Analog Feed Forward Neural Network Neuron Design and Implementation

Daniel J. B. Clarke
Gildart Haase School of
Computer Sciences & Engineering
Fairleigh Dickinson University
Email: danieljbclarke@gmail.com

***Abstract*—With Machine Learning methods being employed for many different use cases from speech and handwriting recognition to autonomous vehicles in big data analytics, the demand for high performance learning continues to increase. Artificial Neural Networks are widely used in software because of their theoretical simplicity and versatility for tackling many types of problems. While GPU acceleration is helping industry meet processing demand for network training, an analog integrated circuit that performs the training required by these networks may prove extremely valuable for scaling up neural network performance and bringing larger and faster networks to smaller devices.**

A single neuron of an analog feed forward neural network is designed in this paper with two input synapses and one output. The ability of the circuit to learn logic functions, AND, OR, and XOR is tested. Given the nature of artificial neural networks, such a network would have the ability to scale upwards to provide NxM-layer feed-forward neural network capabilities. Utilizing GPIO pins on a Raspberry PI, software is written to interface with the circuit and performance is benchmarked against an equivalent software-based neural network.

I. Introduction

Neural Networks model the operation of a biologic brain. Used extensively in machine learning, a simplified model of these networks, feed forward neural networks, have proven useful for a number of problems including speech and handwriting recognition. Neural networks operate like binary classifiers driven by linearly separated features across multiple dimensions. The limitation of linear separability, though initially thought to be too great to warrant any attention to the algorithm has become less of multi-layer networks. [1]

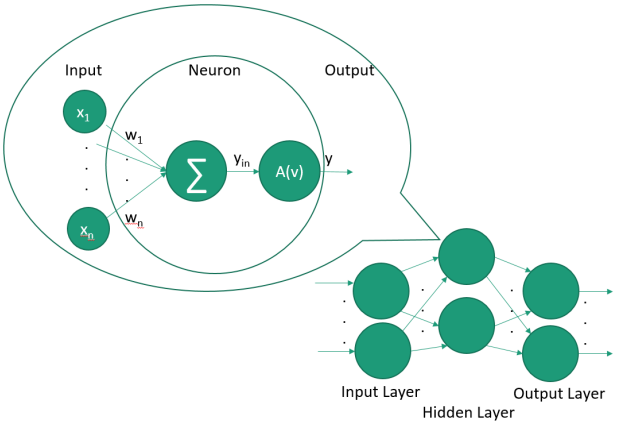

Fig. 1. Feed Forward Neural Network

The sigmoid function is often used in the field of machine learning due to its uniformity and differentiability as an activation function. The activation function receives the sum of the weights times the inputs and turns it into the output y_j. $\tanh(x)$ is a bipolar sigmoid function. It results in a training algorithm known as delta rule of the form:

$$\Delta w_{ji} = \alpha(t_j - y_j)x_i \tag{1}$$

Where Δw_{ji} is the change to be applied to a single weight in the network, α is a small learning rate, t_j is the expected output, y_j is the network output, and x_i is the input.

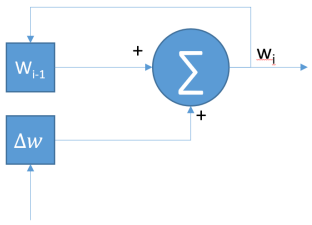

Fig. 2. Weight Circuit Block Diagram

A. Discussion

Graf et. al. suggested the construction of an analog electronic neural network circuit as far back as 1989 focusing on potential interconnected architecture. A balance needed between resolution and complexity is suggestedworrying about the limitations of electronic hardware above modeling the biological function precisely. [2] Algorithms have since been developed that are being widely used in software.

Nishitani et. al. explore back-propagation operation with analog neural networks while using FeMEM, a ferroelectric memristor, for the synaptic weights [3]. The primary issue they dealt with is the effect of hysterisis on the conductance of their memristors. A focus is placed on mitigating the effects caused by the highly non-linearity of those devices.

Rosenthal et. al. build an analog neural network utilizing a structure very similar to the one in this paper, but they also use memristors for the synaptic weights. They experienced the effects discussed in [3] but still managed to get some results. While they found ten-fold reductions in runtime benchmarked to a MATLAB based implementation, MATLAB itself is about ten times slower than a typical neural network implementation in C, these results are therefore, not too impressive.

