computing, computational complexity, neural network, neuromorphic, computing limits, thermodynamic limits, analog computer, digital computer

Editor: Erik P. DeBenedictis, Sandia National Laboratories; epdeben@sandia.gov

Rebooting Computing

Computational Complexity and New Computing Approaches[1]

Erik P. DeBenedictis, Sandia National Laboratories

Computational complexity analysis allows us to quantify energy-efficiency scaling potential—an important task for assessing research options.

As we search for new ways to increase both computer performance and energy efficiency, it would be helpful to be able to predict long-term potential in advance. Here, I'll show how computational complexity theory can quantify the energy efficiency potential of analog computing. The method could be applied to other computing approaches.

Analog computers are one option to restore growth in the computer industry. Such growth requires families of computers that can run applications more cost-effectively over time, which today means improving energy efficiency. The improvement rate for the energy efficiency of digital computers has slowed, raising the question of whether analog computers could overtake them.

If analog and digital are viewed broadly as alternative computer implementations, then they should be subject to the same general principles. However, a specific digital computer's effectiveness depends on its architecture and the algorithms running on it. These correspond to the circuitry of an analog computer.

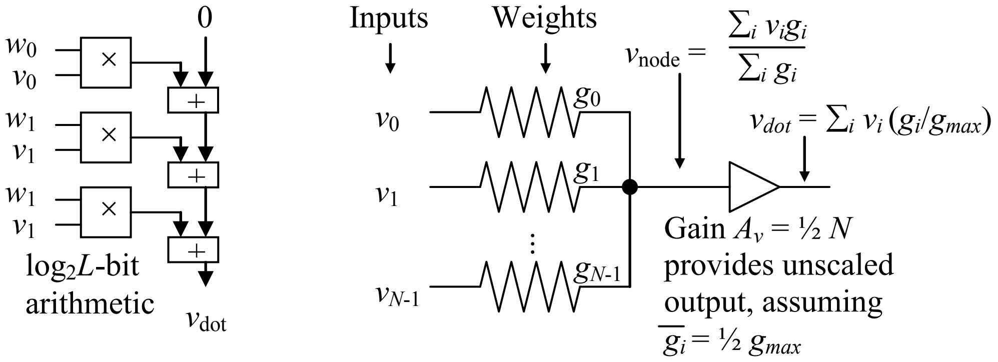
Here, I'll analyze digital and analog "neuromorphic" calculations using a computational complexity theory first developed for digital computer algorithms. The analysis doesn't find a winner but provides new insights into which approach has more potential.

Comparing A Common Function

Meaningful comparison of analog and digital requires a computing task amenable to both approaches. I'll focus on artificial neural networks, where the comparison is between a digital implementation such as deep learning[1-silver] and an analog neuromorphic implementation such as the ohmic weave circuit based on memristors.[2-mountain]

Biological neurons, which fill a role similar to N-input logic gates, mathematically evaluate the computational primitive called dot or inner product. N "presynaptic" neurons generate signals that become inputs of the N-input neuron under consideration. Each of the N signals v_i is multiplied by a synapse weight w_i and the products added to become the neuron's output. A digital implementation multiplies and adds as shown in Figure 1a. The analog circuit illustrated in Figure 1b performs the same computation with a resistor network and an amplifier. The network computes the weighted average of input signals, represented as voltages v_i, weighting each input by the conductance g_i of the programmable resistor or memristor. The conductance is the synapse weight scaled by some factor α. Conductance is the inverse of resistance, so $\alpha w_i = g_i = 1/R_i$. The voltage at position v_{node} in the circuit is the weighted average of the input voltages divided by the sum of the conductances. If the average conductance is half the maximum value $g_{\max,}$ amplifying v_{node} by $A_v = ½\, N$ produces an output voltage equal to the dot product of the inputs.

[1] Sandia National Laboratories approved for unlimited unclassified release SAND2017-8698 J.
Published as DeBenedictis, Erik P. "Computational Complexity and New Computing Approaches." *Computer* 49.12 (2016): 76-79. **DOI:** 10.1109/MC.2016.353


0
w_0
v_0
w_1
v_1
w_1
v_1
×
+
$\log_2 L$-bit arithmetic
v_{dot}
Inputs
Weights
$v_{\text{node}} = \frac{\sum_i v_i g_i}{\sum_i g_i}$
v_0
v_1
$v_{N\text{-}1}$
g_0
g_1
$g_{N\text{-}1}$
$v_{dot} = \sum_i v_i (g_i/g_{max})$
Gain $A_v = ½\, N$ provides unscaled output, assuming $\overline{g_i} = ½\, g_{max}$

(a) Digital (b) Analog

Figure 1: Dot product implementations. (a) A digital implementation performs repeated multiplies and adds. (b) An analog circuit performs the same computation with a resistor network and an amplifier.

