# A Study of In-Memory Computing Non-Idealities

Justin Doong, Kourosh Hakhamaneshi, and Alexis Moreno

***Abstract*—While machine learning has been a tremendously powerful tool in many applications, its data-centric nature has resulted in memory access dominating power consumption. Recently, there has been considerable work on in-memory computing, a paradigm that can significantly reduce energy consumption but also introduces reliability concerns due to device variations. This work investigates the non-idealities of in-memory computing using modulated word lines and standard 6T SRAM cells. Monte Carlo simulations were run for amplitude modulated and pulse-width modulated word lines to quantify output variation for each method over a range of inputs and stored weights. We found that the pulse-width modulated word line approach consistently produced less output variation.**

## I. Introduction

In many cases, training a machine learning (ML) model can take several hours or even longer and thus is typically performed in the cloud. Inference, on the other hand, can be performed either in the cloud or at the edge (*e.g.*, IoT or mobile). Many applications require ML inference to be performed close to the sensors because of communication bandwidth limitations. For example, biomedical diagnostics often need to be performed right on the sensor rather than in the cloud.

The basic functions that need to be performed during inference are multiply and accumulate (MAC), $f(\Sigma_i w_i x_i)$, and a non-linear clipping function which determines the class (*e.g.*, $\mathrm{sgn}(x)$). Typically, fetching data in and out of memory is the bottleneck for energy consumption as ML models require a lot of data to be communicated between memory and the processor unit. For example, in $45\,\mathrm{nm}$ CMOS, a 16-b word demands $100\,\mathrm{pJ}$ per access from $1\,\mathrm{MB}$ memory versus just $1\,\mathrm{pJ}$ per multiply [1]. While there are a couple of approaches both in hardware and software to reduce this data movement [2], a lot of attention has been given to in-memory computing (IMC). In this approach, instead of fetching weights out of the memory, inputs are applied to the memory, and classification results are directly taken from the output.

The ML community has shown that if the weights are constrained to $\pm 1$, there is minimal loss in accuracy, provided the model is trained properly [3]. Almost all of the recent work on IMC has leveraged this binary representation of weights.

The fundamental idea is that the model is trained outside of the chip/SRAM with co-optimization between hardware and algorithm in mind (*e.g.*, binary weighted networks, or BWN), and then the learned weights are loaded onto the SRAM to do the inference task. Then, digital input features (*i.e.*, sensory data) are fed into the SRAM and converted to an analog representation which cause a MAC operation (either in current or voltage domain) to take place, and then the sense amplifiers (SA) clip this accumulation to perform classification.

The main challenge is that since the computation is mainly in the analog domain, it is very susceptible to random variations (*e.g.*, threshold voltage $V_{\mathrm{th}}$ variation of bit cells). It is worth noting that some ML algorithms are less affected by such variations if these variations are taken into account while training. There have been hardware approaches which have leveraged this immunity to perform accurate computations (*e.g.*, EACB [4]).

This work investigates the effect of the choice of representation of the analog inputs on output variation due to circuit non-idealities. We compared two approaches, amplitude modulated word lines (AM-WL) and pulse-width modulated word lines (PWM-WL). To quantify output variation, we ran Monte Carlo simulations using different inputs and stored weights to test the system over a range of computation conditions.

The remainder of this paper is organized as follows. Section II provides an overview of the state of the art and the different approaches. Section III presents the approach in this work. Section IV describes the experimental setup. Section V details the results of this work. Finally, Section VI concludes this paper.

## II. State of the Art

There have been demonstrations of a few different approaches to IMC using mostly standard SRAM cells [5], [6], and [7]. The approaches all involve analog word-line driving to represent inputs, and all demonstrate significant energy savings over conventional architectures.

### A. Amplitude Modulated Word Lines

In the system demonstrated in [5], computation is performed in-place by the bit cells of an SRAM, with word-line DACs (WLDACS) translating input data to analog voltages driving the word lines (WL). This approach suffers from circuit non-idealities, especially high

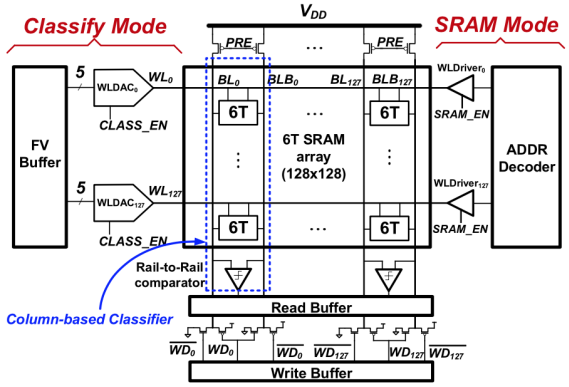


Fig. 1. Architecture of in-memory classifier in [5], showing periphery for two modes of operation.

