

Analyzing the Effects of Noise and Variation on the Accuracy of Analog Neural Networks

Devon Janke

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250
Email: djanke3@gatech.edu

David V. Anderson

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250
Email: anderson@gatech.edu

Abstract—Traditional machine learning algorithms and neural networks implemented using digital architectures such as GPUs, TPUs, and FPGAs demonstrate high performance, but the power required to train and predict is too high to be implemented in energy-constrained systems such as implants and edge devices. Although analog classifiers offer the possibility to significantly reduce power consumption by two or three orders of magnitude, the nonidealities inherent to analog circuits such as noise, drift, and process variations make it very difficult to implement accurate analog neural networks. This paper explores the effects of these nonidealities on classification performance.

Index Terms—Analog circuits, neural networks, ultra-low power, IoT, voice activity detection, edge computing

I. INTELLIGENT SENSOR PROCESSING

In the age of the “Internet of Things” (IoT), data is transmitted to and from almost anything imaginable including phones, televisions, cameras, ovens, and even doorbells. These connected devices have integrated sensors that collect information about their surroundings, including audio, visuals, and temperature. After collecting the data, so-called smart devices adjust their performance and carry out tasks based on user interaction and preferences.

The key to transforming simple transducers into smart sensors lies in machine learning (ML) and Neural Networks (NN). In order to reach the impressive performance levels that consumers come to expect, large datasets of terabytes of data are processed via billions of multiplications, additions, and read and write operations are performed on devices made up of billions of transistors. When the classifier finally reaches a desired level of functionality, it is deployed and packaged with appropriate sensors.

While modern implementations of NN architectures and ML algorithms demonstrate high performance in a wide range of fields, the energy required to reach this performance has grown along with the underlying NNs. The process of training can demand between 100 W and 250 W on a single GPU, with each training iteration costing at least 200 mJ per image [1]. The energy required to make predictions varies depending on the architecture, quoted at about 60 mJ and 1180 mJ for state-of-the-art architectures AlexNet and VGG-16 respectively on the Nvidia GeForce GTX Titan X GPU [2]. Not only does this inhibit classifiers from being implemented in power-starved

IoT edge devices, it goes contrary to the energy-reduction mantra in an economy focused on a greener future.

In the search for an alternative, many are re-exploring the possibilities of analog computation. Rather than relying on digital arithmetic and logic, analog computing uses the physical properties of hardware devices [3]. An operation such as multiplication, which requires thousands of transistors in digital, can be reduced to only a few transistors in analog. Because of this, analog circuits are comparatively low-power, faster, and smaller [3]–[14]. Because analog circuits are not dependent on clocking or shared resources as are digital computers, they are inherently parallelizable, further enhancing their speed [12]. A research team at IBM has recently demonstrated these benefits by employing analog in-memory computation [15].

The most significant limitation in analog circuits is the inter-device variability. Due to small inconsistencies in the fabrication process, voltage sources, and temperature, often referred to collectively as PVT variation, post-fabrication, devices such as capacitors and resistors can vary up to 25% from their expected values making it difficult to know their actual properties and causing offset and mismatch errors [16]–[18]. Analog also has no perfect long-term storage; most storage methods suffer from leakage, low-precision, or limitations in writability [12], [13], [19]. For this reason, analog circuits are not as flexible as their digital counterparts [4], [5], [13], [16], and training methods such as backpropagation are more difficult to implement on chip [16], [20]. All these issues limit analog computation to applications with low- to medium-precision data [4], [13], [18], [21]. This paper focuses on demonstrating the effects PVT variations can have on classifier performance and describe a simple tweak to help reduce them.

II. ANALOG IMPLEMENTATIONS

The standard approach used for training basic NN architectures is shown in Fig. 1. The incoming information (e.g., the training data) is pre-processed to extract desired features as inputs to the neural network. A forward propagation step generates a predicted output for each input sample in order to quantify the current accuracy of the NN. In the back-propagation step, the strength of the effect of each synaptic weight on the error (i.e., the gradient) is calculated; the weights

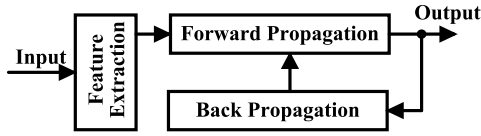


Fig. 1: Illustration of three common steps in a ML algorithm.

are all adjusted proportional to their respective gradient. This structure is common in ML algorithms.

