Coded Deep Neural Networks for Robust Neural Computation

Netanel Raviv[*], **Pulakesh Upadhyaya**[†], **Siddharth Jain**[‡], **Jehoshua Bruck**[‡], **Anxiao (Andrew) Jiang**[†]
[*]Computer Science and Engineering Department, Washington University in St. Louis
[‡]Electrical Engineering Department, California Institute of Technology
[†]Computer Science and Engineering Department, Texas A&M University

I. Introduction

Deep Neural Networks (DNNs) are the revolutionary force supporting AI today. When DNNs are implemented in hardware, their weights are often stored in non-volatile memory devices, including memristor, phase-change memory (PCM), flash memory, CBRAM, spin devices, *etc*. Hardware implementation can help DNNs be used ubiquitously. However, numerous types of errors will appear, making their reliability a critical challenge, which is especially significant for long-term usage of AI systems due to error accumulation and becomes increasingly severe as devices become smaller. To conquer the challenge, new robust DNN architectures are needed.

There exist a number of previous works on building robust neural networks. A typical approach is to compute the same result multiple times (sometimes by replicating each layer several times), and aggregate the results to handle errors in them [4]. Other techniques also exist, including retraining and providing statistical frameworks for testing fault tolerance of DNNs [1, 2].

In this paper, we present a new scheme for robust DNNs called *Coded Deep Neural Network* (CodNN). It transforms the internal structure of DNNs by adding redundant neurons and edges to increase its reliability. The added redundancy can be seen as a new type of error-correcting codes customized for machine learning.

Consider a DNN, which usually has many layers of neurons. Consider two groups of neurons in two adjacent layers, as shown in Fig. 1a. The outputs of the n neurons in the $(\ell-1)$-th layer $v_{\ell-1,1}, v_{\ell-1,2}, \cdots, v_{\ell-1,n}$ are transmitted via the edges to the k neurons in the ℓ-th layer $v_{\ell,1}, v_{\ell,2}, \cdots, v_{\ell,k}$ (multiplied by the edge weights in the process), where they are summed and passed through activation function $\sigma(\cdot)$ to become the k outputs. When errors/erasures appear in the edge weights (e.g., because of errors/erasures in the memory cells that store the weights, or due to circuit faults), the values passed to the k neurons $v_{\ell,1}, v_{\ell,2}, \cdots, v_{\ell,k}$ will be erroneous and their k outputs can be wrong. To make the DNN more fault tolerant, we transform the architecture to a new form, as shown in Fig. 1b, where a new middle layer is added.

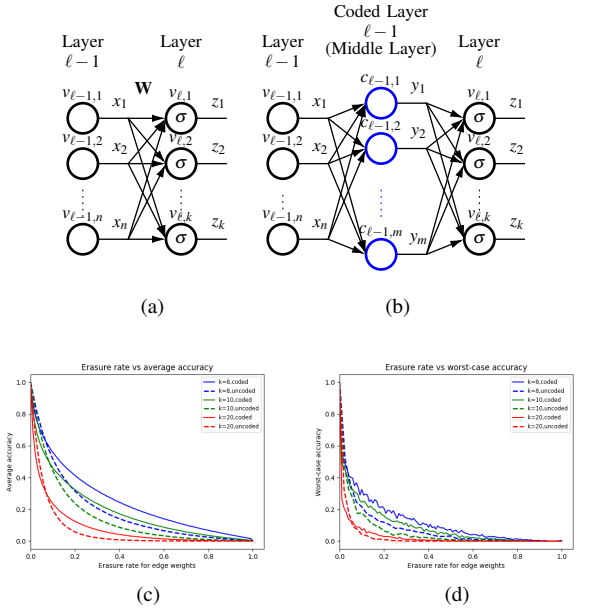

Fig. 1: (a)-(b) An illustration of the Coded Deep Neural Network (CodNN) scheme, where (a) shows neurons in two adjacent layers, and (b) shows a CodNN scheme that adds a new middle layer (which is a coded version of its previous layer). (c)-(d) Improvement in *average accuracy* and *worst-case accuracy* by using CodNN schemes. Here the input layer has $n = 10$ neurons, the middle layer has $m = 20$ neurons, and the output layer has k neurons (for $k = 8, 10, 20$). The solid curves (denoted by "coded") are for CodNN, and the dashed curves (denoted by "uncoded") are for the original neural network component.

The middle layer has m neurons, whose m outputs are a coded version of the n inputs from the previous layer. One important way to use this new architecture is to make the m outputs a codeword of the n inputs with redundancy for fault tolerance; then the m outputs are transmitted to the next layer of k neurons.

