A Neural Network Accelerator to Avoid Inference Inaccuracy

Amritanand Sebastian
Pennsylvania State University https://orcid.org/0000-0003-4558-0013
Saptarshi Das (sud70@psu.edu)
Pennsylvania State University https://orcid.org/0000-0002-0188-945X

Article

Keywords:

Posted Date: February 4th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1322588/v1

Version of Record: A version of this preprint was published at Nature Communications on October 17th, 2022. See the published version at https://doi.org/10.1038/s41467-022-33699-7.

A Neural Network Accelerator to Avoid Inference Inaccuracy

Amritanand Sebastian[1,] & Saptarshi Das[1,2,3,4,*]*

[1]Deparment of Engineering Science and Mechanics, Penn State University, University Park, PA 16802

[2]Department of Materials Science and Engineering, Penn State University, University Park, PA 16802

[3]Department of Electrical Engineering and Computer Science, Penn State University, University Park, PA 16802

[4]Materials Research Institute, Pennsylvania State University, University Park, PA 16802

Abstract : Artificial neural networks (ANNs) have demonstrated their superiority over traditional computing architectures in tasks such as pattern classification and learning. While ANNs demonstrate high prediction accuracy, they do not measure uncertainty in predictions, and hence they can make wrong predictions with high confidence, which can be detrimental for many mission-critical applications. In contrast, Bayesian neural networks (BNNs) naturally include such uncertainty in their model. Unlike ANNs, where the synaptic weights are point estimates (single-valued), in BNNs, the weights are represented by probability distributions (e.g. Gaussian distribution). This makes the hardware implementation of BNNs challenging since on-chip Gaussian random number generators (GRNGs) based on silicon complementary metal oxide semiconductor (CMOS) technology are area and energy inefficient. Stochastic switching in memristors can be used to build probabilistic synapses but with very limited tunability. Additionally, memristor technology rely heavily on CMOS-based peripherals to emulate neurons. Here we introduce three-terminal memtransistors based on two-dimensional (2D) materials, which can emulate both probabilistic synapses as well as reconfigurable neurons. The cycle-to-cycle variation in the programming of the 2D memtransistor is exploited to achieve GRNG-based synapses,

whereas 2D memtransistor based integrated circuits are used to obtain neurons with hyperbolic tangent and sigmoid activation functions. Finally, memtransistor-based synapses and neurons are combined in a crossbar array architecture to realize a BNN accelerator and the performance is evaluated using the IRIS data classification task.

Introduction

Machine learning has seen unprecedented growth and success in the recent years owing to the development of artificial neural networks (ANNs). By mimicking the biological neural architecture and employing deep learning algorithms, ANNs have demonstrated notable advantages over standard computing methods for tasks such as image classification, facial recognition, data mining, weather forecasting, and stock market prediction [1-5]. While ANNs offer high performance, especially in terms of high prediction accuracy, they often suffer from overfitting due to lack of generalization as they do not model uncertainty. Large datasets and various regularization techniques are often required to reduce overfitting in ANNs [6]. However, this can limit the use of ANN in applications where the data is scarce. Additionally, uncertainty estimation is important in applications like autonomous driving, and medical diagnosis, where machine learning must be complemented with uncertainty-aware models or human intervention [7, 8]. The integration of probabilistic computing paradigms with ANNs allows regularization and enables us to model uncertainty in predictions [9-12]. This is achieved in Bayesian neural networks (BNNs) by incorporating Bayes theorem to the traditional neural network scheme [12, 13]. BNNs are capable of modelling uncertainty and avoiding overfitting, while working well with small datasets [14]. In fact, BNNs are extremely powerful as they represent an *ensemble* model, which is equivalent to the combinations of numerous ANNs, but with a small number of parameters.

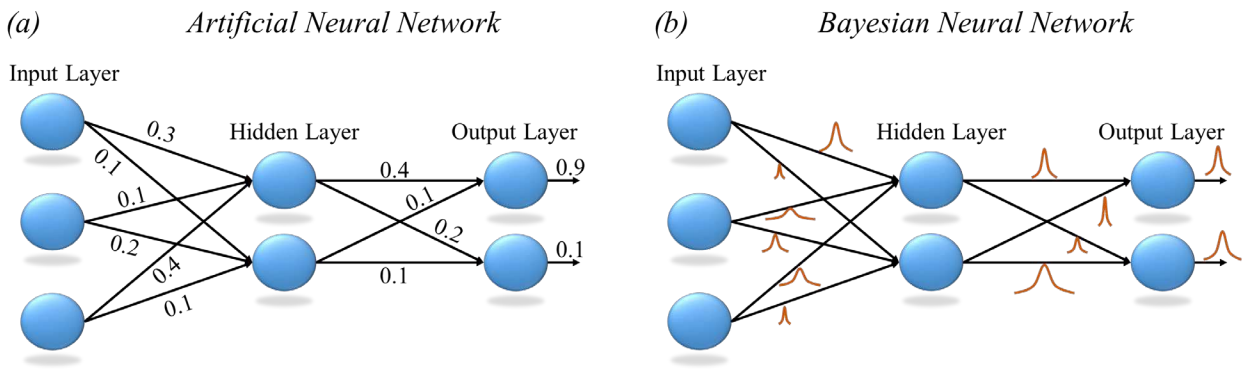

Figure 1. Comparison of an artificial neural network (ANN) and a Bayesian neural network (BNN)*. Schematic of a) an ANN and b) a BNN. The synapses of ANN are represented by single-valued weights while the synapses of a BNN is represented by probability distributions.*

Unlike ANNs, where the synaptic weights are point estimates (single-valued), in BNNs, the weights (W) are represented by probability distributions, as shown in Fig. 1. According to the Bayes theorem, these weights are given by posterior probability distributions given by Eq 1.

$$P(W|D) = \frac{P(D|W).P(W)}{P(D)} \quad [1]$$

Here, D is the training data, $P(W|D)$ is the posterior distribution, $P(D|W)$ is the likelihood, $P(W)$ is the prior, and $P(D)$ is the evidence. The true posterior distribution is untraceable in BNNs and hence, methods such as variational inference [12] and Markov chain Monte Carlo (MCMC) sampling [15] are used to approximate the posterior distribution. Variational inference is typically preferred due to better convergence and scalability compared to MCMC [16]. In the variational inference method, $P(D|W)$ is estimated using a family of variational posterior distributions (typically Gaussian distributions), $q(W;\theta)$, where θ represents the variation parameters. For a Gaussian distribution, the variation parameters are its mean and standard deviation i.e., $\theta = \mu, \sigma$. The estimation is performed by minimizing the Kullback-Leibler divergence between $P(D|W)$ and $q(W;\theta)$. In the training phase, for each synapse, μ and σ are learned using the traditional backpropagation method [12]. Here, σ represents the uncertainty introduced by each synapse. To

perform inference using a BNN, multiple forward passes of the trained network is evaluated. During each forward pass, each of the Gaussian weight distributions are sampled once. The output of the network (y) is obtained by averaging the outputs of these forward passes obtained by sampling the weight distributions (Eq. 2). It can be approximated by drawing S Monte Carlo samples and finding its mean given by Eq. 2.

$$y = E_{q(W;\theta)}[g(x|W)] \approx \frac{1}{S}\sum_{i=1}^{S} g(x|W^i) \quad [2]$$

Here, x is the input, g is the transfer function of the neural network and W^i represents the i^{th} Monte Carlo weight sample.