Tensor Processing Units by Google are ASICs (application-specific integrated circuits) designed to do tensor calculus operations, used for Google TensorFlow, their framework for building and training large neural networks. They operate at lower precision to get better efficiency. While they are dedicated chips for neural network processing that acheive a ten-fold performance increase above using GPUs, they still rely on digital approaches for neural network training. [4]

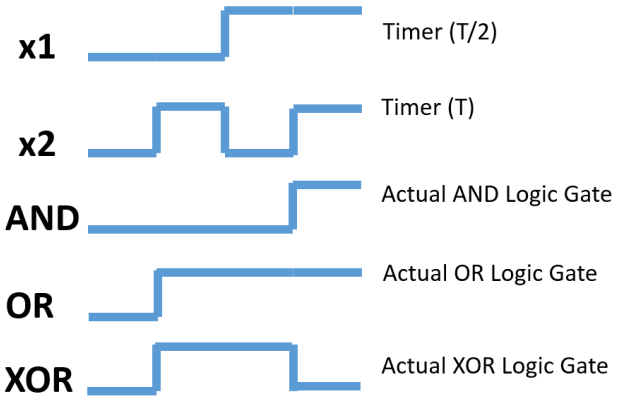

Fig. 3. Test Signal

In this paper, we construct an analog neural network and fall back to a less precise but much more predictable form of synaptic weights by charging capacitors as suggested in [2], but by using bipolar weights instead of digital memory. Using analog internally and a digital interface we achieve the power of analog learning while still retaining the ability to interface with existing digital device drivers.

II. Objectives

To validate the feasibility and performance of our approach we design and construct a single analog neuron with 2 inputs and 1 output, we test it against logic gates in hardware and benchmark it against a software based implementation.

The proper functioning of the circuit can be verified by ensuring the error tends to zero as training occurs. The following signals can be provided to the circuit for the logic functions learning.

In an effort to illustrate the circuit's use as hardware acceleration, a raspberry pi will interface with the circuit over its GPIO pins. It will drive the circuit to learn AND, OR, and XOR and will perform the same analysis in software as a comparison.

III. Design

Ultimately, neural networks are defined by a few simple equations. The ability to calculate an output given input and synaptic weights–quite simply:

$$\begin{aligned} X_0 &= \text{Inputs} \\ Y_{in,j} &= \sum_i X_{ij} W_{ij} \\ Y_j &= \tanh(Y_{in,j}) = X_{j+1} \\ Y_N &= \text{Output} \\ E_i &= T_i - Y_i \end{aligned}$$

The ability for the network to actually learn is fundamental for a neural network circuit to be useful. While weights can ultimately be solved mathematically as an optimization problem, gradient descent for instance, the actual function is not often known. This is where neural networks becomes useful–using the derivative of the hyperbolic tangent error function. This approach is known as back propagation. It can be shown that an iterative approach will follow the following relationship:

$$\delta w_i = \alpha(T_i - Y_i)X_i$$

We design an analog circuit which can perform this learning in real-time.

A. Binary Product

The Binary product is essentially an XOR gate, but as an Analog voltage multiplied by a binary one things become a little more complicated. The anticipated output function looks like so:

V	X	V*X
$V<0$	0	$-V$
$V<0$	V_{cc}	V
$V>0$	0	$-V$
$V>0$	V_{cc}	V

In other words, when X is 0, we want an inverter, else we want a voltage follower. We consider the difference between the two terminals of a MOSFET. Assuming negligible current gain, when the transistor is ON, $V_D = V \approx V_S$ while when the transistor is OFF, $V_D = V, V_S = 0$. So using a differential amplifier we can set it up such that $V_o = 2V_S - V_D$ to achieve our desired operation.

The general equation of any differential amplifier is

$$V_o = V_2\frac{(R_3+R_1)R_4}{(R_4+R_2)R_1} - V_1\frac{R_3}{R_1}$$

Therefore we need

$$2A = \frac{(R_3+R_1)R_4}{(R_4+R_2)R_1}, A = \frac{R_3}{R_1}$$

Where A is the difference gain we want. Let R be some base relative resistance.

$$AR_1 = R_3 = R$$

$$\begin{aligned} 2A &= \frac{(R_3+R_1)R_4}{(R_4+R_2)R_1} \\ 2A &= \frac{(A+1)RR_4}{(R_4+R_2)AR} \\ 2A &= \left.\frac{(A+1)R_4}{(R_4+R_2)A}\right|_{A=1} \\ 2 &= \frac{2R_4}{R_4+R_2} \\ R_4+R_2 = R_4 &\rightarrow R_2 = 0, R_4 = R \end{aligned}$$

