting capability of a stronger ECOC. We also observe that *ECOC*-63 performs slightly better than *ECOC*-15. This is because the positive effect of stronger code slightly out-weights the negative aspect of error correlation. After applying *COLA*, we minimize the negative effect of error correlation, and hence *ECOC+TC+Sep+orth*-63 achieves a larger gain over *ECOC+TC+Sep+orth*-15.

4.2.3. Comparison of AAWO with SOTA Weight Orthogonalization

We use VGG-16-CIFAR100 as an example to compare the performance of our proposed amplitude-adaptive weight orthogonalization (AAWO) and the state-of-the-art (SOTA) weight orthogonalization (WO) techniques (Huang et al., 2018; 2020), i.e., encouraging the rows of the weight matrix to be orthonormal. Note that, though both weight orthogonalization techniques are simulated based on the method proposed in (Huang et al., 2020), any method that solves Equation (7) can be used in our scheme. As shown in Figure 4, by relaxing the constraint in Equation (7), the robustness greatly improves, which is consistent with our analysis in Section 3.2. The results demonstrate that a strong constraint could increase error magnitude even though it helps reduce error correlation, which would potentially lead to lower classification accuracy.

4.2.4. Effectiveness of *COLA* on Original DNNs

While the overall goal of our proposed *COLA* is to reduce error correlation, its applicability is not limited to the ECOC framework. In essence, it can be also generalized to improve the performance of original clean DNNs. To verify this, we conduct experiments based on an example setting–the original AlexNet CIFAR10 and then further apply *COLA* and comparable designs to it–the original DNN, the original DNN with *COLA*, and ECOC with *COLA*. Accuracy is evaluated under state errors (γ) in analog accelerators, where γ is chosen as 0 (clean accuracy), 0.1, 0.3 and 0.5 (robust accuracy). Note that the differences between Original-*COLA* and ECOC-*COLA* are: 1) Original-*COLA* uses one-hot codewords while ECOC-*COLA* uses Hadamard codes; 2) the output activation for Original-*COLA* is softmax activation, while ECOC-*COLA* uses sigmoid activation; 3) Original-*COLA* is trained with categorical cross entropy with total correlation regularizer while ECOC-*COLA* is trained with binary cross entropy with total correlation regularizer. We report the corresponding results in Table 5 and make the following observations: 1) After applying *COLA* to original DNNs, clean accuracy ($\gamma = 0$) is improved. This suggests that even though the softmax activation function penalizes error correlation to some extent, original DNNs still suffer from residual error correlation that can be reduced by applying *COLA*; 2) Applying *COLA* to original DNNs also improves robust accuracy ($\gamma = 0.1, 0.3, 0.5$) since *COLA*

Table 5. Performance Comparison of ECOC and the original DNNs on AlexNet/CIFAR10 with *COLA*.

γ	Original	Original-*COLA*	ECOC-*COLA*
0	72.33	76.68	79.04
0.1	70.65	76.29	78.88
0.3	57.84	73.37	76.53
0.5	36.11	66.74	68.55

reduces not only residual error correlation but also state error correlation; 3) In comparison, ECOC-*COLA* outperforms Original-*COLA*, which can be attributed to the larger inter-class distance in ECOC coding. This enlarged distance between codewords in ECOC coding enhances the fault tolerance capability of the network, resulting in improved performance over the original DNNs.

5. Related Works

5.1. Applications of ECOCs

ECOCs have been applied to DNNs to increase fault-tolerance and robustness (Deng et al., 2010; Liu & Wen, 2019; Liu et al., 2019; Verma & Swami, 2019; Song et al., 2021). Deng *et al.* apply ECOC on CNN to achieve a better trade-off between high reliability and low false rejection rate (2010). Liu *et al.* propose a framework with ECOC to increase the reliability of memristive DNN accelerators with specially designed codewords (2019). All of these works fail to satisfy the two aforementioned requirements, that is, keeping model accuracy and improving robust accuracy more prominently. Recent work uses ECOCs against adversarial attacks. Verma *et al.* show that the error margin is enlarged by using sigmoid output layer activation and propose an ensemble-based separation architecture to mitigate the error (2019). Though they use separation architecture to alleviate errors, the achievable effectiveness is limited due to incapable of comprehensively decorrelating feature errors in DNNs.