Over the years we have witnessed the development of neural network accelerators aimed at improving the size, energy consumption, and speed of neural networks, especially for edge computing applications [17-19]. Since the training process in neural networks is energy and resource intensive, these works typically rely on off-chip training and on-chip inference. Hence, BNN accelerators have also mostly focused on implementing Bayesian inference on-chip [16, 20-24]. A crucial component of the BNN accelerator is an on-chip Gaussian random number generator (GRNG)-based synapse that can sample weights from a Gaussian distribution. Additionally, a BNN requires a circuit to implement a neuron, i.e., to perform the multiply and accumulate (MAC) operation and the neural activation. BNN implementations based on Si complementary metal oxide semiconductor (CMOS) and field-programmable gate array (FPGA) typically require elaborate hardware for GRNGs, MAC operation, and the activation function, rendering them area and energy inefficient [16, 20, 21]. Moreover, these demonstrations are based on the von-Neumann architecture with separate memory and logic units, requiring frequent shuttling of data between

the two. BNN accelerators based on emerging and non-von Neumann memristive and spintronic synapses utilize cycle-to-cycle variability in switching to generate Gaussian random numbers (GRNs) [22-24]. However, these GRNG-based synapses are limited to μ=0 and σ=1 and require extensive CMOS-based peripherals circuitry to obtain unrestricted μ and σ values. For example, multiplication and addition operations are used to transform $N(0,1)$ to $N(\mu,\sigma) = \sigma * N(0,1) + \mu$. Finally, two-terminal memristors also lack the capability to emulate neurons for the activation functions. Therefore, energy and area efficient acceleration of BNNs will benefit from a standalone hardware platform, which can offer both neurosynaptic functionalities as well as programmable stochasticity.

In this work, we introduce three-terminal memtransistor technology based on two-dimensional (2D) monolayer MoS_2 offering all computational primitives needed for a BNN accelerator. First, we realize an ultra-low power GRNG-based synapse by exploiting the cycle-to-cycle variability in programming/erasing operation in the 2D memtransistor. Next, using a circuit comprising of two memtransistors we achieve reconfigurable μ and σ. Activation functions such as hyperbolic tangent (tanh) and sigmoid are also realized using the 2D memtransistor-based circuits. Finally, we demonstrate a crossbar array architecture in order to implement on-chip BNN inference. Furthermore, the entire network is simulated using LTSpice.

2D memtransistor

The schematic of a 2D memtransistor is shown in Fig. 2a (***Supplementary Fig. 1a*** shows the optical image). This 2D memtransistor has a local back-gated geometry, where, atomic layer deposition grown 50 nm Al_2O_3 is used as the gate dielectric and TiN/Pt is used as the local gate

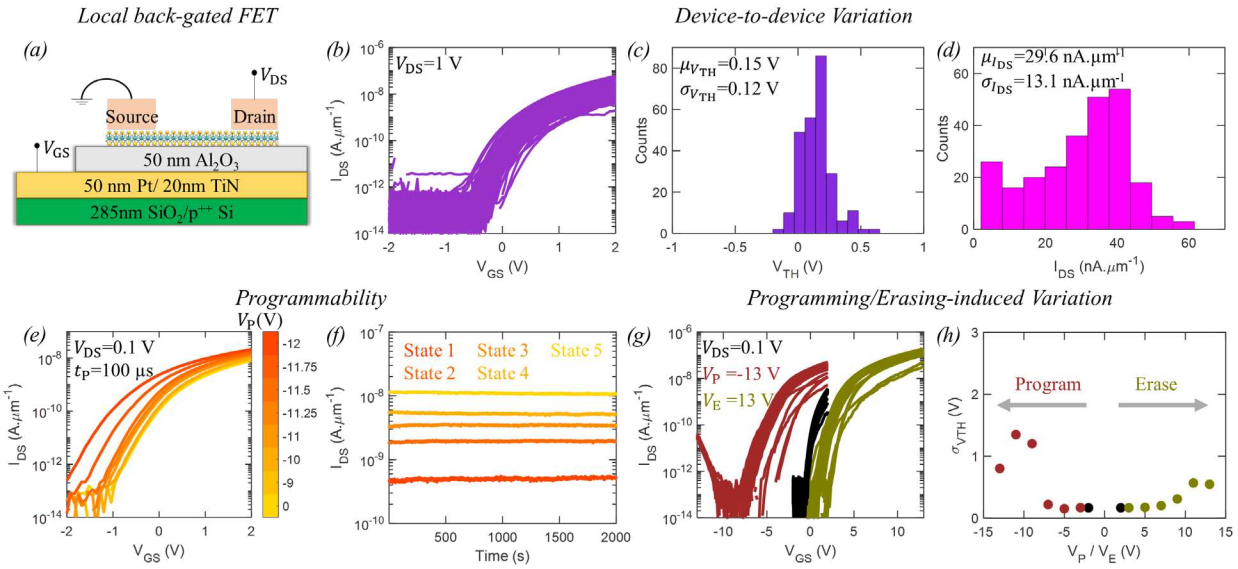

Figure 2. Programmable MoS_2 memtransistors*. a) Schematic of a MoS_2 memtransistors with a local back-gated geometry, with a channel length and channel width of 5 μm. b) Device-to-device variation in the transfer characteristics, i.e., drain current (I_{DS}) versus gate-to-source voltage (V_{GS}) for drain-to-source voltage (V_{DS}) of 1 V for 250 MoS_2 memtransistors. Distribution of c) threshold voltage (V_{TH}) and d) I_{DS} extracted from 250 MoS_2 memtransistors. The corresponding means (μ) and standard deviations (σ) are denoted in the inset. MoS_2 memtransistors offers analog and non-volatile memory, where V_{TH} of the memtransistors can be adjusted by applying a programming pulse to the back gate. d) Transfer characteristics of a post-programmed MoS_2 memtransistor obtained by applying negative programming voltages (V_P) of different amplitudes for pulse duration (t_P) of 100 μs. f) Analog retention characteristics, i.e., post-programmed I_{DS} versus time measured at V_{GS} = 0 V, for 5 different states. g) The impact of programming/erasing on device-to-device variation, demonstrated using different V_{GS} sweep ranges. h) $\sigma_{V_{TH}}$ as a function of V_P and erasing voltage (V_E) applied during V_{GS} sweeps. An increase in variation is seen for high V_P/V_E.*

electrode (see ***Methods*** section for details on fabrication). This geometry (similar to the top-gated geometry) enables independent modulation of each memtransistor and the development of circuits necessary for a BNN. Note that we have used monolayer MoS_2 grown using metal organic chemical vapor deposition (MOCVD) described in our previous reports [25, 26]. The choice of MoS_2 as the element of memtransistor is motivated by recent demonstrations highlighting the technological viability of 2D materials [27-29] and their wide scale adoption in brain-inspired computing [30-34].

