CODEX: Stochastic Encoding Method to Relax Resistive Crossbar Accelerator Design Requirements

SUBMISSION DATE / POSTED DATE

26-04-2021 / 06-07-2021

CITATION

Amirsoleimani, Amirali; Liu, Tony; Xu, Jianxiong; Alibart, Fabien; Beilliard, Yann; Ecoffey, Serge; et al. (2021): CODEX: Stochastic Encoding Method to Relax Resistive Crossbar Accelerator Design Requirements. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.14485215.v2

DOI

10.36227/techrxiv.14485215.v2

CODEX: Stochastic Encoding Method to Relax Resistive Crossbar Accelerator Design Requirements

Tony Liu, Amirali Amirsoleimani, Jianxiong Xu, Fabien Alibart, Yann Beilliard,
Serge Ecoffey, Dominique Drouin, and Roman Genov

***Abstract*—A stochastic input encoding scheme (CODEX) is presented that aims to relax the digital-to-analog converter (ADC) design requirements in memristor crossbar systems. CODEX reduces the ADC input range by encoding the input bits using Bernoulli statistics so that the bit-line current distribution becomes a narrow Gaussian. By reducing ADC input range, CODEX can be used to reduce ADC power and area or increase ADC resolution for faster in-situ training. Besides input data encoding, CODEX includes probability thresholding for sparse input data as well as a random re-sampling method for dealing with ADC overflow. CODEX is evaluated on CIFAR-10 dataset image classification and reconstruction, sentiment classification, and audio classification. The results show an averaged 68.5% reduction in ADC power and 35.5% reduction in ADC area as well as 25.8% increase in in-situ training speed when applied to the state-of-the-art ISAAC and PUMA accelerators.**

***Index Terms*—Memristor, Analog-to-Digital Converter, Vector-Matrix Multiplication, Inference, Deep Neural Network.**

I. Introduction

RESISTIVE crossbars based on non-volatile memories have garnered increased attention as low-power and high-speed approximate computing accelerators for a variety of computationally-intensive applications including deep neural networks (DNNs) [1]–[5] and neuromorphic computing [6]–[9]. Memristor crossbars perform vector-matrix multiplication (VMM) by mapping a $N \times M$ matrix onto the memristors' conductance range, applying an input voltage vector to the rows, and then sensing the current from the columns. Although, memristor crossbars can significantly increase energy, storage and area efficiency, peripheral input and output circuits will impose a considerable cost [10], [11]. Power consumption in memristor crossbar systems can be separated into on-chip passive power, and peripheral circuit power. Normally, peripheral circuits consume most of the total power ($> 90\%$) with analog-to-digital converters (ADCs) in particular making up over 57% of peripheral circuit power in such platforms [12]. To ensure about the dot-product operation, a high-resolution ADCs (8 to 10 bits) will be required in large resistive crossbar arrays. Such high-resolution ADCs consume higher energy and area in comparison with resistive array and finding appropriate techniques to reduce their design costs will be critical to improve the resistive VMM platform performance metrics. One common method to relieve ADC design constraints include bit-slicing of weights [13] where a single high-resolution weight is split and mapped onto multiple lower-resolution memristors. While effective in reducing ADC resolution requirements, bit-slicing results in severe area overhead. The work in [14] utilizes a quantization algorithm that compresses DNNs to relax ADC resolution constraints in machine learning (ML) applications. In [15] and [16] input encoding method to reduce the input range of the ADC has been presented. However, the encoded inputs in [15], [16] have $\log_2(N/2)$ additional bits where N is the number of rows of the crossbar. These extra input bits can entail a heavy cost in terms of system speed and throughput when used on larger crossbars.

Tony Liu, Amirali Amirsoleimani, Jianxiong Xu and Roman Genov are with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada. Fabien Alibart, Yann Beilliard, Serge Ecoffey, and Dominique Drouin are with the Department of Electrical and Computer Engineering, University of Sherbrooke, QC, Canada. (A. Amirsoleimani and R. Genov corresponding authors email: {amirali.amirsoleimani, roman.genov}@utoronto.ca).