The fault-tolerance performance of the architecture in Fig. 1a and 1b can be defined as follows. Assume that the k neurons $v_{\ell,1}, v_{\ell,2}, \cdots, v_{\ell,k}$ perform a classification task, and each of their outputs $z_1, z_2, \cdots, z_k$ takes its value in the set $\{-1, 1\}$. Let $\mathscr{X}$ denote the set of values that the inputs $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ can take with positive probability (the definitions here can

be extended from discrete cases to continuous cases). Let $\mathscr{E}$ denote the set of error patterns for the edge weights. Let $f(\mathbf{x})$ denote the correct outputs of the neurons $v_{l,1},\cdots,v_{\ell,k}$ when the edge weights have no errors; and let $\bar{f}(\mathbf{x},e)$ denote their outputs when the edge weights have the error pattern $e\in\mathscr{E}$. Let $\mathbb{1}(a,b)$ be the indicator function, which equals 1 if $a=b$ and 0 otherwise. We define two metrics for fault-tolerance performance as follows: (1) define the *average accuracy* as $\mathbb{E}_{\mathbf{x}\in\mathscr{X},e\in\mathscr{E}}\left[\mathbb{1}(f(\mathbf{x}),\bar{f}(\mathbf{x},e))\right]$, where $\mathbb{E}(\cdot)$ denotes the expectation of a random variable; (2) define the *worst-case accuracy* as $\min_{e\in\mathscr{E}}\left\{\mathbb{E}_{\mathbf{x}\in\mathscr{X}}\left[\mathbb{1}(f(\mathbf{x}),\bar{f}(\mathbf{x},e))\right]\right\}$. We present designs and analysis for CodNN next.

II. CodNN Construction By Analog Codes

In this section, we present a construction for CodNN based on analog error-correcting codes [5]. Let $\mathbf{W}=(w_{ij})_{n\times k}$ be the weight matrix for Fig. 1a, whose elements are the $n\times k$ edge weights. Let $\mathbf{G}=(g_{ij})_{n\times m}$ be a matrix constructed as follows: from a randomized $m\times m$ orthogonal matrix, delete any $m-n$ rows, then multiply the matrix by a scaling factor $\sqrt{2}$. Let $\mathbf{A}=(a_{ij})_{n\times m}=(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G}$. Then, in the fault-tolerant architecture in Fig. 1b, we let $(y_1,y_2,\cdots,y_m)=(x_1,x_2,\cdots,x_n)\mathbf{G}$, and let $(z_1,z_2,\cdots,z_k)=\sigma((y_1,y_2,\cdots,y_m)\mathbf{A}^T\mathbf{W})$, where $\sigma(\cdot)$ is the activation function.

The coded layer $(y_1,y_2,\cdots,y_m)$ is an analog error correcting code of $(x_1,x_2,\cdots,x_n)$, which contains redundancy and can help correct errors/erasures. As an important case, consider erasures for edge weights, which can be caused by stuck-at errors in memories, by circuit disconnections or signal delay, *etc.* An edge-weight erasure can be seen as changing the edge weight to 0, thus removing the edge from computation.

Let the error model be i.i.d. edge-weight erasures, with erasure probability $p\in[0,1]$. Let the activation functions of the k output neurons be $\text{sign}(\cdot)$. We study how the accuracies of the network in Fig. 1a and 1b change with the erasure probability p. Assume that the n inputs $x_1,x_2,\cdots,x_n$ are real values with a uniform distribution in the range $[-1,1]$. The results (based on extensive experiments) are shown in Fig. 1c and 1d . It can be seen that for a wide range of p, for both average and worst-case accuracies, the CodNN construction outperforms the original neural network significantly.

III. Design and Analysis for CodNN

In this section, we present more analysis and designs for CodNN. A neuron $\tau(\mathbf{x})=\text{sign}(\mathbf{w}\mathbf{x}^\top-\theta)$ is given to the encoder in the form of $(\mathbf{w},\theta)$, where $\mathbf{w}\in\mathbb{R}^n$ and $\mathbf{x}\in\mathbb{F}^n$ for some domain $\mathbb{F}$. The purpose of the encoder is to come up with an encoding function $E:\mathbb{F}^n\to\mathbb{F}^m$ for some $m\geq n$, a vector $\mathbf{v}\in\mathbb{R}^m$, and some $\mu\in\mathbb{R}$, such that the *coded neuron* $\tau'(\mathbf{x})=\text{sign}(\mathbf{v}E(\mathbf{x})^\top-\mu)$ agrees with τ on all possible inputs $\mathbf{x}$, and moreover, outputs $\tau(\mathbf{x})$ even if at most s of the entries of $\mathbf{v}$ are set to zero and at most t of the entries of $\mathbf{v}$ have erroneous values. The former is referred to as s erasures, and the latter as t errors.