5.2. Reliable DNN Hardware Accelerators

Analog DNN accelerators like memristive accelerators suffer from state error, stuck-at-faults, and electrical noise. Geng *et al.* propose an on-chip training scheme to compensate for the weight disturbance (2021). An early de-noising scheme is proposed to compensate for the influence of errors at the early layers and prevent error propagation (Yu et al., 2022). A digital offset technique and a method to optimize the digital offset are proposed to reduce the area of the accelerator and compensate for the errors (Meng et al., 2021). These works increase the fault-tolerance by assuming the knowledge of the error model or the application instead of increasing the inherent error correction capability of DNN. Differently, our work focuses on adapting ECOC on DNN so that DNN is inherently fault-tolerant without knowing the error types ahead of time.

For digital DNN accelerators like GPU, errors could occur in weight memory or buffers. Chandramoorthy *et al.* study the impact of bit-flip errors in different layers of DNN and show the accuracy degradation (2019). To reduce the influence of such errors, Srinivasan *et al.* propose storing important bits in the robust cells (2016). Stutz *et al.* propose a comprehensive scheme that combines random bit error training, robust fix point quantization, and weight clipping to improve the inherent robustness of DNN against bit-flip error (2021). Nevertheless, such protections are only effective on bit-flip errors, while our proposed methods are able to protect DNN against different kinds of errors, including bit-flip errors.

6. Conclusion

In this paper, we identify a fundamental limitation of applying ECOCs to DNNs: error correlation. Inspired by this, we rethink the design of error coding for DNNs, and propose a comprehensive error decorrelation framework *COLA* to improve both clean accuracy and robust accuracy. First, we propose amplitude-adaptive weight orthogonalization (AAWO) on the early layers to reduce error correlation propagation and accumulation. Second, we propose a regularization technique based on total correlation to mitigate output error correlation. Third, we propose a holistic framework for error decorrelation in DNNs, including AAWO, separation architecture and total correlation regularization across inner and output layers, so as to facilitate the adoption of stronger ECOCs and maximize their impact on both clean accuracy and robust accuracy. Experimental results based on different models show that our proposed techniques achieve up to 53% accuracy improvement.

Acknowledgements

This work is partially supported by the National Science Foundation (NSF) under Grant No. 2011236, Grant No. 2006748, Grant No. 2238873 and Grant No. 1835278, as well as by Lehigh University under an Accelerator Grant.

References

Bretagnolle, J. and Huber, C. Estimation des densités: risque minimax. *Séminaire de probabilités de Strasbourg*, 12: 342–363, 1978.

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

International Symposium on High Performance Computer Architecture (HPCA), pp. 147–158. IEEE, 2019.

Chen, K.-C., Ebrahimi, M., Wang, T.-Y., and Yang, Y.-C. Noc-based dnn accelerator: A future design paradigm. In *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, pp. 1–8, 2019.

Chen, L. et al. Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 19–24, 2017.

Dash, S., Luo, Y., Lu, A., Yu, S., and Mukhopadhyay, S. Robust processing-in-memory with multibit reram using hessian-driven mixed-precision computation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(4):1006–1019, 2021.

Deng, H., Stathopoulos, G., and Suen, C. Y. Applying error-correcting output coding to enhance convolutional neural network for target detection and pattern recognition. In *2010 20th International Conference on Pattern Recognition*, pp. 4291–4294. IEEE, 2010.

Dietterich, T. G. and Bakiri, G. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *The Mathematics of Generalization*, pp. 395–407. CRC Press, 2018.

Geng, Y., Gao, B., Zhang, Q., Zhang, W., Yao, P., Xi, Y., Lin, Y., Chen, J., Tang, J., Wu, H., et al. An on-chip layer-wise training method for rram based computing-in-memory chips. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 248–251. IEEE, 2021.