II. Bit-Serial VMM and ADC Characteristics

In order to perform VMM on a memristor crossbar system, the input data must be converted into a series of voltage signals (Fig. 1(a)). To discern all possible output states, J bit ADCs are required where $J = I + W + log_2(N + 1)$. Here, I and W represent the bit-resolution of the input voltage signals and the memristors while N is the number of rows. As such, it is normal that relatively-high resolution ADCs are required for many practical VMM applications. However, it may be possible to use lower-resolution ADCs by considering a valid assumptions on the input data distribution or memristor state distribution. Typically, the most significant bits (MSBs) which hold most of the information will be heavily correlated as shown in Fig. 1(b) while the least significant bits (LSBs) are usually mostly independent as shown in Fig. 1(c). Similarly, the distribution of memristor resistances on the crossbar can vary greatly as can be seen in a small sample of 20 low resistance memristors in Fig. 1(a). The typical sensing circuit for the memristor array usually consists of trans-impedance amplifier (TIA) and ADC [11], in which the TIA converts the memristor array output current to voltage that is digitized by ADC. High-resolution ADC architectures are the main sensing blocks in resistive VMM crossbar arrays to guarantee the accuracy in typical machine learning applications (Fig. 1(d)). The ADC can be loosely classified into two main categories, which are Nyquist-rate converters and oversampling converters. The output of the Nyquist-rate converter (such as the SAR/flash ADC) has a one-to-one correspondence to the input signal. In comparison, the oversampling converters operating much faster than the signal Nyquist frequency to increase the signal-to-noise ratio (SNR). Most resistive crossbar platforms utilizes the Nyquist-rate converter on account of its high throughput, and low power. As shown in Fig. 1(e), the ADC power consumption reduces as the technology nodes becoming smaller, while it is exponentially proportional to the needed effective number of bits (ENOB). Fig. 1(f) illustrates the dynamic range of the ADC is inversely proportional to the sampling frequency and technology nodes for the given power consumption. Fig. 1(g) shows the relationship between the required ADC ENOB and the number of memristors per ADC. The needed resolution of the ADC is linearly proportional to the product of the number of memristors per ADC and the memristor resolution.

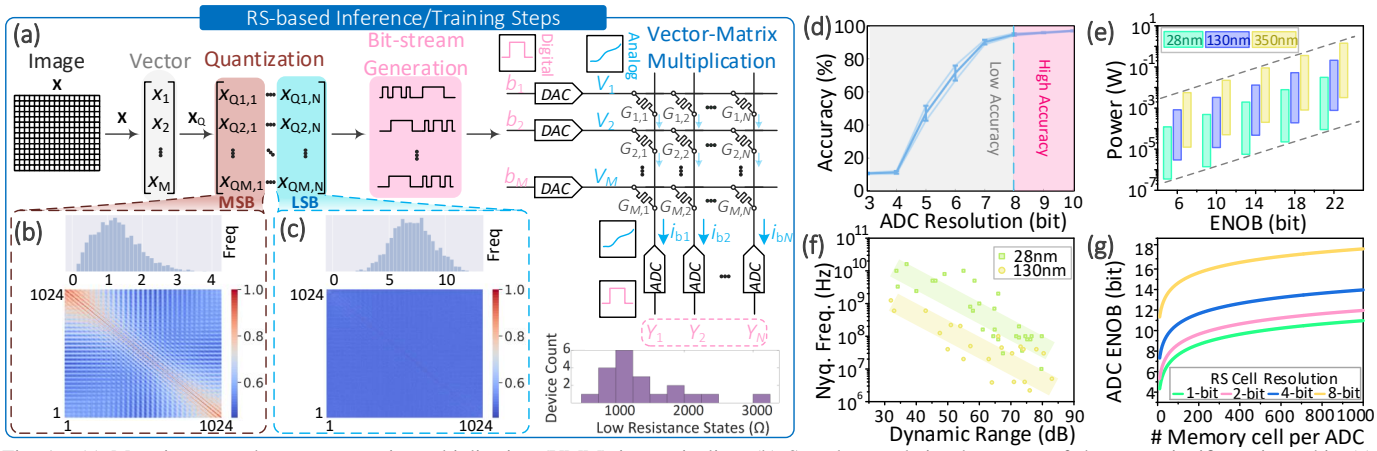

Fig. 1. (a) Memristor crossbar vector-matrix multiplication (VMM) input pipeline. (b) Sample correlation heat map of the most significant input bit. (c) Sample correlation heatmap of the least significant bit. (d) The memristor cross bar array classification accuracy for a given ADC resolution. (e) The ADC power consumption versus the needed ENOB in different technology nodes. (f) The ADC dynamic range versus the sampling frequency in different technology nodes. (g) The relationship among the ADC resolution, memristor cell per ADC and memristor accuracy.