Energy Efficiency Comparison

The energy efficiency of a computer that you might hold in your hand isn't what we're analyzing, but rather how that energy efficiency grows over time and as the computer family matures and runs larger problems. Owing to the rapidly advancing semiconductor industry, the digital approach is more energy efficient today and is likely to closely approach its physical limits before maturing in a decade or two. The analog approach would require considerable investment just to come out even, but such an investment might be justified if its energy-efficiency limits were higher.

In the 1960s, Rolf Landauer found that the minimum energy for a computation is kT times the change information entropy due to the computation—with no device dependence (k is Boltzmann's constant and T is the temperature in kelvins).[3-landauer] Landauer's principle is widely understood as an endpoint for digital scaling but hasn't been applied to analog computers.

Table 1 contains computational complexity expressions of the minimum energy for dot product in digital and analog implementations. The table highlights not only the expressions' similarities in form but also their differences in detail.

Table 1: Minimum energy of an N-element dot product of L-level or $\log_2 L$-bit values

Approach	Computational complexity expression for minimum energy				
	Constant factor ×	Reliability ×	Accuracy ×	Scale ×	Energy unit
E_{digital} =	~24	$\ln(1/p_{\text{error}})$	$\log_2{}^2(L)$	N	kT
E_{analog} =	~1/24	$\ln(1/p_{\text{error}})$	L^2	N^2	kT

Digital Implementation

Identifying the energy efficiency limit of the digital approach is straightforward but must be explained using exactly the same terminology as the analog limit that follows. If we assume signals have L distinguishable levels, the digital implementation will require $\log_2 L$-bit binary numbers. The multiplier would be the dominant digital circuit, requiring a $\log_2 L \times \log_2 L$ array of gated full adders of typically six Boolean gates each. For a discussion of the minimum energy of a Boolean gate, see "The Boolean Logic Tax," my April installment of this column.[4-debenedictis] There, I included the equation $\text{p}_{\text{error}} = \exp(-CV^2/(kT))$ as the error rate of an ideal gate, relating the probability of error p_{error} to the energy of a signal CV^2. The gates in a multiplier each have two inputs, so the limiting energy of a gate would be $E_{\text{gate}} = {\sim}2 \ln(1/p_{\text{error}})kT$ by algebraic rearrangement of the previous expression. If we assume 100 percent overhead for addition, control, and so on, the minimum energy for an N-element digital dot product E_{digital} will be as shown in Table 1's first row.

Analog Implementation

I'll discuss the limits of an analog implementation in greater detail because they aren't widely known. The calculation below is based on Sapan Agarwal and his colleagues' work[5-frontiers] but rewritten using the same terminology as the digital implementation discussed above. I use the new terminology in a technical report recommended for readers seeking more detail.[6-turon]

Only the amplifier in Figure 1b draws energy from the power supply, but let's assume an ideal amplifier that transfers energy from power supply to outputs without loss. This leaves the resistors as the only elements dissipating energy, so the expression for the resistors' minimum dissipation is also the expression for the minimum energy of the dot product.

Johnson–Nyquist noise at the amplifier's input v_{node} sets its performance limit, where the noise power is

$$\overline{P_{\text{noise}}} = 4\ kT f = \overline{V_{\text{noise}}}^2\ ½\ g_{\max} N,$$

and f is the amplifier's bandwidth. Conductivities are in the range $0\ldots g_{\max}$, averaging ½ $g_{\max}$.

Assuming the system performs the dot product at the Johnson-Nyquist rate of $2f$, we're interested in the size of noise transients that occur with probability p_{error} at each calculation interval. These peaks will be $\sqrt{}\ln(1/p_{\text{error}})$ error times the average noise. Amplifying the signal to produce the correct output also amplifies the noise, so peak noise at the amplifier's output is

$$\overline{V_{\text{peaknoise}}} = \overline{V_{\text{noise}}}\, A_v \sqrt{}\ln(1/p_{\text{error}}).$$

To match the sampling error of the digital implementation (1/2 of the least significant bit), the signal voltage range must be set at $\pm V = 1/2\ V_{\text{peaknoise}}\ L$. This makes the power dissipated by the resistors in Figure 1b

$$\overline{P_{\text{dot}}{}^{(\text{B})}} = 1/6\ V^2\, g_{max} N,$$

where the factor of 1/6 is the result of averaging signal voltages, and the superscript (B) indicates that a correction will be needed when there are only a few inputs.[6-turon]

The dot product's minimum energy is now the resistors' dissipation multiplied by the sample-to-sample time interval $1/(2f)$.

If Landauer's theory[3-landauer] is correct, the implementation-specific terms f, $g_{\max}$, and V should algebraically cancel and yield a problem-dependent multiple of kT. And this is exactly what happens (see Table 1's second row):

$$\overline{E_{\text{dot}}{}^{(\text{B})}} = \frac{\overline{P_{dot}{}^{(\text{B})}}}{2f} = 1/24\ \ln(1/p_{\text{error}})\ L^2 N^2\ kT,$$

Interpreting the Computational Complexity Expressions

Table 1 contains unusual computational complexity expressions that tell a novel story.[7-compcomplexity] Computational complexity theory developed alongside digital computers, growing strongest where it applied to the computers in use at the time. The theory developed precise methods of counting Turing machine steps, Boolean gate operations, and arithmetic operations. However, Table 1 applies computational complexity to the tallying of minimum energy in units of kT—a valid extension, but only becoming relevant because of today's concern about computational energy efficiency.