We can't achieve a better gain than 1 but with $R_2 << R_4$ we can get near 1, for this reason we use a short circuit. As R_4 doesn't matter, we'll just use 220Ω as is used throughout the rest of the circuit because at 3V it gives us 14mA, a good current for operating the transistors throughout the circuit. $R_1 = R_3 = R_4 = 220\Omega$

B. Synaptic Weight Control

Storing the value of w_{ij} as V_c due to charge in a capacitor, the back-propagation equation can charge or discharge the capacitor until the error becomes zero according to the delta rule. α is dictated by the natural time constant $\tau = RC$ of the capacitor-resistor circuit. While this τ value is related to the learning rate, we

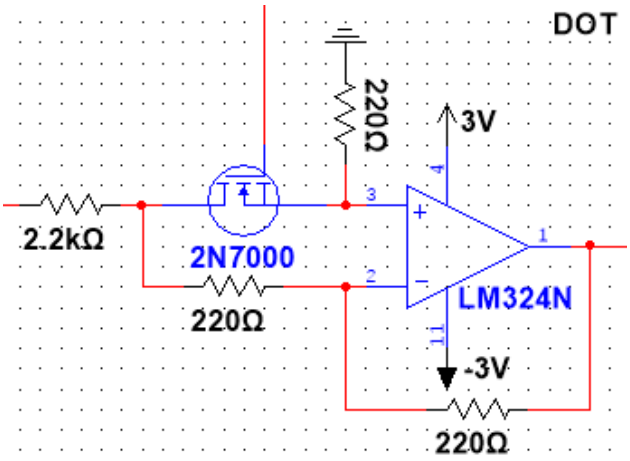

Fig. 4. Bipolar Binary Multiplication

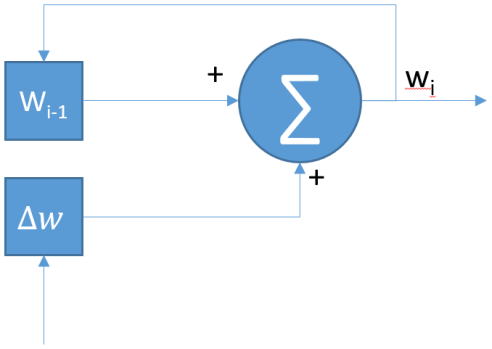

Fig. 5. Weight Control Block Diagram

leave its derivation for future direction. For now sane values that seem to work in a range we can analyze the circuit have been chosen. $C = 1\mu F, R = 220\Omega, \tau = RC \approx 0.22ms$ Because the capacitors train in parallel, it's expected that this time should be a **ceiling** for training time.

Other than the capacitor, we need to keep it from draining charge to fast so it doesn't forget what it learns. The theoretical block diagram is shown below.

Keeping to the theory, we're using a summation amplifier circuit–but we connect a capacitor as forward feedback. The old value of the weight gets put back into the circuit to keep the capacitor charged with itself but the calculated error adjustment is added on top of that.

Despite the leakage due to the input into the

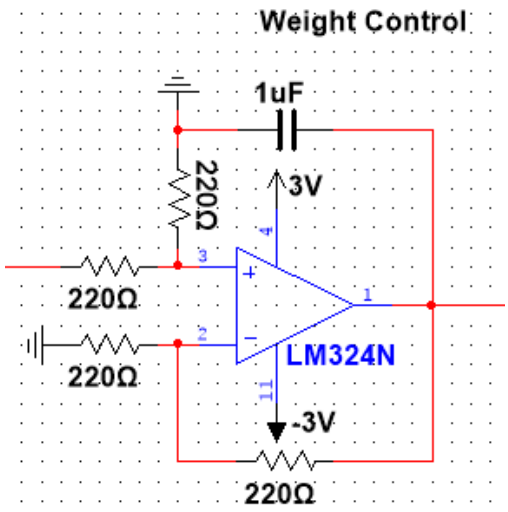

Fig. 6. Weight Control Circuit

op amp not truly being 0 amps, the charge in the capacitor stays relatively stable in operation, certainly stable enough to learn properly. This approach limits the circuit to being able to learn but not really able to reliably retain its knowledge; this can still prove valuable if the analog values of the weights are extracted–the weights provide the solution to the problem that was being solved in order of $O(n)$ where n is the number of weights; because then it's just a dot product of the weights with the input to get the output.