Hao, C., Zhang, X., Li, Y., Huang, S., Xiong, J., Rupnow, K., Hwu, W.-m., and Chen, D. Fpga/dnn co-design: An efficient design methodology for 1ot intelligence on the edge. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2019.

He, Z., Lin, J., Ewetz, R., Yuan, J.-S., and Fan, D. 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*, pp. 1–6, 2019.

Hoeffding, W. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pp. 409–426. Springer, 1994.

Huang, L., Liu, X., Lang, B., Yu, A., Wang, Y., and Li, B. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Huang, L., Liu, L., Zhu, F., Wan, D., Yuan, Z., Li, B., and Shao, L. Controllable orthogonalization in training dnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6429–6438, 2020.

Ibrahim, Y., Wang, H., Liu, J., Wei, J., Chen, L., Rech, P., Adam, K., and Guo, G. Soft errors in dnn accelerators: A comprehensive review. *Microelectronics Reliability*, 115: 113969, 2020.

Kim, S., Howe, P., Moreau, T., Alaghi, A., Ceze, L., and Sathe, V. Matic: Learning around errors for efficient low-voltage neural network accelerators. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6. IEEE, 2018.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Li, G., Hari, S. K. S., Sullivan, M., Tsai, T., Pattabiraman, K., Emer, J., and Keckler, S. W. 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*, pp. 1–12, 2017.

Liu, B. et al. Vortex: Variation-aware training for memristor X-bar. In *Proceedings of the 52nd Annual Design Automation Conference*, pp. 1–6, 2015.

Liu, T. and Wen, W. Making the fault-tolerance of emerging neural network accelerators scalable. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–5. IEEE, 2019.

Liu, T., Wen, W., Jiang, L., Wang, Y., Yang, C., and Quan, G. A fault-tolerant neural network architecture. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2019.

Long, Y., She, X., and Mukhopadhyay, S. Design of reliable dnn accelerator with un-reliable reram. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1769–1774. IEEE, 2019.

Meng, Z., Oian, W., Zhao, Y., Sun, Y., Yang, R., and Jiang, L. Digital offset for rram-based neuromorphic computing: A novel solution to conquer cycle-to-cycle variation. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1078–1083. IEEE, 2021.

Mittal, S. A survey on modeling and improving reliability of dnn algorithms and accelerators. *Journal of Systems Architecture*, 104:101689, 2020.

Ni, L., Liu, Z., Yu, H., and Joshi, R. V. An energy-efficient digital ReRAM-crossbar-based CNN with bitwise parallelism. *IEEE Journal on Exploratory solid-state computational devices and circuits*, 3:37–46, 2017.

Reagen, B., Whatmough, P., Adolf, R., Rama, S., Lee, H., Lee, S. K., Hernández-Lobato, J. M., Wei, G.-Y., and Brooks, D. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 267–278. IEEE, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S., and Srikumar, V. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.

Song, Y., Kang, Q., Tay, W. P., and Tay, Y. Error-correcting output codes with ensemble diversity for robust learning in neural networks. In *AAAI*, pp. 9722–9729, 2021.

Srinivasan, G., Wijesinghe, P., Sarwar, S. S., Jaiswal, A., and Roy, K. Significance driven hybrid 8t-6t sram for energy-efficient synaptic storage in artificial neural networks. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 151–156. IEEE, 2016.

Stutz, D., Chandramoorthy, N., Hein, M., and Schiele, B. Bit error robustness for energy-efficient dnn accelerators. *Proceedings of Machine Learning and Systems*, 3:569–598, 2021.

Stutz, D., Chandramoorthy, N., Hein, M., and Schiele, B. Random and adversarial bit error robustness: Energy-efficient and secure dnn accelerators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Verma, G. and Swami, A. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Wright, J. and Ma, Y. *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications*. Cambridge University Press, 2022.

Ye, H., Zhang, X., Huang, Z., Chen, G., and Chen, D. Hybriddnn: A framework for high-performance hybrid dnn accelerator design and implementation. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Yu, A., Lyu, N., Wen, W., and Yan, Z. Reliable memristive neural network accelerators based on early denoising and sparsity induction. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 598–603. IEEE, 2022.