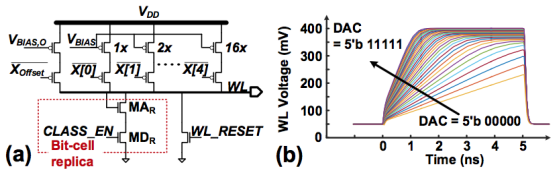


Fig. 2. (a) Current mode WLDAC in [5]. (b) Transient WL pulses with variable settling time for different input codes.

variability in bit cells, which degrades the quality of outputs. Fig. 1 illustrates this architecture.

This system operates in two modes: 1) SRAM mode, or standard read and write, and 2) classify mode, in which the WLs are driven by the analog representation of the input feature vector, which is generated by the WLDAC circuit in Fig. 2. Each column is a weak classifier, and EACB [4] is used to overcome the fitting errors of the weak classifiers as well as the non-idealities and variabilities of the circuit. The operation is as follows. First, the bit lines (BLs) are pre-charged. Then, all the WLs are driven at once to analog voltages corresponding to the elements of the inputs $X$ (*i.e.*, the features); this is done via the peripheral WLDACs. Thus, each bit cell pulls a current modulated by its WL voltage from either BL or BLB, depending on its stored state. This approximates multiplication by $\pm 1$. The bit-cell currents from the column then sum, as in an inner product, discharging BL or BLB. Finally, a comparator provides sign thresholding of the differential signal. Note that all bit cells in the row provide a modulated current $I_{\mathrm{bc}}$ (scaled by the replica sizing ratio) meaning a large number of bit cells drive a single BL/BLB at the same time (128 in the prototype). To not saturate discharge all of the BL/BLB capacitances, $Q_{\mathrm{bc}}$ of each is designed to be low ($< 10\,\mathrm{fC}$), requiring WL voltages $< 0.4\,\mathrm{V}$.

The demonstrated system using EACB achieved the ideal accuracy after 18 iterations. This system achieved 113X lower energy than a discrete system with a standard training algorithm and 13X lower energy than a discrete system with the proposed training algorithm.

### *B. Pulse-Width Modulated Word Lines*

The use of pulse-width modulated word line (PWM-WL) signals to read multiple rows simultaneously was proposed in [6], as shown in Fig. 3. There are key similarities between this approach and the amplitude-modulated WL (AM-WL) scheme proposed in [5]; both perform multi-bit read accesses/operations by first pre-charging the BLs/BLBs and then driving WLs with pseudo-analog signals that correspond to inputs (*e.g.*, features for classification). In [5], the WL amplitude was used to encode information, while in [6], WL pulse width was used. For a given column, driving the WLs with PWM signals results in a BL voltage drop proportional to the weighted sum of the bits (*e.g.*, weights) stored along the column. The longer the pulse-width, the more charge is delivered for the given current, which itself is just set by the bit stored in the SRAM cells and the voltage of the word-line. This contrasts with the mechanism in [5], in which the current itself varies. For example, to perform a dot product for an ML algorithm such as support vector machine (SVM), binary weights are stored in the SRAM cells while the pulse-width of the WLs are varied to represent the input data. A single column, with the various inputs from the various pulse-width modulated word lines, would produce a single dot product.

Like the AM-WL approach, the PWM-WL has significant limitations related to throughput, power efficiency, and robustness. The PWM-WL pulse width is limited by the BL $RC$ time constant for linearity and read stability reasons. In [6], the pulse width was chosen to be less than 40% of the time constant. Moreover, just like in [5], the WL voltage $V_{\mathrm{WL}}$ cannot swing the full voltage range in order to prevent destructive read. To aid SRAM read given short pulse-width requirements, sub-ranged read was used in [6].

When compared to a conventional architecture, the approach in [6] performed 16 times fewer read accesses and had 5.8 times greater throughput while still maintaining greater than 90% accuracy. This resulted in approximately an order of magnitude improvement in energy consumption over conventional architectures as shown in Fig. 4. However, the energy-delay product (EDP) and memory size were much greater than other IMC approaches [5].