C. Summation and Activation

The sum of the weight-input products is passed through a hyperbolic tangent activation function to give us a differentiable output that can be used for training via hebb rule. While a diode clipper can be used to constrain the circuit between $\pm V_{on,diode}$, the output of an op amp is already clipped at $\pm V_{sat}$ and has a natural tendency to behave more like a hyperbolic tangent function as opposed to being perfectly linear. For this reason, no extra effort is taken beyond the summation–this seems to work just fine given the differentiable nature of the universe.

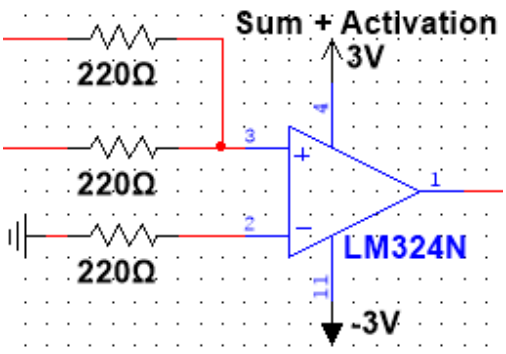

Fig. 7. Summation and Activation Circuit

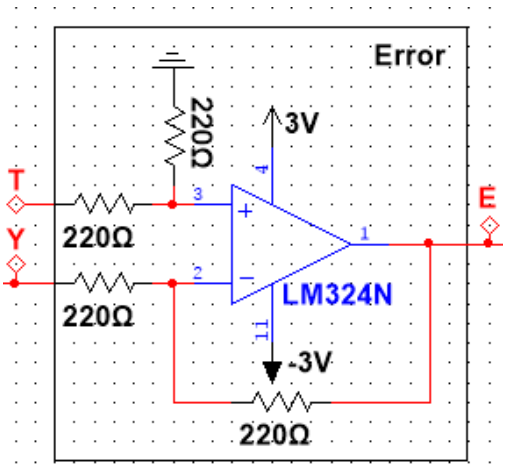

Fig. 8. Summation and Activation Circuit

D. Error Calculation

The error of the circuit is quite simply $T-Y$, but with a bipolar Y and a binary T it's actually necessary to pass T through a zero-crossing detector to get output in the same range as Y. This is not performed here because it's easy enough to set out power supply accordingly but would be necessary in a production circuit.

E. Full Circuit

IV. Testing

The network can be tested by providing inputs X_i and targets T_i. This is provided with circuit elements like buttons, timers, or simply a signal generator. In the future it is expected that software will interface with the circuit.

Because a software-driven element has yet to be created, the validity of the circuit operation can be assessed by setting up a signal generator and watching that the error tends to decrease over time and the weights converge given that the circuit can learn what is being taught (e.g. OR/AND not XOR).

V. Costs

For each synapse, LM324N $\times$ 0.75, 2N7000 $\times$ $2, 220\Omega$ $\times$ $10, 2.2k\Omega$ $\times$ $2Resitors, 1\mu FCapacitor$ (0.75 $LM324N$ because these are Quad Op Amps and 3 are required)

For each neuron, N synapses $+ 220\Omega \times 7 +$ LM324N $\times$ 0.5 (0.5 $LM324N$ because these are Quad Op Amps and 2 are required)

1) OP LM324N $0.40 each
2) NMOS 2N7000 $0.40 each
3) Resistor $220\Omega, 2.2k\Omega$ $0.05 each
4) Capacitor $1\mu F$ $0.40 each

Totals for our single neuron with two inputs: $4.75/neuron.

VI. Results

Because the circuit is able to adjust weights and get an output in the same direction as the output we train towards, it is able to learn properly. That-is when we press the T button to make the target +1, the network adjust itself so that a +1 output is returned, when we let go of the button making that target -1, the network adjusts itself so that a -1 output is returned. This occurs regardless of the state of the two inputs because the weights, stored in the capacitors, are adjusted to compensate for those inputs.

Because of the tao constant, watching the training on an oscilloscope requires a frequency high enough that the network doesn't forget the data but also slow enough to not just look like a voltage follower. In these graphs, the middle smaller voltage is the error while the higher values are the training data.