When analog computation is introduced into a ML system, the strategy is normally to partially or completely implement one or more of the aforementioned steps with analog/mixed-signal circuits. Analog-enhanced classifiers are commonly realized via full replacement, acceleration, and deployment.

A. Full Replacement and Acceleration

Introducing analog circuits by full replacement is the effort to complete the entire machine learning process entirely with analog blocks. Since all of the needed mathematical functions can be implemented in analog, it is theoretically possible to complete the full ML cycle. Unfortunately, the information learned by these types of classifiers is very difficult to share and reproduce because each fabricated device can vary significantly from another. At the same time, each read process will corrupt the learned weights [18]. Because of this, the classifier becomes a black box, where very little intuition can be gained about what has been learned.

Rather than completely do away with the entire digital system, one can augment the capabilities of the system by using analog circuits for the most demanding tasks, such as matrix multiply-and-accumulate functions [15]. However, while this may accelerate the actual computation, latency and system complexity and size increase due to the data conversion between analog and digital domains.

One huge issue with using analog computation in a digital system is that high linearity and precision are required. For the acceleration to be useful, there must be a direct translation from analog circuit to digital model, which requires precise trimming and control of each data converter and arithmetic block. This level of precise control is especially required if the intention is to use the learned weight values in a digital system or other hardware classifiers for deployment. These two methods are both very important steps toward low-power ML, but they have significant hurdles to becoming reality.

B. Deployment

The third method for replacing digital with analog in machine learning systems is to implement the trained digital NN as an analog classifier. Training is completed on a digital system such as a computer, and the learned network characteristics, including feature extraction hyperparameters and multiplicative weights, are downloaded to the device [3], [7], [11], [22]–[25]. After the device is programmed, it is ready for use in its selected application.

However, just as with the other methods, the best set of parameters for one device is not the best for another. The trained weights of an ideal system cannot be downloaded to an

analog classifier and produce the same accuracy. To minimize the performance degradation due to these nonidealities, the architecture and training algorithm must generalize well to all variations that could occur within the hardware classifier.

Because of the potential for large-scale production without needing to overcome the hurdles of on-chip training, the work in this paper is focused on deployment-only implementations of analog classifiers. Specifically, the goal is to better understand how various parameter variations affect the accuracy of deployed NNs to identify how to design analog NNs that minimize the impact of PVT noise. This concept is explored in the context of a voice activity detection (VAD) classifier. This application is well-fit for analog NNs because voice data tends to perform well in low-precision systems [26].

III. CLASSIFIER MODELING

One of the earliest NN architectures implemented in analog and digital hardware is the multilayer perceptron (MLP); an example architecture is shown in Fig. 2. The MLP is made up of a sequence of k fully-connected (FC) layers and nonlinear activation functions. A FC layer has m inputs and n outputs, where each output is a linear combination of the inputs. The outputs of the FC layer undergo a nonlinear activation, such as the hyperbolic tangent, to generate the neuron outputs, which then become the inputs to the next layer. The functions governing the input/output relationship of a general MLP are:

$$\bar{z}_i = W_i \bar{a}_{i-1} \quad (1)$$

$$\bar{a}_i = g(\bar{z}_i) \quad (2)$$

$$\bar{a}_0 = \bar{x}, \quad \bar{y} = \bar{a}_k \quad (3)$$

where $g(x)$ is a nonlinear activation such as a hyperbolic tangent or sigmoid function.

For the VAD classifier in this paper, the inputs x to the NN are generated by passing the input audio signal through an array of bandpass filters and one low-pass filter at the lowest frequency. The envelope of each band is detected. The background noise level is estimated by tracking the minimum of the envelope while the voice level is estimated by tracking

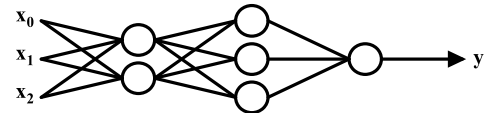


Fig. 2: Block diagram of a multilayer perceptron neural network.