## III. Proposed Approach

Employing adaptive methods that consider circuit non-idealities during training has shown great promise for

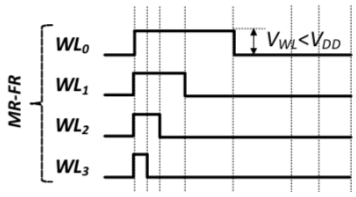


Fig. 3. PWM-WL approach to analog representation of inputs for IMC architecture [6].

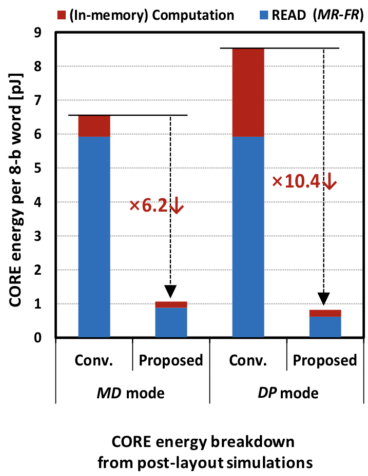


Fig. 4. Energy consumption comparison between the PWM-WL and conventional architectures [6].

achieving high accuracy in IMC systems for simpler classification tasks. But for more sophisticated tasks, *e.g.* image classification on a mobile system while capturing videos, performing training locally may be very challenging. Traditionally, training has been performed offline, *e.g.*, in the cloud. However, in order to take circuit non-idealities into account, training must be performed locally. Training on a large amount of data is expensive in terms of time, power, and sometimes also area (if an auxiliary training co-processor is necessary). In addition, some of these circuit non-idealities may change over the lifetime of the device, meaning occasional retraining may be required. Therefore, the cost of training must be reduced as much as possible. In the case of the EACB approach, this means reducing the number of iterations required to achieve a high accuracy. This in turns demands reduced computational variation relative to the nominal case.

While a complete EACB implementation employs multiple columns (*i.e.*, multiple weak classifiers), this study focuses on the variation of the output of a single column or weak classifier. This reduced output variation for a single column in turn translates into fewer training iterations and/or fewer columns (*i.e.*, fewer weak classifiers) required to achieve the same accuracy.

In particular, this work investigates the effect of the choice of word-line modulation method on the variation in the computational output of a single column (*i.e.*, one weak classifier). The two state-of-the-art methods that we compared are amplitude modulated word-lines (AM-WL), as used in [5], and pulse-width modulated word lines (PWM-WL), as used in [6].

## IV. Experimental Setup

Fig. 5 illustrates the structure of the setup for both AM-WL and PWM-WL. For both cases, the input was a 5-bit value representing the input features to the network. The number of rows of SRAM corresponds to the number of input features. An 8-row setup was used in this study. We designed a standard 6T SRAM cell with pull-down, pull-up, and access transistors sized appropriately to balance read stability, read access, and writeability. Monte Carlo simulations were performed using the SAED32 technology to study the variation in the computation due to the variation of the cells. In this design kit, the Monte Carlo simulation only takes into account the threshold voltage variations of the transistors. However, this is a reasonable effect to study because design techniques can be used to mitigate the effect of process corners and temperature variations, as done in [5]. For a given set of inputs and weights, we measured the variation in the voltage difference between BL and BLB. This was repeated over a functionally exhaustive range of weights and a large set (100) of random inputs.

As was demonstrated in [5], depending on the inputs applied to the WL and the weights stored in the SRAM, the current direction and value can differ. However, the worst case is when all the inputs are maximum and all the weights are the same. Although this case almost never happens in practical applications of ML, it is best to design a hardware system that can handle such a case. In that situation, the sum of the currents in the same direction should still not saturate the BL/BLB signals. Therefore, under that worst case condition, we measured the maximum WL voltage or pulse width that can be applied to the WL such that the BL/BLB signals do not saturate. The clock frequency was set to $500\,\mathrm{MHz}$, and extracted data from standard SRAM cells in this design kit showed that for 8 rows, the BL/BLB capacitance is about $2.2\,\mathrm{fF}$. To produce a swing of $1\,\mathrm{V}$ on BL/BLB at the fixed sampling time of $0.4T_{\mathrm{Clk}}$ for both approaches, the maximum WL voltage for AM-WL was found to be $0.2\,\mathrm{V}$, and the maximum WL pulse-width was found to be $0.8\,\mathrm{psec}$. Fig. 6 and Fig. 7 show the voltage waveforms for AM-WL and PWM-WL, respectively.