The transfer characteristics, i.e., drain current (I_{DS}) *versus* gate-to-source voltage (V_{GS}) for different drain-to-source voltage (V_{DS}) of 1 V for 250 MoS_2 memtransistor are shown in Fig. 2b.

Fig. 2c and Fig. 2d shows the distributions of threshold voltage (V_{TH}) and I_{DS}, respectively, extracted from the transfer characteristics. Here, I_{DS} is extracted at V_{GS} = 2 V and V_{DS} = 1 V, and V_{TH} is extracted using the constant-current method, at 0.1 nA.μm^{-1}. The device-to-device variation is seen to be low, with variation in V_{TH} ($\sigma_{V_{TH}}$) of 0.15 V.

The MoS_2 memtransistor offers analog and non-volatile charge-trap memory. The MoS_2 memtransistor can be programmed i.e., the threshold voltage (conductance) of the device can be decreased (increased) by applying a program pulse to the back-gate with a large negative voltage (V_P). Fig. 2e demonstrates the post-programmed transfer characteristics of a MoS_2 memtransistor for V_{DS} = 0.1 V, measured after programming with different V_P. Similarly, MoS_2 memtransistor can be erased i.e., the threshold voltage (conductance) of the device can be increased (decreased) by applying an erase pulse to the back-gate with a large negative voltage (V_E), as shown in ***Supplementary Fig. 1b***. A pulse duration (t_P) of 100 μs is used for both programming and erasing. The dependence of programming and erasing, on t_P is shown in ***Supplementary Fig. 1c and Fig. 1d,*** respectively. The non-volatile nature of the MoS_2 memtransistor is shown in Fig. 2f, where the retention characteristics for 5 different conductance states are demonstrated for 2000 s. The working principle of this analog and non-volatile memory has been described in detail in our earlier report [32].

Gaussian random number generator-based synapse

BNN accelerators rely on techniques such as cumulative density function inversion, central limit theorem (CLT)-based approximation, and the Wallace method to generate standard GRNs [16, 20, 21]. These methods typically require linear feedback shift registers, multipliers, and adders,

involving numerous transistors to implement the GRNGs, rendering them area and energy inefficient. In contrast, here we use cycle-to-cycle variation in the programmability of our MoS_2 memtransistor to generate GRNs. While cycle-to-cycle variation is undesirable for traditional computing, it can be exploited to reduce the design complexity of a BNN accelerator [22, 23, 35]. To demonstrate the effect of programming variation, we use dynamic programming on 40 MoS_2 memtransistors, where we measure the transfer characteristics with different V_{GS} sweep ranges. To evaluate the effect of V_{P} (V_{E}), the maximum positive (negative) V_{GS} is fixed at +2 V (-2 V), while the maximum negative (positive) V_{GS} is stepped from -3 V to - 13 V (3 V to 13 V). As shown in Fig. 2g, high V_{P} and V_{E} (± 13 V) increases the device-to-device variation (post programming/erasing). High V_{P} and V_{E} (beyond ±7 V) results in significant V_{TH} shift (see ***Supplementary Fig. 2)***, while also increasing $\sigma_{V_{\mathrm{TH}}}$, as shown in Fig. 2h. This increase in device-to-device variation for high V_{P} and V_{E} is also accompanied by an increase in the cycle-to-cycle variation.

To utilize the cycle-to-cycle variation, the gate of a MoS_2 memtransistor is subjected to successive erase-program-read pulse cycles with V_{E} = 13 V, V_{P} = -13 V, and read voltage (V_{R}) of 0 V as shown in Fig. 3a. The corresponding V_{DS} values were 0, 0 and 0.1 V, respectively. The conductance (G) of the memtransistor, measured at each read step, is shown in Fig. 3b for 200 cycles. As evident from the histogram shown in Fig. 3c, G follows a Gaussian distribution, with mean, $\mu_G = 3.5$ nS and, standard deviation, $\sigma_G = 0.9$ nS. The quantile-quantile (Q-Q) plot of G further confirms the Gaussian distribution. The quantiles of G (represented using circles) are plotted against the theoretical quantiles from a Gaussian distribution as shown in Fig. 3d. As expected, it closely follows a straight line. Note that the slope of the Q-Q plot represents σ_G and G corresponding to

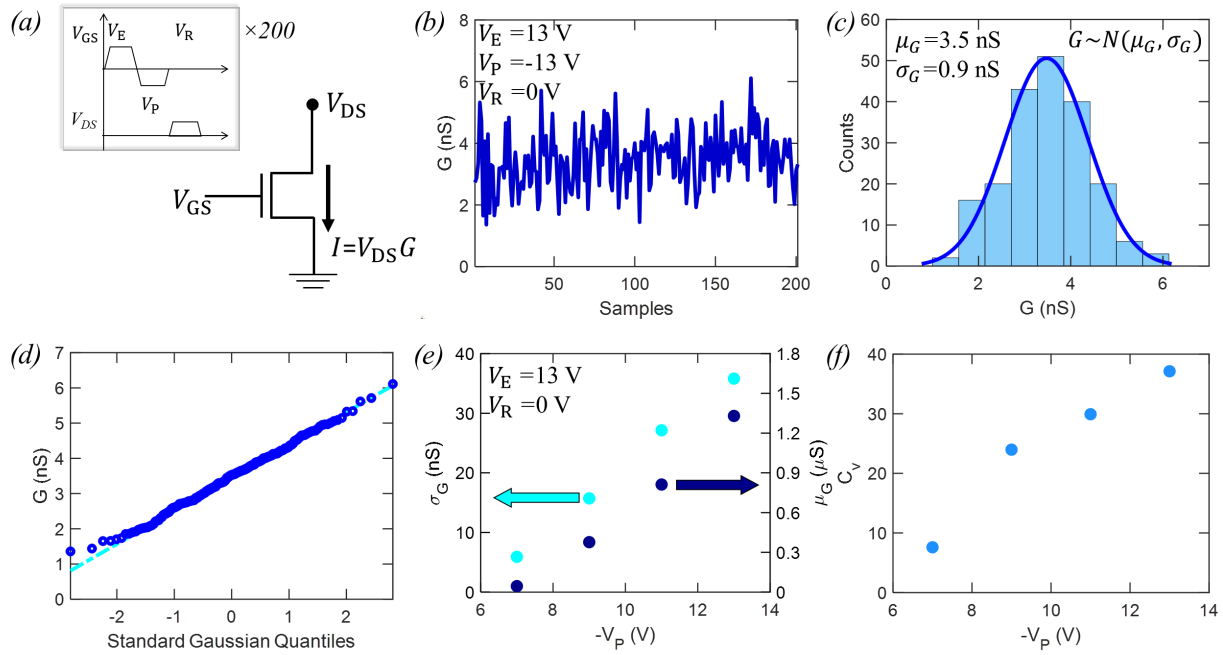

***Figure 3. Gaussian random number generator (GRNG) using MoS_2 memtransistor**. a) Schematic of a MoS_2 memtransistor used as a GRNG. To generate gaussian random numbers (GRNs), the gate of a MoS_2 memtransistor is subjected to an erase-program-read pulse cycle 200 times with V_E of 13 V, V_P of -13 V, and read voltage (V_R) of 0 V, while the drain is subjected to V_{in} of 0, 0 and 0.1 V, respectively. b) The corresponding conductance (G) of the MoS_2 memtransistor measured at each read step, demonstrating random fluctuations in G. c) Histogram demonstrating that G follows a Gaussian distribution, with μ_G = 3.5 nS and σ_G = 0.9 nS. d) The quantile-quantile (Q-Q) plot of G, confirming its Gaussian distribution. The quantiles of G (represented using circles) are plotted against the theoretical quantiles from a Gaussian distribution, which follows the straight line expected for a Gaussian distribution. e) Dependence of μ_G and σ_G and the corresponding f) coefficient of variation (C_v) on V_P. Here, V_P is changed in the erase-program-read pulse cycle. μ_G and σ_G are seen to be coupled.*