III. Methodology

A. Stochastic Input Modulation

Fig. 2(a) introduces CODEX, which stochastically modulates the input data by adding a randomly-generated binary input vector to each in order to decorrelate the input bits from each other. After CODEX is applied, the decoupled encoded input bits can be modelled as independent, non-identical random Bernoulli variables. As such, the output currents for each column are represented as a weighted sum of N bernoulli variables: $I_j = \sum_i^N w_{i,j} b_i(p)$ where N is the number of rows on the crossbar. Here, I_j is a random variable that represents the current in the j-th column, $w_{i,j}$ is the conductance of the i-th row, j-th column memristor, and $b_i(p)$ is the bernoulli variable for the i-th row input. The mean of I_j is given as $\mu_j = \sum_i^N p_i w_{i,j}$ with variance $\sigma_j^2 = \sum_i^N p_i(1-p_i)w_{i,j}^2$ where p_i is the probability of $b_i(p)$ being 1. As $N \to \infty$, the Lyapunov Central Limit Theorem (CLT) can be applied because Lyapunov's condition is satisfied by I_j for $\delta = 1$:

$$\lim_{N\to\infty} \frac{1}{s_n^{2+\delta}} \sum_i^N (b_i(p) - \mu_{b_i(p)}{}^{2+\delta}) = 0 \quad (1)$$

As a result, $I_j \to \mathcal{N}(\mu_j, \sigma_j^2)$ as $N \to \infty$. Essentially, random inputs with high dimensionality N will result in an output current range that is much smaller than normal. The general flowchart of using CODEX in in-situ and ex-situ applications is shown in Fig. 2(b). The binary input matrix $\mathbf{X} \in R^{N\times k}$ is modulated by adding random binary matrix $\mathbf{U} \in R^{N\times k}$ such that $\mathbf{X'} = \mathbf{X} + \mathbf{U}$ where $\mathbf{X'} \in R^{N\times k+1}$. The random encoding matrix $\mathbf{U}$ is sampled from a Bernoulli distribution with $P(\mathbf{U}_{ij} = 1) = 0.5$ as illustrated in Fig. 2(c). The modulated input $\mathbf{X'}$ is then fed into the memristor crossbar as voltage pulses and the output current is recorded by the ADCs. The intended output current for the m-th bit, $X_m\mathbf{G}$, can be recovered by subtracting $U_m\mathbf{G}$ from the recorded output current: $X_m\mathbf{G} = I_m - U_m\mathbf{G}$ where I_m is the output current of the m-th column and $\mathbf{G}$ is a matrix that represents the conductances of the memristor crossbar. Ideally, a random $\mathbf{U}$ would be sampled for every input vector, but this method would be impractical due to the computational overhead of computing a new $\mathbf{UG}$ for every VMM. Instead, we found that randomly generating a couple fixed $\mathbf{U}$ matrices beforehand is sufficient to produce a roughly normal output current distribution for practical applications. Due to the small pool of encoding matrices U^*, $\mathbf{UG}$ can also be calculated for every $\mathbf{U} \in U^*$ beforehand with negligible overhead.

B. Sparse Input Optimization and ADC Overflow

Each element x_{ij} in the $\mathbf{X}$ input matrix has a probability p_{ij} of being one. Ignoring carry-over from previous bit columns, the probability of modulated input $\mathbf{X'}$ being one can be formulated as:

$$P(x'_{ij} = 1) = \begin{cases} p_{ij}, & \text{if } u_{ij} = 0 \\ 1 - p_{ij}, & \text{if } u_{ij} = 1 \end{cases} \quad (2)$$

To estimate p_{ij}, a random sample of the input can be taken and the relative frequency of ones can be used to approximate p_{ij}. One method of reducing crossbar power consumption is to keep the current running through memristors and bit- or word-lines low by minimizing the frequency of non-zero input voltage pulses. In others words, we would like to minimize $P(x'_{ij} = 1) = p'_{ij}$ for all i and j. To completely minimize p'_{ij}, u_{ij} would be 1 for all $p_{ij} > 0.5$ and 0 otherwise. However, this $\mathbf{U}$ assignment would not be random and would prevent our proposed method from normalizing the output current distribution. On the other hand, fully randomizing $\mathbf{U}$ results in $EX(p_{ij}) = 0.5$ regardless of input data distribution which can result in a significant increase in bit-line current. To address this issue, we propose a probability threshold illustrated in Fig. 2(d) where if p_{ij} is less than the threshold th, then $u_{ij} = 0$. This probability threshold prevents low p_{ij} from being randomly flipped into high p_{ij}, thus, preventing any significant increase in average bit-line current. In addition, $\mathbf{U}$ remains random above th, keeping the output current distribution roughly normal. The main overhead in implementing this concept would be calculating p_{ij} which can be reasonably estimated with a small ($<$1%) random sample of the dataset. Assuming a normal output current distribution, there is a 4.55% chance that the output current falls outside 2σ of the mean and a 0.27% chance for the output to fall outside three σ. No matter what ADC input range is used, there will be a non-zero chance of the output being outside this range. A solution to this issue is to prepare multiple spare random