## V. Results

In general, the PWM-WL approach was found to have less output variation than the AM-WL approach. Fig. 8

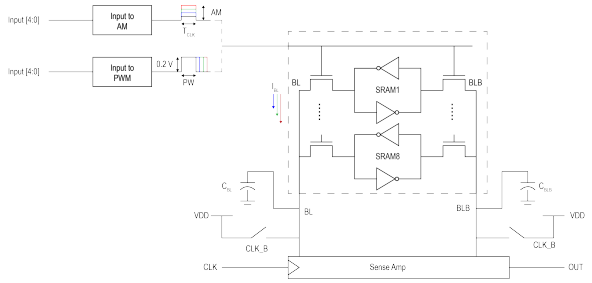


Fig. 5. Experimental setup showing AM-WL and PWM-WL approaches, a column of 8 SRAM cells, and BLs/BLBs differentially sampled by a sense amplifier.

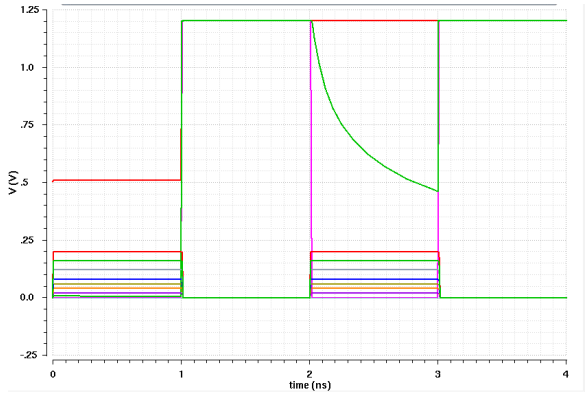


Fig. 6. AM-WL. The multi-colored lines grouped together are WLs set to different analog inputs/amplitudes for different rows. The green line is BL. The red line is BLB. The pink line is CLK.

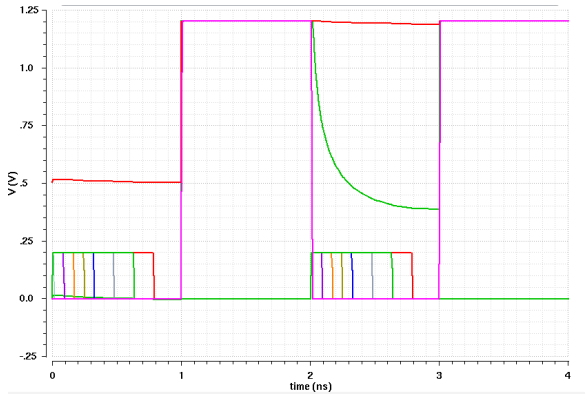


Fig. 7. PWM-WL. The multi-colored lines grouped together are WLs set to different analog inputs/pulse widths for different rows. The green line is BL. The red line is BLB. The pink line is CLK.

and Fig. 9 show the voltage waveform outputs of the AM-WL and PWM-WL testbenches for one particular input vector and combination of stored weights. In this case, the inputs are all maximum ($V = 0.2\,\mathrm{V}$ for AM-WL and $\mathrm{PW} = 0.8$ for PWM-WL) and the weights are all ones.

Fig. 12 summarizes the difference in output variation between the AM-WL and PWM-WL methods. Each subplot is for a different combination of stored weights. Only five combinations of weights need to be tested to be exhaustive because the order of the weights does not affect the computation performed. The x-axis is the difference between the standard deviation of the AM-WL differential BL voltage and the standard deviation of the PWM-WL differential BL voltage. The y-axis is the number of input vectors that produced that difference in standard deviation. By using a sufficiently high number of random input features, we were able to statistically compare the output variation of the two approaches. For example, on average, the standard deviation in differential BL voltage is $7.5\,\mathrm{mV}$ smaller for PWM-WL than for AM-WL. In fact, the PWM-WL almost always has less output variation. This is further illustrated in the box plots shown in Fig. 10 and Fig. 11. Here, it is again clear that mean standard deviation in differential BL voltage for PWM-WL is less than that of AM-WL across all combinations of stored weights.

We have shown how variability within SRAM cells can change the result of the MAC operation required in ML applications and found that PWM-WL is more robust to these variations than AM-WL. Therefore, employing adaptive boosting algorithms can be more effective using the PWM-WL approach. However, another consideration is how these pseudo-analog WL signals are generated. In [5], the authors used current DACs instead of voltage DACs to implement AM-WL. This itself reduces the variability significantly, particularly for variations that arise from process corners and temperature variations. However, this technique still suffers from inherent SRAM bit-cell threshold voltage variations. Moreover, current DACs require significant area overhead. Fig. 13 illustrates the layout of a 5-bit current DAC with a size of $88 \times 27\ \mu\mathrm{m}^2$ for $45\,\mathrm{nm}$ technology. In our $32\,\mathrm{nm}$ technology, this current DAC would take up $44 \times 12\ \mu\mathrm{m}^2$ an SRAM cell of $1.65 \times 1.3\ \mu\mathrm{m}^2$.