While the error doesn't converge to zero due to inaccuracies accumulating throughout the

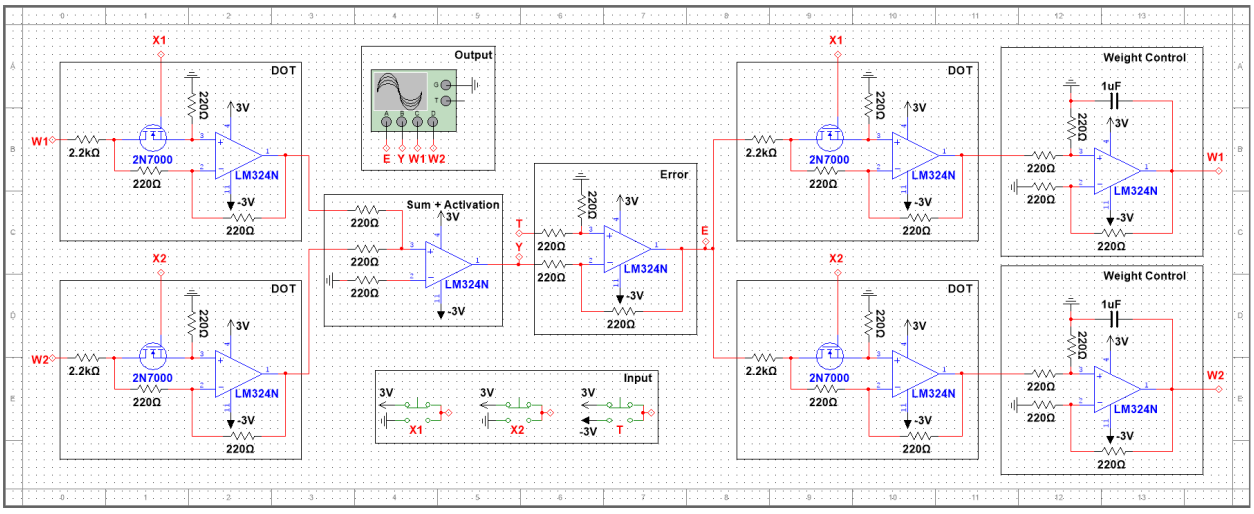

Fig. 9. Full circuit

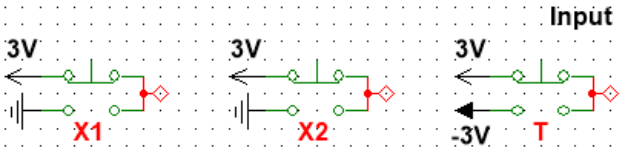

Fig. 10. Input controls

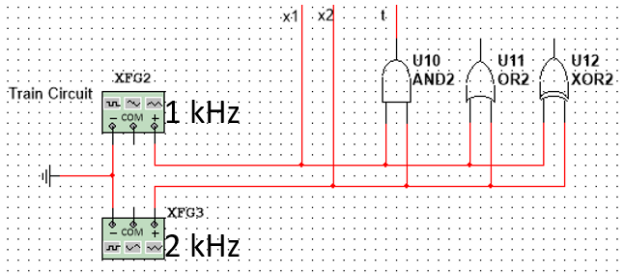

Fig. 11. Test Circuit

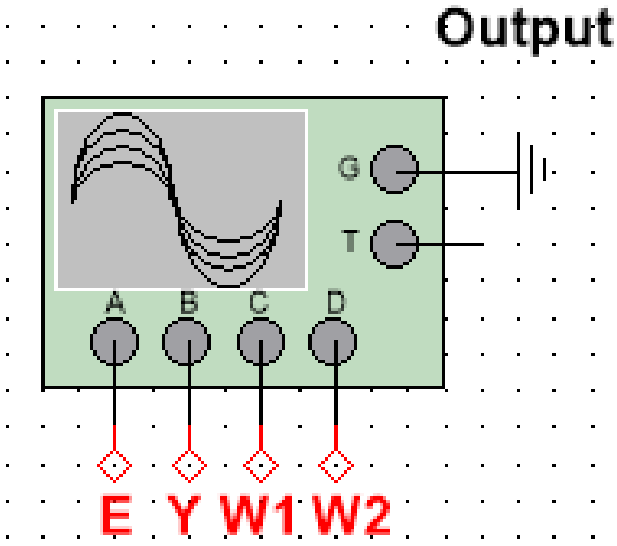

Fig. 12. Outputs of the network

circuit around 0 as well as an apparent oscillation at the solution caused by a learning rate that is too high (capacitor not operating slow enough); the circuit does, however, successfully learn the input, and quite quickly as well.