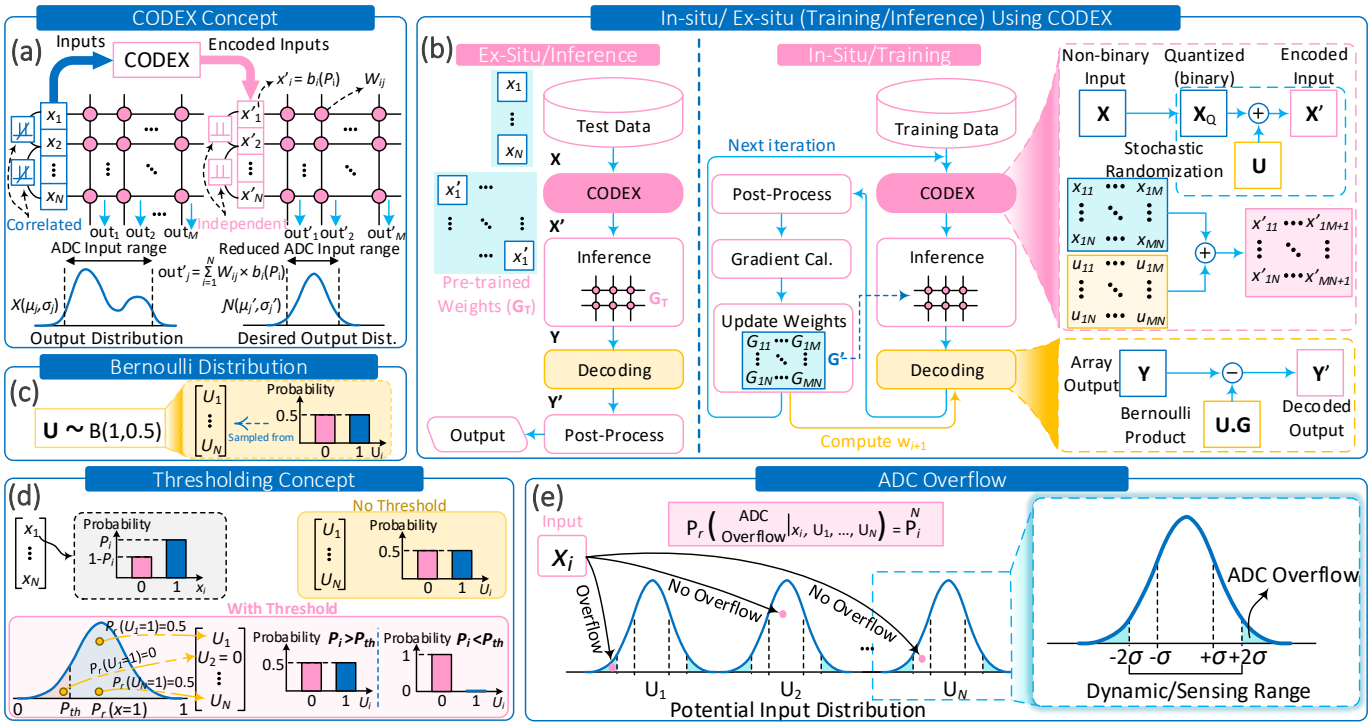

Fig. 2. (a) CODEX encodes inputs randomly to normalize output distribution and reduce ADC input range. (b) CODEX in-situ and ex-situ usage flowchart. (c) Stochastic encoding matrix **U** is sampled from a Bernoulli distribution with an even probability distribution. (d) Threshold inputs by only randomizing non-sparse input bits to reduce average output current. (e) Prepare multiple spare CODEX encoding matrices to minimize ADC overflow probability.

encoding matrices to substitute in the case of an overflow. Let's denote these random matrices as $U^* = \{\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_N\}$. Given an input matrix **X**, the output distribution for the k-th bit is $I_k = \sum_i^N (X_i + b(0.5)) \cdot G_{ij}$. Assuming the probability that I_k falls outside the ADC input range is p_{over}, then the probability of every independent matrix $\mathbf{U}_i \in U^*$ causing an ADC overflow is $(p_{over})^N$ as shown in Fig. 2(e). Since the probability of ADC overflow decreases exponentially with the number of random encoding matrices prepared, this problem can be fully addressed in practice by randomly sampling from a relatively small pool of ten encoding matrices. Even with a narrow ADC range of 2σ, the probability of overflow with all ten encoding matrices is negligible ($3.8 \times 10^{-12}\%$).