## VI. Conclusion

Recent work on IMC has demonstrated significantly reduced power consumption for ML applications by reducing the frequency and number of memory accesses. However, a major hurdle has been susceptibility to variations. While adaptive boosting algorithms have been developed to mitigate this weakness, these techniques require time-consuming and power-intensive iterative training. Thus, minimizing output variation is important for reducing training time, particularly for applications in which training must be performed locally. By running Monte Carlo simulations over a range of inputs and stored weights, we found that the PWM-WL approach consistently outperformed the AM-WL method in terms of output variation. Therefore, it appears that PWM-WL

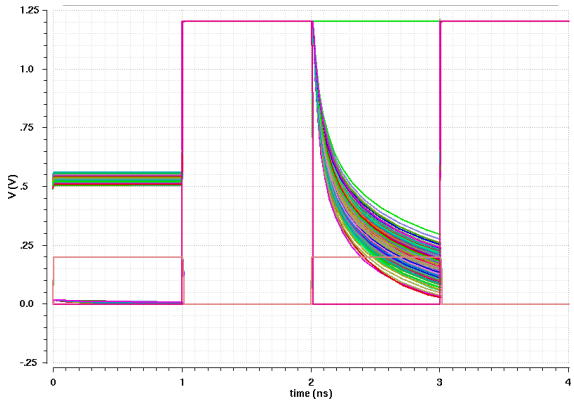


Fig. 8. A single Monte Carlo simulation (200 runs) for one particular combination of inputs and weights for the AM-WL approach. The multi-colored lines are BL. The orange line is WL. The green line is BLB. The pink line is CLK.

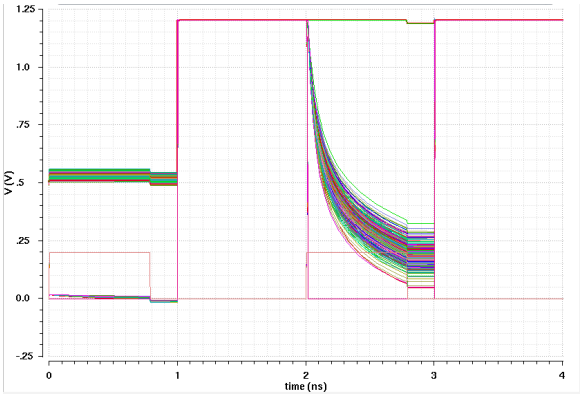


Fig. 9. A single Monte Carlo simulation (200 runs) for one particular combination of inputs and weights for the PWM-WL approach. The multi-colored lines are BL. The orange line is WL. The green line is BLB. The pink line is CLK.

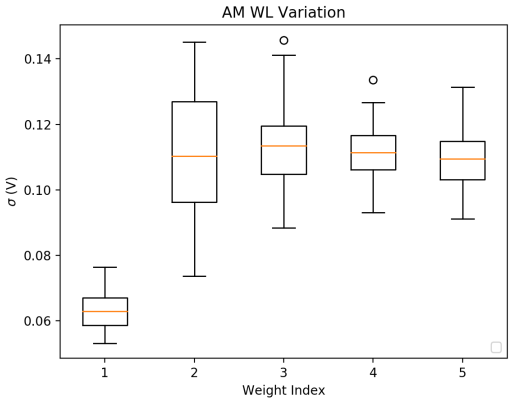


Fig. 10. AM-WL box plot. The y-axis is standard deviation of differential BL/BLB voltage over the Monte Carlo simulation. Each box provides the range of standard deviation over different inputs given a particular combination of stored weights (x-axis).