quantile 0 represents μ_G. Further characterization of 2D memtransitor-based GRNG has been done in our previous report [36]. Nevertheless, MoS_2 memtransistors are used to generate a physical random variable that samples analog conductance values from a Gaussian distribution i.e., $G\sim N(\mu_G,\sigma_G)$. Moreover, the MoS_2 memtransistor can be used as a synapse, which scales the input by its synaptic weight. If input is applied as voltage to the drain terminal of the memtransistor, the output current is scaled by G, i.e., $I_{DS}= G.V_{DS}$, as shown in Fig. 3a. Therefore, by combining the cycle-to-cycle variation in G with the synaptic functionality of the memtransistor, we are able to realize a GRNG-based synapse.

Note that to implement a BNN accelerator, it's important to tune both μ_G and σ_G of the GRNG-based synapse independently. μ_G and σ_G can be tuned by modulating V_{P} in the erase-program-read pulse cycle, as shown in Fig. 3e. However, μ_G and σ_G are found to be coupled, and the coefficient of variation ($C_v = \mu_G/\sigma_G$) depends on V_{P}, as shown in Fig. 3f. A similar trend is seen in μ_G, σ_G and C_v as a function of V_{E}, as shown in ***Supplementary Fig. 3***. This dependence of μ_G, σ_G and C_v on V_{P} and V_{E} is demonstrated across multiple memtransistors in ***Supplementary Fig. 4***.

Fig. 4a shows the design of our GRNG-based synapse with independent control over its μ and σ, using two MoS_2 memtransistors, T_+ and T_. While prior demonstrations rely on additional mathematical manipulations of the generated GRNs to establish control over their μ and σ, we are able to achieve it without any additional manipulations or circuitry [22-24, 35]. It is common practice in neural network accelerators to use two devices per synapse in order to map both positive and negative weights [37]. Here, the input to the synapse, V_{in} is applied as $+V_{\mathrm{in}}$ and $-V_{\mathrm{in}}$ to T_+ and T_, respectively, as shown in Fig. 4a. The current at the output node (I_{out}) is then given by sum of currents through T_+ and T_ i.e., I_{T_+} and I_{T_-}, according to the Kirchhoff's current law (KCL) given by Eq. 3.

$$I_{\mathrm{out}} = I_{T_+} + I_{T_-} = G_+.V_{\mathrm{in}} - G_-.V_{\mathrm{in}} = (G_+ - G_-).V_{\mathrm{in}} = G_{\mathrm{eff}}.V_{\mathrm{in}} \quad [3]$$

Here G_+ and G_- are the conductance of T_+ and T_ respectively and G_{eff} is the effective conductance of the synapse. While the conductance of a device is always positive, by modulating G_+ and G_-, using V_{G_+} and V_{G_-} (by applying different V_P), we can obtain both positive and negative G_{eff}. Here, we use V_{P} to modulate G_+ and G_- as V_{P} shows better linearity and lower device-to-device variation compared to V_{E} in GRN generation (see ***Supplementary Fig. 4***). To control $\mu_{G_{\mathrm{eff}}}$ and $\sigma_{G_{\mathrm{eff}}}$, T_+ is subjected to successive erase-program-read pulse cycles, while T_ is programmed to a given state

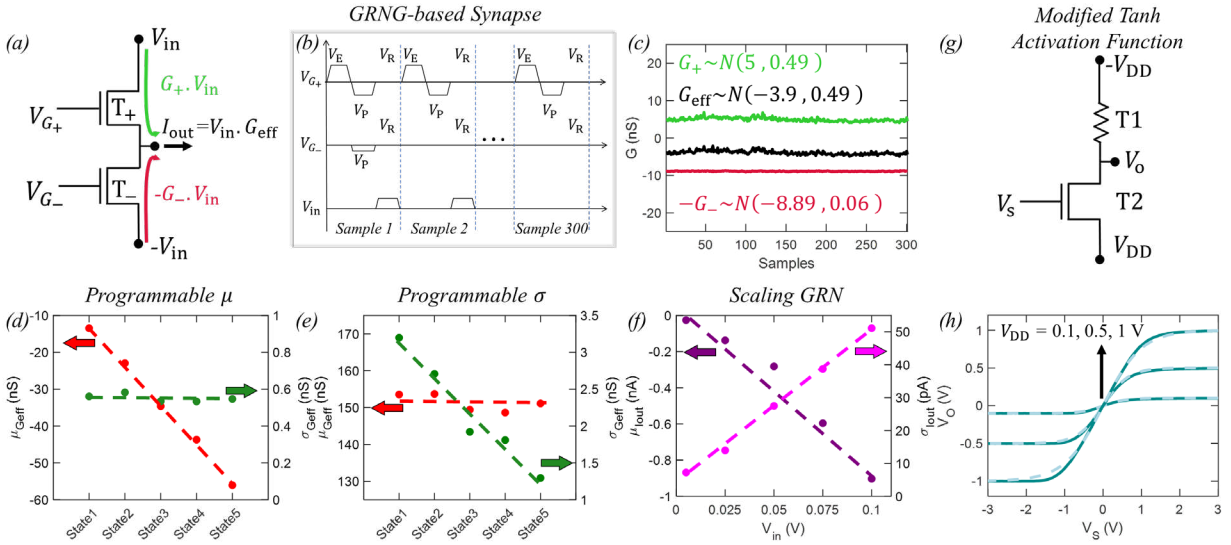

Figure 4. GRNG-based synapse and modified tanh activation function. *a) Schematic of the GRNG-based synapse. The input to the synapse, V_{in} is applied as $+V_{in}$ and $-V_{in}$ to the memtransistors, T_+ and T_- with conductance G_+ and G_- (modulated using V_{G_+} and V_{G_-}), respectively. The effective conductance of this synapse is given by $G_{eff} = G_+ - G_-$, allowing positive and negative conductance. b) Waveform applied to the synapse to generate GRNs with independent control over μ and σ. c) G_+, $-G_-$ and G_{eff} of the synapse sampled 300 times, where T_+ controls σ and T_- controls μ. GRNG-based synapse showing d) independent control of $\mu_{G_{eff}}$ for constant $\sigma_{G_{eff}}$ and e) independent control of $\sigma_{G_{eff}}$ for constant $\mu_{G_{eff}}$. f) Linear scaling of the synaptic output (I_{out}) distribution as the function of V_{in}. g) Schematic of circuit for the modified tanh activation function using two MoS_2 memtransistors ($T1$ and $T2$), where the input voltage (V_S) is applied to $T2$ and $T1$ acts as a resistive load. h) The transfer characteristics of the circuit (solid line) i.e., output voltage (V_O) versus V_S, which closely models the tanh activation function (dotted line).*