IV. Results and Discussion

A. CODEX Application and Performance Analysis

Fig. 3(a) shows CODEX is applied to a multi-layered neural network. Every layer in the network has a randomly assigned encoding matrix $\mathbf{U}_i$ which is used to encode the input to the layer ($\mathbf{X'}_i = \mathbf{X}_i + \mathbf{U}_i$) and decode the subsequent output of the layer ($\mathbf{Y'}_i = \mathbf{Y}_i - \mathbf{U}_i\mathbf{G}_i$). Fig. 3(b) illustrates the two main ways that crossbar systems can benefit from a reduction in ADC input range: decrease ADC design requirements which can reduce ADC area and power consumption or increase ADC resolution by reducing the space between subsequent quantization levels. By increasing ADC resolution, CODEX provides improvements in training speed during in-situ/on-chip neural network training. Fig. 3(c) shows an accuracy curve for a simple 2-layer neural network training on the MNIST digit classification task [17]. We can observe that CODEX reaches an acceptable accuracy of 90% after 96000 training images which is 30.2% faster than the baseline model. CODEX has two main parameters that need to be tuned before being applied to a crossbar system. First, we must consider the probability threshold when dealing with sparse input bits. Fig. 3(d) highlights the probability threshold's effect on the output current distribution for the previously described 2-layer neural network on the MNIST dataset. Increasing the probability threshold decreases the average output current as intended with the most significant decrease occurring from $th = 0$ to $th = 0.1$ where the average current decreases by 0.63 mA. The second parameter to consider is how narrow should the ADC input range be set to. Fig. 3(e) analyzes the trade-off between decreasing ADC input range and the induced time overhead due to dealing with ADC overflow. CODEX results a semi-normal output distribution which aligns with the exponentially increasing time overhead when decreasing ADC range as seen in Fig. 3(e). As such, it is most efficient to set the ADC range such that less than 5% (2σ) of expected outputs result in ADC overflow. CODEX is also affected by the specific applications occurring on the crossbar and the properties of the crossbar system. Specifically, CODEX can cause extra error in a crossbar VMM operation because of the additional decoding and encoding operations involved in CODEX. As shown in Fig. 3(f), neural network depth and width can heavily influence the amount of induced error caused by CODEX. From Fig. 3(f), we can see that computational error is only unacceptably high when a neural network has many (≥ 6) layers with over 300 hidden neurons each. However, neural networks with over 6 wide consecutive fully-connected layers are almost never used in practice because of the existence of more parameter-efficient architectures such as deep CNN's like ResNet. Similarly, ADC resolution can also impact the amount of induced error caused by CODEX as analyzed in Fig. 3(g). We observe that the mean square error (MSE) with CODEX