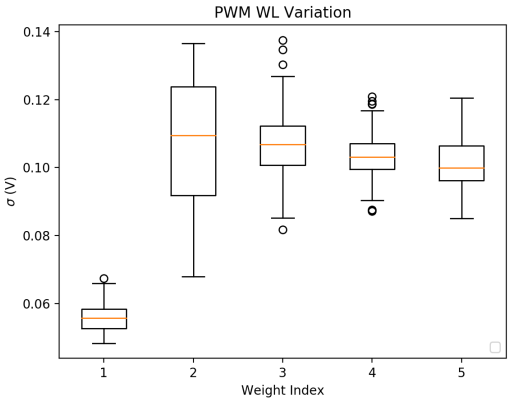


Fig. 11. PWM-WL box plot. The y-axis is standard deviation of differential BL/BLB voltage over the Monte Carlo simulation. Each box provides the range of standard deviation over different inputs given a particular combination of stored weights (x-axis).

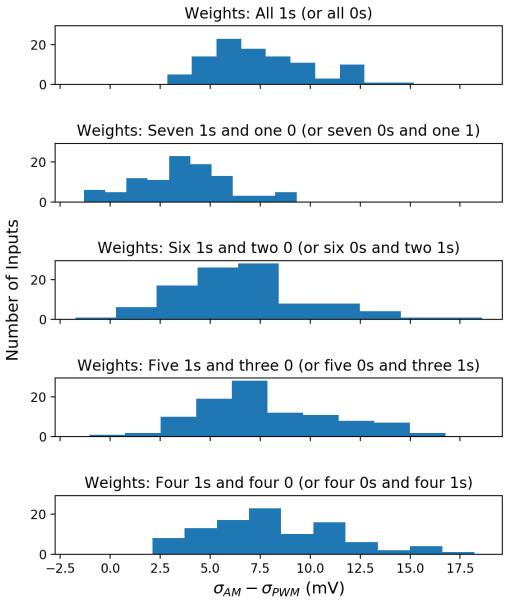


Fig. 12. Histogram of number of inputs for differences between AM-WL and PWM-WL in standard deviation of differential BL voltage.

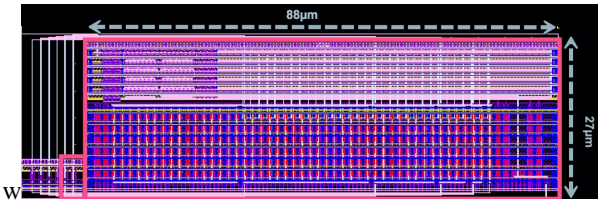


Fig. 13. layout of a 5 bit current DAC in 45nm node

is superior to AM-WL when reducing the amount of iterative training is critical.

However, reducing the output variation of a single column (*i.e.*, a single weak classifier) is not necessarily crucial. ML algorithms are inherently robust to variation as they rely on statistics rather than exact computation. More investigation will have to be done to determine whether the difference in variation between PWM-WL and AM-WL is significant from an algorithmic perspective. For instance, it is possible that a much greater difference in output variation is necessary to decrease the number of training iterations required in EACB.

Moreover, there are also other variations that were not tested in this study that may dominate a real system. These include global process corners and temperature variations. In the case of [5], the authors employed current DACs that used replicas to track the SRAM columns across these variations. By combining this technique with a precisely-produced pulse width, a similarly robust PWM-WL approach is possible. Thus, these other sources of variation can likely be all but eliminated while threshold voltage mismatch will remain.

In addition, there are also peripheral circuit considerations. For example, to generate robust AM-WL signals, the authors in [5] needed to build current DACs that took up considerable area. Finely controlling the pulse width for PWM-WL is more easily achieved without resorting to using a large, specialized DAC.

There is still much work to be done to confirm that PWM-WL is a better approach for IMC than AM-WL. We believe this work begins to make the case for PWM-WL by quantitatively establishing its reduced output variation due to inherent threshold voltage mismatch and variations.

## REFERENCES


[1] M. Horowitz, "Computings energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 10–14.

[2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," in *Proc. of the IEEE*, vol. 105.

[3] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *arXiv*, Nov. 2015. [Online]. Available: https://arxiv.org/abs/1511.00363

[4] Z. Wang, R. E. Schapire, and N. Verma, "Error adaptive classifier boosting (eacb): Leveraging data-driven training towards hardware resilience for signal inference," vol. 62.

[5] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6t sram array," vol. 52.

[6] M. Kang, S. Gonugondla, A. Patil, and N. Shanbhag, "A 481pj/decision 3.4m decision/s multifunctional deep in-memory inference processor using standard 6t sram array," *arXiv*, Oct. 2016. [Online]. Available: https://arxiv.org/pdf/1610.07501.pdf

[7] A. Biswas and A. P. Chandrakasan, "Conv-ram: An energy-efficient sram with embedded convolution computation for low-power cnn-based machine learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 488–489.