The computational complexity of numerical software algorithms such as matrix inversion and dot product designate multiplies and adds to have unit cost and then count arithmetic steps to solve a given problem. Ideally, we're seeking an analog "drop-in replacement" for digital multipliers and adders that would be more energy efficient yet still work in the same algorithms. Unfortunately, the digital expression in Table 1 has a linear N term, whereas the analog expression has a quadratic N^2 term—revealing that analog and digital arithmetic circuits combine differently and thus lead to different algorithms.

The N^2 term's true source is described mathematically elsewhere[5-frontiers][6-turon] but can be explained intuitively from the circuit shown in Figure 1b. Each of the N signals draws energy from the power supply, thus contributing the energy expression's first factor of N. Resistors conduct current in both directions, so some of each input signal's energy leaks backward through the other N–1 resistors, resulting in only 1/N of the energy being available to compete with noise at the amplifier. Energy levels must be increased by a factor of N to raise the signal back up, multiplying the energy by a second factor of N and yielding the N^2. There's no corresponding scenario of a digital value leaking backward into the logic that produced it.

The two approaches also differ in their dependence on the precision L. Both digital and analog approaches need a voltage gap to prevent noise from making a signal value look like an adjacent one. The L gaps must be stacked up in the range of the analog signal, making the signal's range proportional to L. Digital's place-value arithmetic makes a big difference. Each of the $\log_2 L$ bits needs just a single gap between 0 and 1. Digital's logarithmic growth makes for higher accuracy, but its complexity contributes to the constant factor ~24 that offsets analog's ~1/24, giving analog an energy efficiency head start of approximately 242× or 576×.

Figure 2 provides insight into the question of whether analog is better than digital, at least for dot product. The diagram divides the parameter space into regions in which each approach could excel. A reality check with biological neural networks can provide additional understanding. The human brain has many neurons of modest precision, so why does the diagram include it in the "digital best" region? Unlike Figure 1b's deep-learning and neuromorphic circuits, the human brain's architecture is based on spike signaling – which is essentially digital. Did biology figure out that level-based neuron signaling had poor energy-efficiency scaling and choose digital instead?

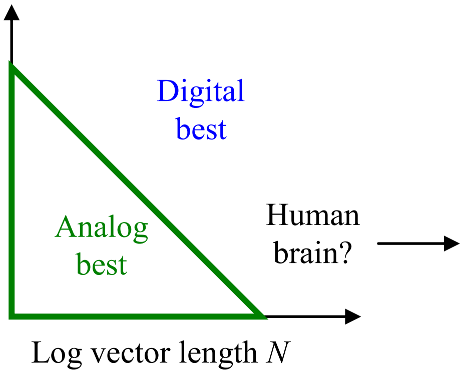


Figure 2: When analog dot product is better than digital.

This analysis presented here refutes the idea that analog computers could be declared fundamentally more energy efficient than digital ones, or vice versa. The two approaches are governed by the same type of physical or thermodynamic limits, although one approach might be better than the other for a specific algorithm or circuit and in a different size range. Computational complexity analysis can identify these domains. Moreover, the idea of using computational complexity analysis for evaluating energy-efficiency potential could be applied to other approaches to computing, such as approximate or stochastic computing with regular logic gates, Josephson junctions, spintronics, quantum computing, and others. Although use of this methodology might not predict the future with 100 percent reliability, the fact that it addresses the right question puts it ahead of other approaches.

Acknowledgments

Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the US Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. [1] Silver et al., "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 529, no. 7587, 2016, pp. 484–489.
2. [2] Mountain et al., "Ohmic Weave: Memristor-Based Threshold Gate Networks," *Computer*, vol. 48, no. 12, 2015, pp. 65–71.
3. [3] Landauer, "Irreversibility and Heat Generation in the Computing Process," *IBM J. Research and Development*, vol. 5, no. 3, 1961, pp. 183–191.
4. [4] DeBenedictis, "The Boolean Logic Tax," *Computer*, vol. 49, no. 4, 2016, pp. 79–82.
5. [5] Agarwal et al., "Energy Scaling Advantages of Resistive Memory Crossbar Based Computation and Its Application to Sparse Coding," *Frontiers in Neuroscience*, vol. 9, 2016, p. 484.
6. [6] DeBenedictis, et al., *Cognitive Computing for Security*, tech. report SAND2015-10868, Sandia Nat'l Labs, 2015; prod.sandia.gov/techlib /access-control.cgi/2015/1510868.pdf.
7. [7] Wikipedia, updated 2016; en.wikipedia.org/wiki/Computational_complexity_theory.

Erik DeBenedictis *is a technical staff member in the Center for Computing Research at Sandia National Laboratories. Contact him at epdeben@sandia.gov.*