Zhang, X., Wang, J., Zhu, C., Lin, Y., Xiong, J., Hwu, W.-m., and Chen, D. Dnnbuilder: An automated tool for building high-performance dnn hardware accelerators for fpgas. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8. IEEE, 2018.

Zhang, X., Ye, H., Wang, J., Lin, Y., Xiong, J., Hwu, W.-m., and Chen, D. Dnnexplorer: a framework for modeling and exploring a novel paradigm of fpga-based dnn accelerator. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

A. Proofs for Section 3

A.1. Proof of Lemma 3.2

Proof. According to the definition of entropy

$$\begin{aligned}\sum_{i=1}^{K} H(Z_i) &= \sum_{i=1}^{K} -\int \log p(z_i)\, dp(z_i) \\ &= -\sum_{i=1}^{K} \mathsf{E}_{Z_i}\left[\log\left(\frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(Z_i-\mu_i)^2}{2\sigma_i^2}}\right)\right] \\ &= \frac{K}{2}(1+\log(2\pi)) + \frac{1}{2}\mathrm{Tr}(\log \mathbf{\Sigma})\end{aligned} \tag{17}$$

where $Z_i \in \mathcal{N}(\mu_i, \sigma_i^2)$.

Define $\mathbf{Z} = [Z_1, Z_2, \ldots, Z_K]$ and $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_K]$, then

$$\begin{aligned}H(\mathbf{Z}) &= -\int \log p(\mathbf{z})\, dp(\mathbf{z}) \\ &= -\mathsf{E}_{\mathbf{Z}}\left[\log\left((2\pi)^{-K/2}|\mathbf{\Sigma}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{Z}-\boldsymbol{\mu})\mathbf{\Sigma}^{-1}(\mathbf{Z}-\boldsymbol{\mu})^T}\right)\right] \\ &= \frac{K}{2}(1+\log(2\pi)) + \frac{1}{2}\log|\mathbf{\Sigma}|\end{aligned} \tag{18}$$

Accordingly,

$$\begin{aligned}TC_G(Z_1, Z_2, \ldots, Z_K) &= \left[\sum_{i=1}^{K} H(Z_i)\right] - H(\mathbf{Z}) \\ &= \frac{1}{2}\mathrm{Tr}(\log \mathbf{\Sigma}) - \frac{1}{2}\log|\mathbf{\Sigma}|\end{aligned} \tag{19}$$

□

A.2. Proof of Theorem 3.3

Proof. As $TC(U_1, U_2, \ldots, U_N) \leq \epsilon$, we have

$$TC(U_1, U_2, \ldots, U_N) = D_{KL}(P_U || \bar{P}_U) \leq \epsilon, \tag{20}$$

where D_{KL} is referred to as Kullback-Leibler (KL) divergence. By the Bretagnolle–Huber inequality (Bretagnolle & Huber, 1978), we have

$$\begin{aligned}\sup_{\mathcal{S}\subset[0,1]^N} |P_U(\mathcal{S}) - \bar{P}_U(\mathcal{S})| &\leq \sqrt{1 - e^{-D_{KL}(P_U||\bar{P}_U)}} \\ &\leq \sqrt{1-e^{-\epsilon}}.\end{aligned} \tag{21}$$