and subsequently only read, using the waveforms shown in Fig. 4b. This results in G_+ being drawn from a Gaussian distribution, with μ_{G_+}= 5 nS and σ_{G_+}= 0.49 nS i.e., $G_+ \sim N(5, 0.49)$ nS and G_- having a constant value of ≈ 8.89 nS, as shown in Fig. 4b. G_{eff} is expected to be drawn from a distribution with $\sigma_{G_{\text{eff}}} = \sigma_{G_+}$ and $\mu_{G_{\text{eff}}} = \mu_{G_+} - G_-$. This is confirmed by our measurements as shown in Fig. 4c, $G_{\text{eff}} \sim N(-3.9, 0.49)$ nS. Note that, G_- is not perfectly constant due to the presence of random telegraph fluctuations. However, the fluctuations were found to have a standard deviation of 0.06 nS, making its contribution negligible. The histograms and Q-Q plots of G_+, G_-, and G_{eff} are shown in ***Supplementary Fig. 5***. Fig. 4d shows the independent control of $\mu_{G_{\text{eff}}}$ for constant $\sigma_{G_{\text{eff}}}$ using the GRNG-based synapse. Here, T_+ is subjected to the same erase-program-read cycle, to obtain constant $\sigma_{G_{\text{eff}}}$, whereas T_- is programmed to different states (using V_P) to tune $\mu_{G_{\text{eff}}}$. Fig.

4e shows the independent control of $\sigma_{G_{\text{eff}}}$ for constant $\mu_{G_{\text{eff}}}$. In order to modulate $\sigma_{G_{\text{eff}}}$, σ_{G_+} is changed by applying different erase-program-read cycles (different V_{P}) to T_+. Since this leads to an unfavorable change in μ_{G_+}, T_- is reprogrammed to account for the change in μ_{G_+}, to maintain a constant $\mu_{G_{\text{eff}}}$.

In a synapse, the distribution of I_{out} is expected to scale linearly with V_{in}, as given by Eq. 4.

$$I_{\text{out}} = G_{\text{eff}}.V_{\text{in}} = N\left(\mu_{G_{\text{eff}}}, \sigma_{G_{\text{eff}}}\right).V_{\text{in}} = N\left(\mu_{G_{\text{eff}}}.V_{\text{in}}, \sigma_{G_{\text{eff}}}.V_{\text{in}}\right) = N\left(\mu_{I_{\text{out}}}, \sigma_{I_{\text{out}}}\right) \quad [4]$$

This is demonstrated in Fig. 4f, where $\mu_{I_{\text{out}}}$ and $\sigma_{I_{\text{out}}}$ show linear dependence with respect to V_{in}. The output characteristics of a MoS_2 memtransistor is shown in ***Supplementary Fig. 6*** for positive and negative V_{DS}. While the current is highly non-linear and asymmetric for large $\pm V_{\text{DS}}$ values, it is seen to be sufficiently linear and symmetric between $\pm$ 0.1 V. Hence, we limit the maximum V_{in} to 0.1 V. The low V_{in} allows us to operate the synapse with extremely low currents, as shown in Fig. 4f, offering significant energy efficiency. Overall, we demonstrate independent control over $\mu_{G_{\text{eff}}}$ and $\sigma_{G_{\text{eff}}}$ to implement a GRNG-based synapse with just two MoS_2 memtransistors resulting in significant area and energy efficiency.

Neurons with modified hyperbolic tangent activation function

The hardware for activation function in neural accelerators is generally realized using standard CMOS-based analog and digital components, and hence these implementations do not utilize the advantages offered by emerging materials [37]. Moreover, hyperbolic tangent (tanh) and sigmoid functions are highly non-linear, significantly complicating their hardware demonstration [38]. We demonstrate a circuit for a modified tanh (m-tanh) activation function using two MoS_2 memtransistors (T1 and T2) as shown in Fig. 4g. The transfer function of the circuit i.e., output

voltage (V_O) *versus* input voltage (V_S) closely follows the tanh activation function as shown in Fig. 4h. The maximum of the m-tanh activation function is determined by the drain voltage (V_{DD}), and V_{DD} of 1 V results in the ideal tanh activation function. Here, V_S is applied to the gate of T2, while T1 is used as a resistive load. We use the charge-trap memory to program T1 to have the required resistance at zero gate voltage, eliminating the need for a constant gate bias. T2 is programmed to ensure that the m-tanh function passes through the origin. Note that when V_S = -3 V, T2 operates in the off-state, i.e., T1 is more conductive than T2, resulting in $V_O = -V_{DD}$, whereas for V_S = 3 V, T2 operates in the on-state and becomes more conductive than T1, which results in $V_O = V_{DD}$. Note that the m-tanh activation function can also be implemented using complementary *n*-type and *p*-type transistors as demonstrated in our previous report [39]. Additionally, modified sigmoid activation function can be realized by applying 0 V to the drain terminal of T1, as shown in ***Supplementary Fig. 7***.

Crossbar array architecture

The crossbar array architecture is routinely used in neural network accelerators to perform the MAC operation of a neuron. Fig. 5a shows the circuit used to implement a portion of a BNN shown in Fig. 5b, where M=4 input neurons are connected to N=1 output neuron. Each input neuron is multiplied with their corresponding synaptic weight distributions and the resultants are summed at the output neuron (MAC operation). The resultant of MAC operation is passed through the m-tanh activation function, to obtain the output. To implement this on circuit, as shown in Fig. 5a, the conductance distribution of a synapse in the i^{th} row, j^{th} column and k^{th} layer ($G_{ij}^{(k)}$), given by the combination of $G_{ij_+}^{(k)}$ and $G_{ij_-}^{(k)}$ is modulated using $V_{Gj_+}^{(k)}$ and $V_{Gj_-}^{(k)}$ lines. Inputs to i^{th} row and k^{th} layer are applied as voltages ($\pm V_i^{(k)}$). The current through the j^{th} column, due to these synapses is then