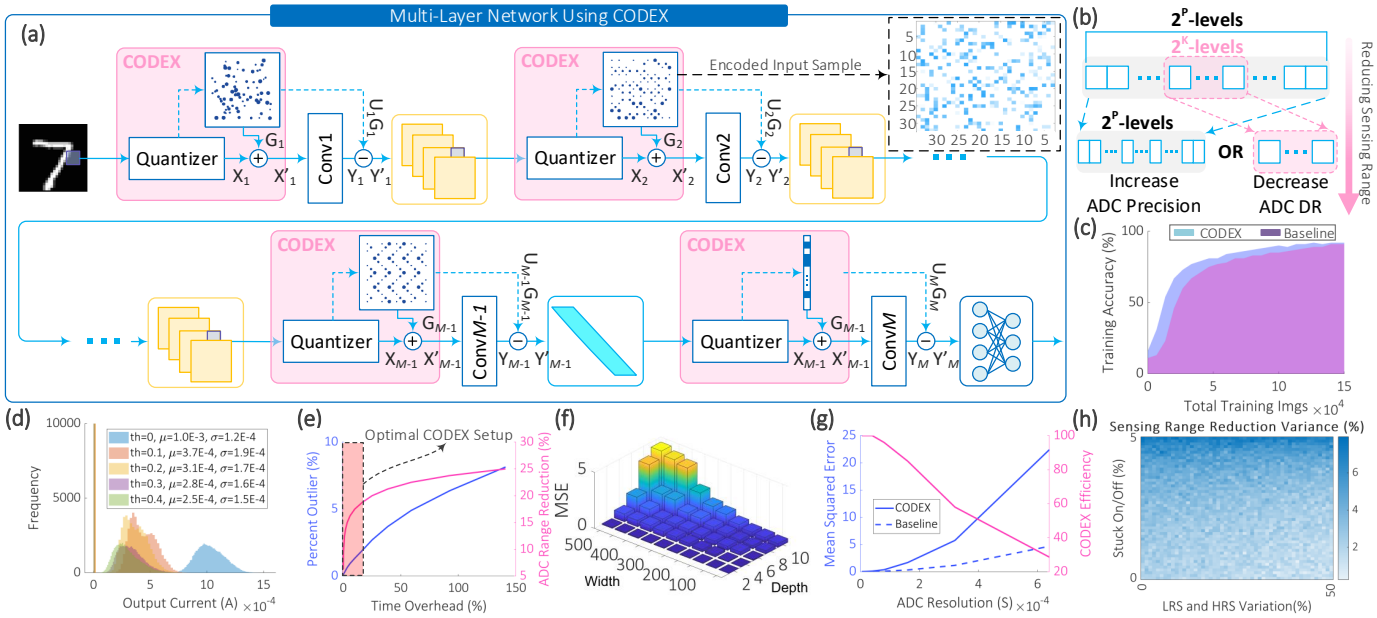

Fig. 3. (a) CODEX is applied sequentially to the inputs of each layer of a DNN. (b) A reduced output input range can be used to increase ADC resolution or decrease ADC design requirements. (c) In-situ training accuracy curves. (d) CODEX output current distributions with varying probability thresholds. (e) Analysis of time overhead with varying cutoffs of ADC input range. (f) 3D bar chart of CODEX induced error with varying neural network width and depth. (g) Variance in MSE and CODEX effectiveness with respect to ADC resolution. (h) Impact of memristor stuck on/off percentage and low/high resistive state variation on ADC input range reduction.

grows much faster with decreasing ADC resolution than the baseline. CODEX rapid error growth occurs because the ADC sensing error, introduced in the encoding and decoding modules of CODEX, scales proportionally with ADC resolution. It is a strength that CODEX effectiveness is maximized at high ADC bit resolution because high-resolution ADCs compose significantly higher proportions of memristive VMM platform's power consumption. Memristor crossbar non-idealities can change the output current distribution which can impact CODEX effectiveness. Fig. 3(h) illustrates how memristor low (LRS) and high (HRS) resistive state variation and stuck-on/stuck-off memristors can cause upwards of 8% increased variance in the CODEX's ADC input range reduction. Other non-idealities were also investigated in this research, but they had negligible impact on CODEX input range reduction.

B. Results

In this paper, all simulations are performed using our extended 1T1R memristor crossbar simulation model [18] that includes the following memristor non-idealities: limited memristor programming precision, high and low conductance state noise for device-to-device variation, stuck-on and stuck-off memristors, and line resistances. To evaluate CODEX performance on existing crossbar systems, we apply it to the state-of-the-art ISAAC [19] and PUMA [20] accelerators. While our test results consider ISAAC and PUMA as baseline architectures, CODEX can be applied to enhance any general memristor crossbar system. CODEX is tested on three datasets across four different applications: CIFAR-10 [21] for image classification and reconstruction, sentiment classification dataset [22], and Freesound dataset for audio classification [23]. Fig. 4(a-c,k) show the output current histograms across the four tasks for CODEX and the baseline system. While the baseline output distribution varies across the different tasks, the CODEX output distribution maintains a relatively consistent truncated normal distribution due to the ADC input range cutoff. On average, CODEX caused a 3.3× reduction in the range of output current CODEX with a max and minimum reduction of 2.37× and 4.16× for the sentiment analysis and CIFAR image reconstruction tasks, respectively. Fig. 4(d-f,j) examines the case where CODEX is used to speed up in-situ training by increasing ADC resolution. After 100000 training images, the baseline model reaches an average validation accuracy 81.3% across the classification tasks. CODEX surpasses the final accuracy of the baseline model with 25.8% less training images on average and was most effective for the CIFAR classification task where CODEX reaches the final baseline accuracy 33.6% faster after only 66400 training images. The second part of these sub-figures show the improvement (gain) in CODEX's test outputs for each application as compared to the baseline model's outputs. In Fig. 4(j), the CODEX and baseline loss curves follow roughly with CODEX having roughly 0.01 less BCE error at all times. A look at the sample test outputs shown shows that CODEX consistently reconstructs noticeably clearer images than the baseline model. CODEX's output range reduction allows for the implementation of ADCs with fewer ENOBs onto these baseline crossbar systems with negligible reduction in system performance. Fig. 4(g-i,l) shows the benefits to power, area, and SNDR when CODEX is used to reduce ADC design requirements. When comparing energy (EE), storage (SE) and computational (CE) efficiency, we keep our definition of an operation consistent with [20]. With reduced ADC design requirements, CODEX can be optimized to decrease ADC power (CODEX-P) or ADC area (CODEX-A). In both cases, the max SNDR is reduced because it is directly a function of ADC ENOB. From Table 1, CODEX provides a 50.0% (PUMA-C) and 63.1% (ISAAC-C) increase in EE over the two baseline accelerators. CODEX provides a smaller 8.62% and