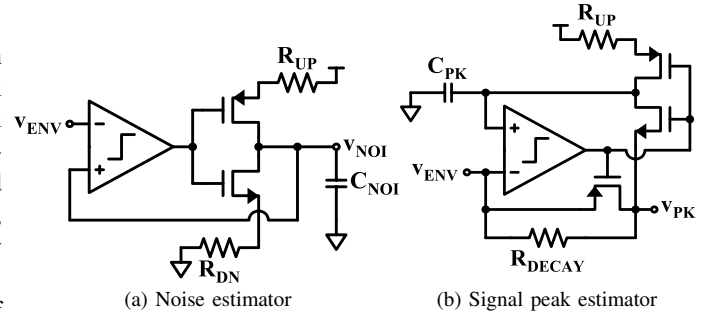


Fig. 3: Analog circuits used for extracting (a) the estimated noise level and (b) the estimated signal peak.

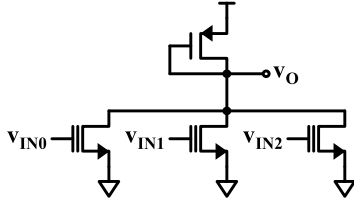


Fig. 4: Example implementation of a three-input neuron. The synaptic weight is stored as a voltage on the floating gate.

the peaks of the envelope. Simplified schematics for generating each of these signals are in Fig. 3. The difference of the voice and noise for each band is the input to the NN.

The NN itself is made up of a series of multipliers and summation blocks. An example of a translation between a neuron model and an actual circuit implementation is shown in Fig. 4, where each synapse is a floating-gate transistor with the synaptic weight determined by the voltage bias stored on the floating gate. The currents are summed together and converted to a voltage to generate the neuron output. The actual implementation and modeling of the full NN using this analog hardware neuron are outside the scope of this paper.

A. Adding Noise

There are limitless parameters in an actual circuit implementation where PVT variations can have an impact. For this analysis, variations will be added generally to a selection of defining parameters, namely, the bandpass center frequencies, Q values and gain, RC time constants used in feature extraction, and the multiplicative weights of the network.

IV. VARIATION DEGRADATION

A. Methodology for Simulating Variations

NN architectures described as small, medium (med), and large as in Table I are used in simulation. All classifiers use the same feature extraction stage with six frequency bands. Randomness will be introduced to each section of the classifier individually and then all together. Before adding the noise, hyperparameters and synaptic weights are learned in order to result in high performance on training and test data.

Randomized instances of the classifier are represented by 500 vectors with values from a gaussian distribution with a mean of one and a standard deviation $3\sigma = 0.2$. Variations are introduced to the ideal network by multiplying each of these parameters by their associated random value, and the new accuracy of the classifier is recorded.

B. Feature Extraction

The first step in the classification task is to generate a set of features from the incoming audio. The feature extraction step

TABLE I: NEURAL NETWORK LAYERS AND SIZES

Size	Inputs	Hidden Layer Sizes	Outputs
Small	6	[6]	1
Med	6	[12,6]	1
Large	6	[18,12,6]	1

introduces a number of manually-tuned hyperparameters such as number of bandpass channels, bandpass center frequency, filter order, time constants, and others. Of these parameters, variations in the bandpass center frequencies, Q values and gain, and RC time constants used in extracting the band's envelope and estimated noise and signal peak levels will likely have the greatest impact on performance.

Fig. 5 shows the distribution of absolute accuracy for each NN size after introducing 500 instances of noise; the main box is the same as it would be in a standard box plot, showing the main quartiles. The impact of the PVT variations in the feature extraction stage is reduced in the deeper NNs.

C. Synaptic Weights

To emulate the effect of floating gate transistors storing a gain value as a voltage on the gate, the learned weight values from training are translated to a “voltage” such that the weight value can be reobtained through (4).

$$w = A \tan(cv) \quad (4)$$

The same voltages are used on all 1000 variations of each NN, and randomness is applied to the A (maximum gain) and c (compression) variables. Fig. 6 shows the effects of this randomness on accuracy when it is applied only to each layer of weights individually and then all together.