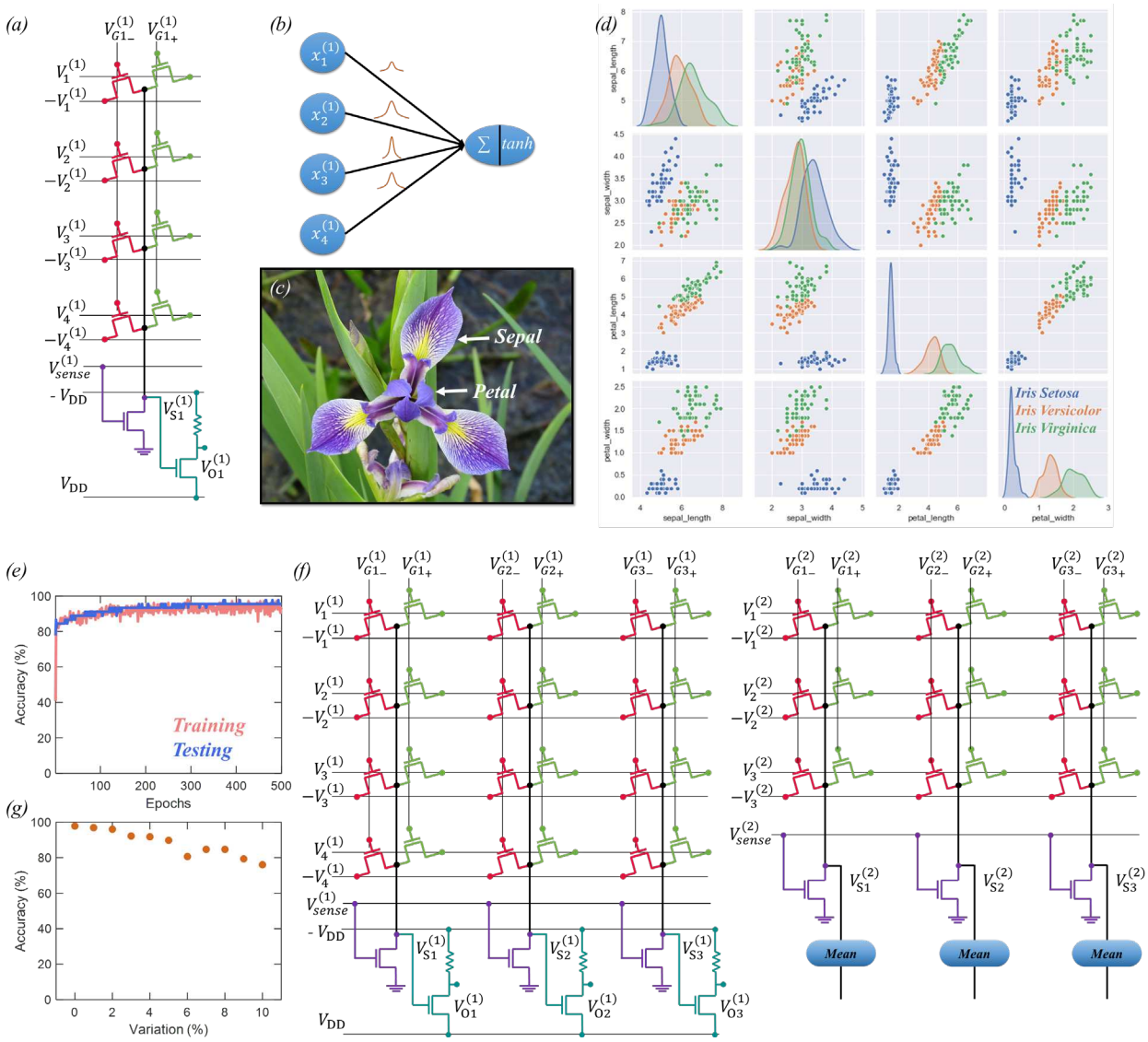

Figure 5. Crossbar array architecture to implement the BNN*. a) Schematic of a portion of a BNN, where multiple input neurons are connected to one output neuron through multiple synapses b) The corresponding circuit implementation using the crossbar array architecture. Here, a sense transistor is used to obtain a voltage ($V_{sj}^{(k)}$) proportional to the dot product between the inputs ($V_i^{(k)}$) and synaptic distributions ($G_{ij}^{(k)}$), where i, j, and k represents the row, column, and layer, respectively. $V_{sj}^{(k)}$ is applied to the tanh activation circuit to obtain the output ($V_{oj}^{(k)}$). Iris dataset consisting of the petal and sepal lengths and widths of c) iris flowers is used to check the performance of the BNN. d) Scatter plot matrix for the iris dataset showing the correlation between different input parameters and their distributions. e) Training and testing curves for 500 epochs of the BNN constructed to classify the iris dataset. f) Circuit implementation of the BNN to perform inference on-chip. g) Test accuracy as a function of variation in synaptic weights averaged over 10 runs.*

given by the dot product of $V_i^{(k)}$ and $G_{ij}^{(k)}$, according to KCL. To obtain a voltage proportional to this dot product, we use a sense transistor, as shown in Fig. 5a. The voltage-drop ($V_{Sj}^{(k)}$) across this sense transistor is given by Eq. 5.

$$V_{Sj}^{(k)} = \frac{\sum_{i=1}^{M} V_i^{(k)} G_{ij}^{(k)}}{G_{Sj}^{(k)} + \sum_{i=1}^{M} G_{ij}^{(k)}} \quad [5]$$

Here, $G_{Sj}^{(k)}$ is the conductance of the sense transistor, modulated using $V_{Sense}^{(k)}$. Using $V_{Sj}^{(k)}$ allows us to seamlessly integrate the circuit for m-tanh activation function into the crossbar array as shown in Fig. 5a, to obtain the corresponding output ($V_{Oj}^{(k)}$). There are some non-idealities which are also accounted for. First, the synaptic weight distribution ($W_{ij}^{(k)}$) is mapped to the crossbar array by using a conductance scaling factor (α) to obtained $G_{ij}^{(k)}$. Second, the denominator of $V_{Sj}^{(k)}$ (Eq. 5) presents a non-ideality, which can be expressed as the product of α and a non-ideality factor ($\gamma^{(k)}$). By mapping the input ($x_i^{(k)}$) to $V_i^{(k)}$, using $\gamma^{(k)}$ as the scaling factor, ideal $V_{Sj}^{(k)}$ and $V_{Oj}^{(k)}$ can be obtained as shown in Eq. 6.

$$V_{Sj}^{(k)} = \frac{\sum_{i=1}^{M} V_i^{(k)} G_{ij}^{(k)}}{G_{Sj}^{(k)} + \sum_{i=1}^{M} G_{ij}^{(k)}} = \frac{\sum_{i=1}^{M} \left(x_i^{(k)} \gamma^{(k)}\right).\left(W_{ij}^{(k)}.\alpha\right)}{\gamma^{(k)}.\alpha} = \sum_{i=1}^{M} x_i^{(k)} W_{ij}^{(k)} \quad [6a]$$

$$V_{Oj}^{(k)} = tanh\left(V_{Sj}^{(k)}\right) = tanh\left(\sum_{i=1}^{M} x_i^{(k)} W_{ij}^{(k)}\right) \quad [6b]$$

$G_{Sj}^{(k)}$ is used to make sure that each column of the crossbar array has the same $\gamma^{(k)}$. With this proposed scheme, we can evaluate the dot product between $x_i^{(k)}$ and $W_{ij}^{(k)}$ in the voltage domain and use the m-tanh activation function to obtain the ideal output, $V_{Oj}^{(k)}$. Note that this scheme is not limited to the implementation of a BNN and can be adopted to implement standard ANN crossbar arrays with tanh and sigmoid activation functions.