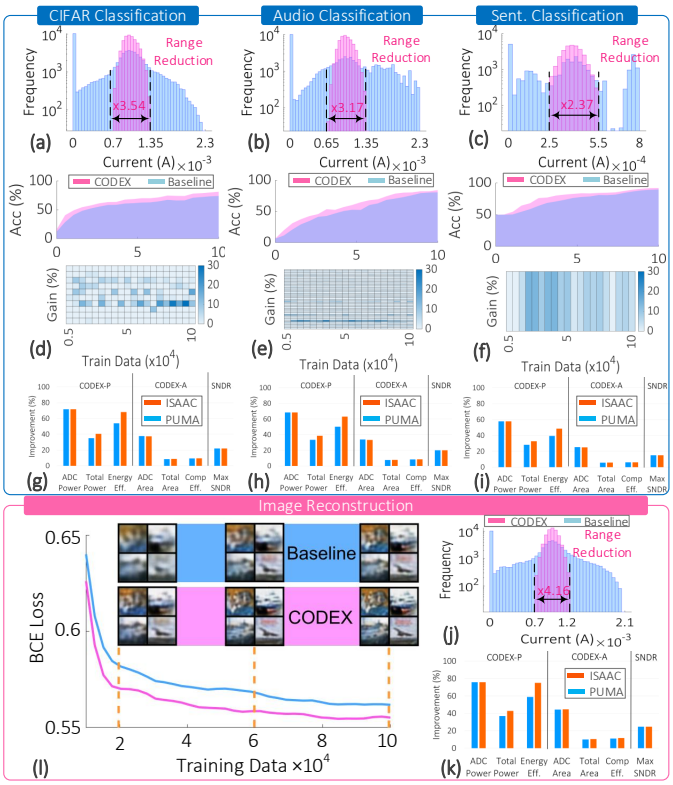

Fig. 4. (a-c,k) Baseline vs CODEX output histograms on CIFAR image classification (a), audio classification (b), sentiment classification (c), and CIFAR image reconstruction (k). (d-f) In-situ sample output predictions and training accuracy across tasks (a-c). (g-i,l) Improvements in energy metrics, area metrics, and max Signal to noise distortion ratio (SNDR) over baseline implementations [19], [20]. (j) In-situ cross entropy loss curve with sample output images.

TABLE I
COMPARISON OF CODEX WITH PREVIOUS WORKS.

Methods	PUMA	PUMA-C	ISAAC	ISAAC-C
ADC Power (W)	35.3	11.1	32.3	**10.2**
ADC Area (mm^2)	21.2	13.7	19.4	**12.5**
Total Power (W)	62.5	**38.3**	65.8	43.7
Total Area (mm^2)	90.6	83.1	85.4	**78.6**
EE (TOPS/J)	0.84	1.37	1.06	**1.59**
SE (MB/mm^2)	0.76	**0.83**	0.74	0.80
CE ($TOPS/mm^2$)	0.58	0.63	0.81	**0.88**

8.64% improvement in CE over PUMA and ISAAC because there is a smaller reduction in ADC area than power. In addition, ADCs take a smaller proportion of total chip area than chip power in the ISAAC and PUMA architectures. In practice, CODEX can be tuned in between CODEX-P and CODEX-A to provide both benefits in ADC area and power at once depending on the needs of the system and application.