Since the set that leads to decoding error is a subset of $\mathcal{A}$, we have

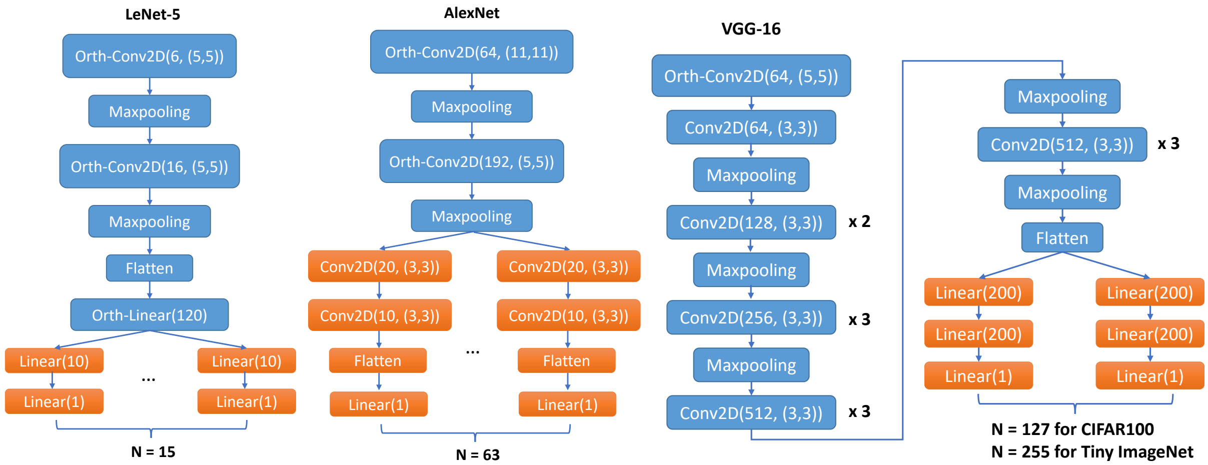
$$P_e \leq P_U(\mathcal{A}) \leq \bar{P}_U(\mathcal{A}) + \sqrt{1-e^{-\epsilon}}. \tag{22}$$

□

A.3. Proof of Corollary 3.4

Proof. By Hoeffding's inequality (Hoeffding, 1994), we have

$$\begin{aligned}\bar{P}_U(\mathcal{A}) &\leq \exp\left(-\frac{2}{N}\left(\frac{d_{min}}{2} - \mathsf{E}\left[\sum_{n=1}^{N} U_n\right]\right)^2\right) \\ &\leq \exp\left(-\frac{2}{N}\left(\frac{d_{min}}{2} - N\beta\right)^2\right).\end{aligned} \tag{23}$$



(a) LeNet-5/MNIST (b) AlexNet/CIFAR10 (c) VGG-16/CIFAR100 and Tiny ImageNet

Figure 5. Detailed architecture for our proposed techniques.

Together with Equation (22) the results can be obtained. □

B. Detailed Architecture for *COLA*

Figure 5 shows the detailed architecture we used in the experiments. In general, the architecture is designed according to (Verma & Swami, 2019), where classifiers are separated after a certain intermediate layer. Amplitude-adaptive weight orthogonalization is used on the first few layers. In order to make a fair comparison, for the same task, the models with different configurations are designed such that the number of parameters used are similar. Model complexity, i.e., total number of parameters, is listed in Table 2. Code is available at `https://github.com/anlanyu66/COLA`.

C. Additional Experimental Results

Additionally, we compare *COLA* and *ECOC* on ResNet-34 and ResNet-50 to verify the scalability of *COLA* on large-sized modern networks. Tiny ImageNet is used as the dataset. Results are given in Table 6 and Table 7 under state errors and bit-flip errors, respectively. These results demonstrate the same trends as the simulation results presented in Section 4. Note that separation architecture (Verma & Swami, 2019) is applied on neither ResNet-34 nor ResNet-50, since it worsens both clean accuracy and robust accuracy. Further investigation on the failure of separation architecture on ResNets will be conducted in our future work.

γ		0	0.1	0.2	0.3
ResNet-34	*ECOC*	55.08	43.68	16.98	0.98
	COLA	59.32	53.42	31.64	4.26
ResNet-50	*ECOC*	56.35	45.24	18.22	1.56
	COLA	60.30	54.12	33.53	5.67

Table 6. Performance of ResNet-34, ResNet-50 evaluated on Tiny ImageNet with different levels of state errors (γ).

α		0	0.001	0.01	0.1
ResNet-34	*ECOC*	54.99	54.12	36.03	0.5
	COLA	58.89	58.09	49.83	2.20
ResNet-50	*ECOC*	56.12	55.28	38.54	0.5
	COLA	59.98	59.12	50.34	3.01

Table 7. Performance of ResNet-34, ResNet-50 evaluated on Tiny ImageNet with different levels of bit flip errors (α).