Neural network evaluation

We evaluate the performance of our BNN accelerator using the iris data classification task [40]. The iris dataset consists of the lengths and widths of both sepals and petals (shown in Fig. 5c) for three different iris flowers (50 each), namely Setosa, Versicolor, and Virginica. The dataset is represented using a scatterplot matrix as shown in Fig. 5d. To classify this dataset, we use a fully connected 4×3×3 BNN i.e., it has an input layer with 4 neurons, one hidden layer with 3 neurons, and an output layer with 3 neurons. The dataset with 150 instances is divided into 105 for training and 45 for testing. *Bayes by Backprop* algorithm, with a Gaussian prior is used to train the synaptic weight distributions [12, 41]. The BNN is trained off-chip for 500 epochs as shown in Fig. 5e, to obtain train accuracy of 98.1 % and test accuracy of 97.78 %.

Fig. 5f shows the BNN accelerator used to classify the iris dataset. Here, the synapses are arranged in the crossbar array architecture. The weight distributions are mapped to conductance distributions using $V_{Gj_+}^{(k)}$ and $V_{Gj_-}^{(k)}$ with $\alpha = 10^{-9}$. Here, V_{E} of 13 V and V_{R} of 0 V is used, while V_{P} is used to tune $G_{ij_+}^{(k)}$ and $G_{ij_-}^{(k)}$. $\gamma^{(k)}$ is determined for each layer and multiplied with $x_i^{(k)}$ to obtain $V_i^{(k)}$. V_{DD} of 1 V is used for the m-tanh activation circuit. Note that, at the output layer we do not use the tanh activation function. Instead, following Eq. 2, the $V_{\mathrm{S}j}^{(k)}$ is sampled S=100 times to obtain a distribution, and its mean is used to make the classification. Note that the distribution of $V_{\mathrm{S}j}^{(k)}$ at the output layer can be used to calculate the uncertainty in classification [42, 43]. The BNN accelerator in Fig. 5f is evaluated using LTSpice simulations, where we are able to obtain a test accuracy of 93.78 %. Here, we use resistors to implement the synapses. The other components are modeled using NMOS transistors. The dip in accuracy is observed due to the non-symmetric output

of the m-tanh circuit (Fig. 4g). By implementing the tanh activation function using complementary *n*-type and *p*-type transistors [39], the test accuracy of 97.78 % can be replicated. It is important to evaluate the effect of device-to-device variation on the performance of the BNN. Fig. 5g shows the effect of device-to-device variation on the testing accuracy. Here, the BNN is simulated with a variation of up to 10 % in the synaptic weights and the testing accuracy is averaged over 10 runs. While, we observe a decrease in the test accuracy, it is not seen to significantly impact the operation of the BNN and an accuracy of ≈ 80 % is maintained for 10 % variation.

Conclusion

This work demonstrates the development of computational primitives needed for a BNN accelerator, using 2D memtransistor. The cycle-to-cycle variation in the programming of the memtransistor is exploited as a source of randomness and a circuit comprising of two such memtransistors is used to obtain an ultra-low-power and stochastic synapse, which allows sampling of both positive and negative weights from a Gaussian distribution with reconfigurable mean and standard deviation. We also developed circuits to implement the modified hyperbolic tangent and sigmoid activation functions based on the 2D memtransistors. Additionally, we integrate these components into a crossbar array architecture to perform efficient MAC operations. Finally, we develop a BNN accelerator to perform on-chip inference to classify the iris dataset and benchmark using circuit simulations.

Methods

Device fabrication: Local back-gated MoS_2 memtransistors are fabricated using photolithography and e-beam lithography. Photo-lithography is used to define the back-gate islands. A p^{++} Si substrate is first spin coated with LOR 5A and baked at 180 ˚C for 120 s, and subsequently spin coated with SPR 3012 and baked at 95 ˚C for 60 s. Using Heidelberg MLA 150, the desired regions are exposed to 405 nm light. The exposed regions are developed using 1:1 CD 26 and DI water. To form the back gate islands, 20 nm TiN followed by 50 nm Pt is deposited through sputtering. 50 nm Al_2O_3 gate dielectric is deposited using atomic layer deposition. Al_2O_3 is etched from back gate contact regions using BCl_3 etch, where the etch region was defined by photolithography. Following this MOCVD MoS_2 is transferred onto this substrate and the MoS_2 transistors are fabricated as discussed in our previous reports [25, 39].

Electrical characterization: Lake Shore CRX-VF probe station and Keysight B1500A parameter analyzer were used to perform the electrical characterization at room temperature. The device-to-device variation measurements were performed using the FormFactor Cascade Summit 12000 semi-automated probe station.

Data availability: The datasets generated during and/or analyzed during the current study are available from the corresponding authors on reasonable request.

Code availability: The codes used for plotting the data are available from the corresponding authors on reasonable request.

References

[1] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," pp. 278-282, 2018.

[2] D. N. Fente and D. Kumar Singh, "Weather Forecasting Using Artificial Neural Network," pp. 1757-1761, 2018.

[3] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications,* vol. 38, pp. 10389-10397, 2011.

[4] T. H. Le, "Applying Artificial Neural Networks for Face Recognition," *Advances in Artificial Neural Systems,* vol. 2011, pp. 1-16, 2011.

[5] S. Nirkhi, "Potential use of Artificial Neural Network in Data Mining," pp. 339-343, 2010.

[6] I. Nusrat and S.-B. Jang, "A Comparison of Regularization Techniques in Deep Neural Networks," *Symmetry,* vol. 10, p. 648, 2018.

[7] L. Ding, D. Li, B. Liu, W.-H. Lan, B. Bai, Q. Hao*, et al.*, "Capture Uncertainties in Deep Neural Networks for Safe Operation of Autonomous Driving Vehicles," *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom),* pp. 826-835, 2021.

[8] S. Chakraborty and M. Ghosh, "Applications of Bayesian Neural Networks in Prostate Cancer Study," vol. 28, pp. 241-262, 2012.

[9] J. L. Ticknor, "A Bayesian regularized artificial neural network for stock market forecasting," *Expert Systems with Applications,* vol. 40, pp. 5501-5506, 2013.

[10] J. Shi, Y. Zhu, F. Khan, and G. Chen, "Application of Bayesian Regularization Artificial Neural Network in explosion risk analysis of fixed offshore platform," *Journal of Loss Prevention in the Process Industries,* vol. 57, pp. 131-141, 2019.

[11] M. Kayri, "Predictive Abilities of Bayesian Regularization and Levenberg–Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data," *Mathematical and Computational Applications,* vol. 21, p. 20, 2016.

[12] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," *ArXiv,* vol. abs/1505.05424, 2015.

[13] D. J. C. MacKay, "A Practical Bayesian Framework for Backpropagation Networks," *Neural Computation,* vol. 4, pp. 448-472, 1992.

[14] Y. Gal and Z. Ghahramani, "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference," *ArXiv,* vol. abs/1506.02158, 2015.