VII. Future Direction

With more time and money, a memristor model might be evaluated and benchmarked against this capacitor based approach to know for sure whether or not it is truly a better approach. Some feasibility tests would also be evaluated.

While we have decent results from a signal generator–whether or not the circuit can properly and quickly be driven by a computer is an essential question to answer. Driving the circuit via a raspberry pi's GPIO pins would be used.

A. Issues

We do successfully train, but the forgetfulness of our circuit is concerning and should

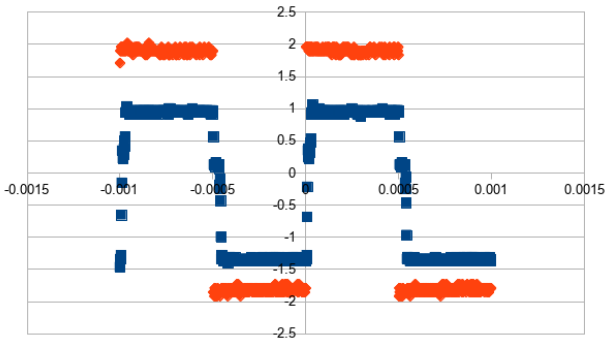

Fig. 13. Training too slow, can't learn

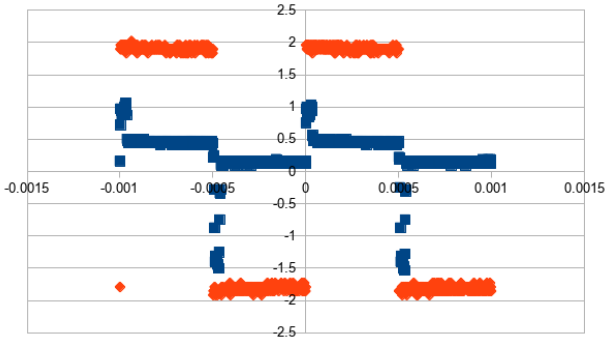

Fig. 14. Training fast, looks like a bad voltage follower

be further investigated. The error rate doesn't truly make it to 0, this may be a manifestation of the errors introduced by the 2N7000's gain and the inaccuracies of OP amps as opposed to instrumentation amplifiers.

Training with more than one neuron is also an essential aspect before truly proving scalability of the approach. Further complications

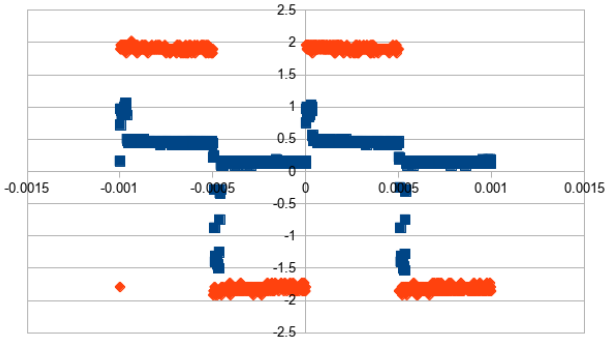

Fig. 15. Looks good, error converges

are expected due to propagation delays of larger circuits.

VIII. Conclusion

A novel approach for analog neural network construction is designed and partially implemented. With individual modules working, and promising full circuit results despite apparent convergence issues, there is more to be done before we can completely say this will work. Falling back to a capacitor based analog memory approach as in [2], our approach depends only on components that can be easily aquired as opposed to costly memristor-style memory components so prevalent in current research [3], [5]. The approach taken is inherently scalable–this scalability will be demonstrated in the future with the construction of a 3 neuron network that can properly learn XOR.

References

[1] P. Auer, H. Burgsteiner, and W. Maass, "A learning rule for very simple universal approximators consisting of a single layer of perceptrons," *Neural Networks*, vol. 21, no. 5, pp. 786 – 795, 2008.

[2] H. P. Graf and L. D. Jackel, "Analog electronic neural network circuits," *IEEE Circuits and Devices Magazine*, vol. 5, pp. 44–49, July 1989.

[3] M. Ueda, Y. Nishitani, Y. Kaneko, and A. Omote, "Back-propagation operation for analog neural network hardware with synapse components having hysteresis characteristics," *PLOS ONE*, vol. 9, pp. 1–10, 11 2014.

[4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, (Berkeley, CA, USA), pp. 265–283, USENIX Association, 2016.

[5] E. Rosenthal, S. Greshnikov, D. Soudry, and S. Kvatinsky, "A fully analog memristor-based neural network with online gradient training," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1394–1397, May 2016.