Let $\mathscr{H}$ be the hyperplane with a normal vector $\mathbf{v}$, and let $d_1(E(\mathbf{x}),\mathscr{H})$ be the ℓ_1 distance between $E(\mathbf{x})$ and $\mathscr{H}$. Define the minimum distance $d=\min_{\mathbf{x}}d_1(E(\mathbf{x}),\mathscr{H})$. The following theorem characterizes the error/erasure correction capability of a robust neuron based on its minimum distance.

Theorem 1. *For a coded neuron $\tau'(E(\mathbf{x}))=\text{sign}(E(\mathbf{x})\mathbf{v}^\top-\mu)$, if $\tau(\mathbf{x})=\tau'(E(\mathbf{x}))$ for every $\mathbf{x}$, and $2s+t<d$, then τ' can correct any s errors and t erasures in $E(\mathbf{x})$.*

Theorem 2. *For any neuron $\tau(\mathbf{x})=\text{sign}(\mathbf{x}\mathbf{w}^\top-\theta)$, where $\mathbf{x},\mathbf{w}\in\{\pm1\}$ and $\theta\in\mathbb{R}$, there exists a coded neuron $\tau'(\mathbf{x})=\text{sign}(E(\mathbf{x})\mathbf{v}^\top-\mu)$ that can correct one erasure, where $E(x_1,\ldots,x_n)=(x_1,\ldots,x_n,\prod_{i=1}^n x_i)$.*

A. CodNN Design by Fourier Analysis

Given a neuron $\tau(\mathbf{x})=\text{sign}(\mathbf{w}\mathbf{x}^\top-\theta)$ with binary inputs $\mathbf{x}$, let the weight vector $\mathbf{v}$ in the coded neuron τ' be the Fourier spectrum $\hat{\tau}=(\hat{\tau}(S))_{S\subseteq[n]}$ of τ. Let $\chi_S(\mathbf{x})=\prod_{j\in S}x_j$ for every $S\subseteq[n]$.

Theorem 3. *For a neuron $\tau:\{\pm1\}^n\to\{\pm1\}$, the coded neuron $\tau'(E(\mathbf{x}))=\text{sign}(\sum_{S\subseteq[n]}\hat{\tau}(S)\chi_S(\mathbf{x}))$ has minimum distance $\|\hat{\tau}_\varnothing\|_\infty^{-1}$, where $\hat{\tau}_\varnothing=(\hat{\tau}(S))_{S\neq\varnothing}$.*

B. CodNN Design by Finite Frame Theory

For real-valued inputs $\mathbf{x}$, robust neurons can be constructed using finite frames [3]. Let $CH(\tau^+)$ (resp. $CH(\tau^-)$) be the *convex hull* of the positive (resp. negative) points $\tau^+=\{\mathbf{x}|\tau(\mathbf{x})=1\}$ (resp. $\tau^-=\{\mathbf{x}|\tau(\mathbf{x})=-1\}$). Let $d_2(CH(\tau^+),CH(\tau^-))$ denote the minimum ℓ_2 distance between the two convex hulls. Let $\mathbf{M}\in\mathbb{R}^{N\times M}$ be a matrix whose columns form a finite frame.

Theorem 4. *If $d_2(CH(\tau^+),CH(\tau^-))=\delta$ for some $\delta>0$, then $d_2(CH(\tau^+\mathbf{M}),CH(\tau^-\mathbf{M}))\geq s\cdot\delta$, where $\tau^+\mathbf{M}=\{\mathbf{x}\mathbf{M}|\mathbf{x}\in\tau^+\}$ (similarly, $\tau^-\mathbf{M}=\{\mathbf{x}\mathbf{M}|\mathbf{x}\in\tau^-\}$), and s is the smallest singular value of $\mathbf{M}$.*

References

[1] B. S. Arad and A. El-Amawy, "On Fault Tolerant Training of Feedforward Neural Networks", *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 539–553, 1997.

[2] A. E. Barbour and A. S. Wojcik, "A General Constructive Approach to Fault-tolerant Design using Redundancy," *IEEE Transactions on Computers*, vol. 38, no. 1, pp. 15–29, 1989.

[3] P. G. Casazza and Gitta Kutyniok, *Finite Frames: Theory and Applications*. Springer Science & Business Media, 2012.

[4] D. S. Phatak and I. Koren, "Complete and Partial Fault Tolerance of Feedforward Neural Nets," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 446–456, 1995.

[5] K. Xie, J. Li and Y. Liu, "Analysis of Performance of Linear Analog Codes," *arXiv:1511.05509*, 2015.