V. DISCUSSION

In this and other simulations repeating this same process, a definite pattern is observed. Layers with a large number of synaptic weights ($ins \times outs$) exhibit a larger range of accuracy variation. Layers further from the output, however, have reduced variability. The reason for the deeper layers having less affect on overall accuracy is the activation function. Though it is mainly intended to introduce nonlinearity to the

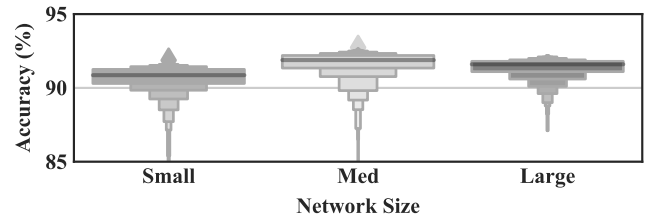


Fig. 5: Distribution of accuracy when introducing noise to the feature extraction stage. Variance is similar for all three NN sizes.

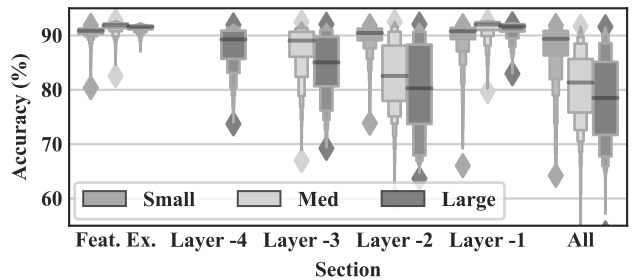


Fig. 6: Distribution of accuracy when introducing noise to the synaptic weights. Layers are numbered according to their position behind the output. The impact of feature extraction variance is much less compared to the effects due to the weights.

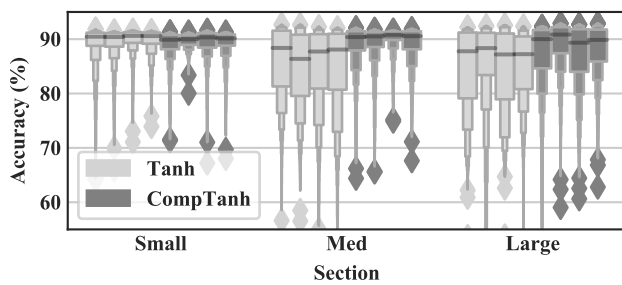


Fig. 7: Effect of replacing the \tanh activation function with (5). The median accuracy and variance improve with a compressed activation.

network, it can also help to suppress variations as they traverse through the network. This is especially true if the activation function has a limited range, such as the \tanh and sigmoid functions, and the suppression is further enhanced when the slope of the activation is also reduced.

If we replace the \tanh activation with a compressed version as in (5) in both the training and deployed models, the variance is noticeably reduced (see Fig. 7). This of course means that the circuit for the activation function must be carefully designed to rely heavily on matching techniques in order to reduce the possible variations in the activation.

$$0.5\tanh(0.5x) + 0.5 \quad (5)$$

The variability due to earlier stages can be limited by reducing the slope of the activation function, and limiting the number of inputs to each layer will also help. Since the final stage in a NN is also commonly the smallest stage, it may be possible to take advantage of transfer learning, where only the final stages are retrained after repurposing a NN for a new task, or in this case, the same task but on different hardware [27].

While neural networks implemented with analog arithmetic currently lag behind current capabilities of modern artificial intelligence, there are still a range of useful tasks, such as voice activity detection, where analog classifiers can provide a much-needed boost. With proper training and design, they can bring all the benefits of artificial intelligence and smart sensing to realms where they are currently impractical.