[15] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Machine Learning,* vol. 50, pp. 5-43, 2003.

[16] R. Cai, A. Ren, N. Liu, C. Ding, L. Wang, X. Qian*, et al.*, "VIBNN: Hardware Acceleration of Bayesian Neural Networks," *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems,* 2018.

[17] W. Chen, H. Qiu, J. Zhuang, C. Zhang, Y. Hu, Q. Lu*, et al.*, "Quantization of Deep Neural Networks for Accurate Edge Computing," *ACM Journal on Emerging Technologies in Computing Systems,* vol. 17, pp. 1-11, 2021.

[18] A. Marchisio, M. A. Hanif, F. Khalid, G. Plastiras, C. Kyrkou, T. Theocharides*, et al.*, "Deep Learning for Edge Computing: Current Trends, Cross-Layer Optimizations, and Open Research Challenges," pp. 553-559, 2019.

[19] A. Ardakani, C. Condo, and W. J. Gross, "Fast and Efficient Convolutional Accelerator for Edge Computing," *IEEE Transactions on Computers,* vol. 69, pp. 138-152, 2020.

[20] Y. Hirayama, T. Asai, M. Motomura, and S. Takamaeda, "A Hardware-efficient Weight Sampling Circuit for Bayesian Neural Networks," *2020,* vol. 10, p. 10, 2020-07-20 2020.

[21] R. Cai, A. Ren, L. Wangy, M. Pedramy, and Y. Wang, "Hardware Acceleration of Bayesian Neural Networks Using RAM Based Linear Feedback Gaussian Random Number Generators," pp. 289-296, 2017.

[22] A. Malhotra, S. Lu, K. Yang, and A. Sengupta, "Exploiting Oxide Based Resistive RAM Variability for Bayesian Neural Network Hardware Design," *IEEE Transactions on Nanotechnology,* vol. 19, pp. 328-331, 2020.

[23] T. Dalgaty, E. Esmanhotto, N. Castellani, D. Querlioz, and E. Vianello, "Ex Situ Transfer of Bayesian Neural Networks to Resistive Memory-Based Inference Hardware," *Advanced Intelligent Systems,* vol. 3, p. 2000103, 2021.

[24] K. Yang, A. Malhotra, S. Lu, and A. Sengupta, "All-Spin Bayesian Neural Networks," *IEEE Transactions on Electron Devices,* vol. 67, pp. 1340-1347, 2020.

[25] A. Sebastian, R. Pendurthi, T. H. Choudhury, J. M. Redwing, and S. Das, "Benchmarking monolayer MoS_2 and WS_2 field-effect transistors," *Nat Commun,* vol. 12, p. 693, Jan 29 2021.

[26] A. Dodda, A. Oberoi, A. Sebastian, T. H. Choudhury, J. M. Redwing, and S. Das, "Stochastic resonance in MoS_2 photodetector," *Nat Commun,* vol. 11, p. 4406, Sep 2 2020.

[27] S. Das, A. Sebastian, E. Pop, C. J. McClellan, A. D. Franklin, T. Grasser*, et al.*, "Transistors based on two-dimensional materials for future integrated circuits," *Nature Electronics,* vol. 4, pp. 786-799, 2021.

[28] K. Zhu, C. Wen, A. A. Aljarb, F. Xue, X. Xu, V. Tung*, et al.*, "The development of integrated circuits based on two-dimensional materials," *Nature Electronics,* vol. 4, pp. 775-785, 2021/11/01 2021.

[29] A. Sebastian, R. Pendurthi, T. H. Choudhury, J. M. Redwing, and S. Das, "Benchmarking monolayer MoS2 and WS2 field-effect transistors," *Nature Communications,* vol. 12, p. 693, 2021/01/29 2021.

[30] A. Sebastian, A. Pannone, S. Subbulakshmi Radhakrishnan, and S. Das, "Gaussian synapses for probabilistic neural networks," *Nat Commun,* vol. 10, p. 4199, Sep 13 2019.

[31] S. Subbulakshmi Radhakrishnan, A. Sebastian, A. Oberoi, S. Das, and S. Das, "A biomimetic neural encoder for spiking neural network," *Nat Commun,* vol. 12, p. 2143, Apr 9 2021.

[32] D. Jayachandran, A. Oberoi, A. Sebastian, T. H. Choudhury, B. Shankar, J. M. Redwing*, et al.*, "A low-power biomimetic collision detector based on an in-memory molybdenum disulfide photodetector," *Nature Electronics,* vol. 3, pp. 646-655, 2020.

[33] S. Das, A. Dodda, and S. Das, "A biomimetic 2D transistor for audiomorphic computing," *Nature Communications,* vol. 10, p. 3450, 2019/08/01 2019.

[34] A. J. Arnold, A. Razavieh, J. R. Nasr, D. S. Schulman, C. M. Eichfeld, and S. Das, "Mimicking Neurotransmitter Release in Chemical Synapses via Hysteresis Engineering in MoS2 Transistors," *ACS nano,* vol. 11, pp. 3110-3118, 2017.

[35] T. Dalgaty, N. Castellani, C. Turck, K.-E. Harabi, D. Querlioz, and E. Vianello, "In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling," *Nature Electronics,* vol. 4, pp. 151-161, 2021.

[36] A. Wali, H. Ravichandran, and S. Das, "A Machine Learning Attack Resilient True Random Number Generator Based on Stochastic Programming of Atomically Thin Transistors," *ACS Nano,* Oct 19 2021.

[37] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, "Analog architectures for neural network acceleration based on non-volatile memory," *Applied Physics Reviews,* vol. 7, p. 031301, 2020.

[38] C.-H. Chang, H.-Y. Kao, and S.-H. Huang, "Hardware Implementation for Multiple Activation Functions," pp. 1-2, 2019.

[39] A. Sebastian, S. Das, and S. Das, "An Annealing Accelerator for Ising Spin Systems Based on In-Memory Complementary 2D FETs," *Adv Mater,* vol. 34, p. e2107076, Jan 2022.

[40] D. Dua and C. Graff, "{UCI} Machine Learning Repository," 2017.

[41] J. Antoran, "Bayesian-Neural-Networks," 2019.

[42] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis,* vol. 142, p. 106816, 2020.

[43] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?," presented at the Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, California, USA, 2017.

AUTHOR INFORMATION

Corresponding Authors

sud70@psu.edu, das.sapt@gmail.com, amritsebastian@gmail.com

Author Contributions

A.S conceived the idea and designed the experiments. A.S performed the experiments, analyzed the data. A.S and S.D discussed the results, agreed on their implications. All authors contributed to the preparation of the manuscript.

Competing Interest

The authors declare no competing interests

Acknowledgement

The work was supported by Army Research Office (ARO) through Contract Number W911NF1920338. Authors also acknowledge Mr. Shiva Subbulakshmi Radhakrishnan for help with m-tanh measurements. Authors also acknowledge the materials support from the National Science Foundation (NSF) through the Pennsylvania State University 2D Crystal Consortium–Materials Innovation Platform (2DCCMIP) under NSF cooperative agreement DMR-1539916.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryInformation.pdf