V. CONCLUSION

In this paper, we propose a stochastic input encoding method called CODEX to reduce the ADC input range in memristor crossbar systems. By taking advantage of Bernoulli statistics to normalize the output range, CODEX can be applied to any general crossbar system and is shown to be effective across a wide range of applications from sentiment classification to image reconstruction. When applied to the state-of-the-art ISAAC and PUMA accelerators, CODEX decreases in-situ training time on average by 25.8% and provide an average 56.6% and 8.63% increase in EE and CE, respectively.

REFERENCES

[1] P. Yao, et al., "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641646, 2020.
[2] C. Li, et al., "Long short-term memory networks in memristor crossbar arrays." Nature Machine Intelligence, vol. 1, no. 1 pp. 49-57, 2019.
[3] Z. Wang, et al. "Reinforcement learning with analogue memristor arrays." Nature electronics, vol. 2, no. 3, pp. 115-124, 2019.
[4] Z. Wang, et al. "In situ training of feed-forward and recurrent convolutional memristor networks." Nature Machine Intelligence, vol. 1, no. 9, pp. 434-442, 2019.
[5] F. Cai, et al. "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations." Nature Electronics, vol. 2, no. 7, pp. 290-299, 2019.
[6] A. Mehonic, et al. "Memristors—From In-Memory Computing, Deep Learning Acceleration, and Spiking Neural Networks to the Future of Neuromorphic and Bio-Inspired Computing." Advanced Intelligent Systems, vol. 2, no. 11, pp. 2000085, 2020.
[7] A. Serb, et al. "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses." Nature communications, vol. 7, no. 1, pp. 1-9, 2019.
[8] H. Huang, et al. "Implementation of Dropout Neuronal Units Based on Stochastic Memristive Devices in Neural Networks with High Classification Accuracy." Advanced Science, vol. 7, no. 18, pp. 2001842, 2020.
[9] M. Rahimi Azghadi, et al., "Complementary Metal-Oxide Semiconductor and Memristive Hardware for Neuromorphic Computing," Advanced Intelligent Systems, Vol. 2, no. 5, pp. 1900189, 2020.
[10] A. Sebastian, et al. "Memory devices and applications for in-memory computing." Nature nanotechnology, vol. 15, no. 7 pp. 529-544, 2020.
[11] A. Amirsoleimani, et al., "In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives," Advanced Intelligent Systems, Vol. 2, no. 11, pp. 2000115, 2020.
[12] I. Chakraborty, et al., "Resistive crossbars as approximate hardware building blocks for machine learning: Opportunities and challenges," Proceedings of the IEEE, Vol. 108, no. 12, pp. 2276-2310, 2020.
[13] P. Narayanan, et al., "Toward on-chip acceleration of the backpropagation algorithm using nonvolatile memory," IBM J. Res. Dev., Vol. 61, nos. 4–5, pp. 1–11, 2017.
[14] W. Wei et al., "A relaxed quantization training method for hardware limitations of RRAM-based computing-in-memory," IEEE J. on Exp. Solid-State Comp. Dev. and Circ., Vol. 6, pp. 56-52, 2020.
[15] R. Genov and G. Cauwenberghs, "Stochastic mixed-signal VLSI architecture for high-dimensional kernel machines," Adv. in Neural Info. Proc. Sys., pp. 1099-1105, 2001.
[16] R. Karakiewicz, et al., "1.1 TMACS/mW Fine-grained stochastic resonant charge-recycling array processor," IEEE Sensors Journal, vol. 12, no. 4, pp. 785–792, 2011.
[17] Y. LeCun, et al., "Gradient-based learning applied to document recognition," in Proc. IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
[18] T. Liu, et al., "AIDX: Adaptive Inference Scheme to Mitigate State-Drift in Memristive VMM Accelerators," in IEEE Transactions on Circuits and Systems II: Express Briefs, doi: 10.1109/TCSII.2020.3026642, 2020.
[19] A. Shafiee, et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA), pp. 14–26, 2016.
[20] A. Ankit, et al., "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in Proc. of the 24th Int. Conf. on Arch. Sup. for Prog. Lang. and Op. Sys., pp. 715-731, 2019.
[21] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
[22] B. Pang and L. Lee, "A sentimental education," in Proc. of the 42nd Annu. Meeting on Assoc. for Comp. Ling., 2004.
[23] E. Fonseca, et al., "Freesound datasets: A platform for the creation of open audio datasets," in Proc. of the 18th Int. Soc. for Music Inf. Retr. Conf., 2017.