REFERENCES

- [1] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pp. 477–484.
- [2] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "Neuralpower: Predict and deploy energy-efficient convolutional neural networks," *ArXiv*, vol. abs/1710.05420, 2017.
- [3] A. Jayaraj, I. Banerjee, and A. Sanyal, "Common-source amplifier based analog artificial neural network classifier," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 26-29 May 2019, p. 5 pp.
- [4] P. Kinget and M. S. J. Steyaert, "A programmable analog cellular neural network cmos chip for high speed image processing," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, 1995, pp. 235–43.
- [5] M. Valle, D. D. Caviglia, and G. M. Bisio, "An experimental analog vlsi neural network with on-chip back-propagation learning," *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 3, 1996, pp. 231–45.
- [6] Y. Maeda and T. Kusuhashi, "An analog neural network system with learning capability using simultaneous perturbation," *IEICE Transactions on Information and Systems*, vol. E82-D, no. 12, 1999, pp. 1627–33.
- [7] D. Puxuan, G. L. Bilbro, and C. Mo-Yuen, "Implementation of artificial neural network for real time applications using field programmable analog arrays," in *2006 International Joint Conference on Neural Networks*, 16-21 July 2006, pp. 1518–24.
- [8] B. Herusetto, E. Prasetyo, H. Afandi, and M. Paindavoino, "Embedded analog cmos neural network inside high speed camera," in *2009 1st Asia Symposium on Quality Electronic Design (ASQED 2009)*, 15-16 July 2009, pp. 144–7.
- [9] R. Dlugosz, T. Talaska, and W. Pedrycz, "Current-mode analog adaptive mechanism for ultra-low-power neural networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 1, 2011, pp. 31–5.
- [10] B. Larras *et al.*, "Analog encoded neural network for power management in mpsoic," *Analog Integrated Circuits and Signal Processing*, vol. 81, no. 3, 2014, pp. 595–605.
- [11] J. Binas, D. Neil, G. Indiveri, S. C. Liu, and M. Pfeiffer, "Precise deep neural network computation on imprecise low-power analog hardware," *arXiv*, 2016, p. 21 pp.
- [12] E. Rosenthal, S. Greshnikov, D. Soudry, and S. Kvatinisky, "A fully analog memristor-based neural network with online gradient training," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 22-25 May 2016, pp. 1394–7.
- [13] S. Mingoo, Y. Minhao, J. Zhewei, A. A. Lazar, and S. Jae-Sun, "Cases for analog mixed signal computing integrated circuits for deep neural networks," in *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 22-25 April 2019, p. 2 pp.
- [14] M. Suhong, S. Kwanghyun, and J. Dongsuk, "Enhancing reliability of analog neural network processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, 2019, pp. 1455–9.
- [15] S. Ambrogio *et al.*, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, 2018, pp. 60–67.
- [16] A. J. Montalvo, R. S. Gyurcsik, and J. J. Paulos, "Building blocks for a temperature-compensated analog vlsi neural network with on-chip learning," in *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, 30 May-2 June 1994, vol. vol.6, pp. 363–6.
- [17] *On-Chip Resistors and Capacitors*. Boston, MA: Springer US, 2006, pp. 67–94.
- [18] S. Agarwal *et al.*, "Designing and modeling analog neural network training accelerators," in *2019 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*, 22-25 April 2019, p. 2 pp.
- [19] T. Lehmann, E. Bruun, and C. Dietrich, "Mixed analog/digital matrix-vector multiplier for neural network synapses," in *12th NORCHIP Seminar*, 8-9 Nov. 1994, vol. 9, pp. 55–63.
- [20] A. J. Montalvo, R. S. Gyurcsik, and J. J. Paulos, "An analog vlsi neural network with on-chip perturbation learning," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 4, 1997, pp. 535–43.
- [21] T. Yamasaki and T. Shibata, "Analog soft-pattern-matching classifier using floating-gate mos technology," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, 2003, pp. 1257–65.
- [22] G. Geske, F. Stupmann, and A. Wego, "High speed color recognition with an analog neural network chip," in *2003 IEEE International Conference on Industrial Technology*, 10-12 Dec. 2003, vol. Vol.1, pp. 104–7.
- [23] D. Maliuk, H. G. Stratigopoulos, H. He, and Y. Makris, "Analog neural network design for rf built-in self-test," in *2010 IEEE International Test Conference (ITC 2010)*, 31 Oct.-5 Nov. 2010, p. 10 pp.
- [24] L. Fick *et al.*, "Analog in-memory subthreshold deep neural network accelerator," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, 30 April-3 May 2017, pp. 195–8.
- [25] Y. Minhao *et al.*, "A luv voice activity detector using analog feature extraction and digital deep neural network," in *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, 11-15 Feb. 2018, pp. 346–8.
- [26] J.-H. Ko, J. Fromm, M. Philipose, I. Tashev, and S. Zarar, "Limiting numerical precision of neural networks to achieve real-time voice activity detection," in *IEEE Int. Conf. Acoustics Speech and Signal Processing (ICASSP)*.
- [27] J. Kaige *et al.*, "Calibrating process variation at system level with in-situ low-precision transfer learning for analog neural network processors," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 24-28 June 2018, p. 6 pp.