TACOCAT: Trainable Acceleration of Classification Operations via Commonly Available Technology

Luke J. Minks, Deven J. Morone, German Romero Castro, and Justin D. Sapp
Department of Electrical and Computer Engineering
University of Central Florida, Orlando, FL 32816-2362

***Abstract*—Recent hardware accelerator designs for machine-learning show great promise for allowing complex image recognition tasks to be carried out on resource-constrained platforms. However, the memristor crossbar arrays that many designs rely on are not currently marketed by any commercial manufacturers and must instead be custom-fabricated. The work in this paper was motivated by a desire to build a small, hardware-based neural network that could provide an example for the construction of a physical proof-of-concept device to complement simulated test results for novel neuromorphic circuit designs in cases where custom VLSI design and fabrication are not practicable. Designs based on commercially-available components are proposed for an artificial neuron and synapses, and these designs are employed in the construction of a hardware-based multilayer perceptron that is successfully trained to classify 100-pixel images in handwritten-character datasets containing three different image classes.**

***Index Terms*—Machine learning, neural network, multilayer perceptron, hardware accelerator, image recognition.**

I. Introduction

In the last 20 years, machine learning technology has made impressive advances, and current machine learning technology can perform sophisticated classification and optimization tasks. However, many machine-learning models are implemented using software packages with high computational demands that are designed to be run on centralized servers that communicate with users' mobile devices and small PCs. Hardware-based acceleration for neuromorphic computing, based on emerging non-volatile memory technologies, could offer dramatic increases in speed and energy efficiency. If realized, this technology would allow complex machine learning tasks to be carried out on resource-constrained edge devices. In light of the potential benefits of hardware-based neural networks and the recent progress in the development of the novel technologies that are needed to produce these networks on a large scale, it seems likely that hardware-based neural networks will emerge as a viable technology within the next decade. While the non-volatile variable resistance devices that will most likely be needed in order to implement complex, highly-versatile, hardware-based neural networks are not commercially available, there are substitutes for these devices that could be used to produce a smaller-scale prototype for a hardware-based neural network. The Trainable Acceleration of Classification Operations via Commonly Available Technology (TACOCAT) project was inspired by the idea that a group of undergraduate students could implement a sophisticated hardware-based neural network prototype using off-the-shelf components, which would serve as a proof-of-concept for hardware-based neural network designs and also provide valuable experience in an up-and-coming field of research and development. TACOCAT is able to employ a surprisingly small number of artificial neurons and synaptic connections in order to perform somewhat specialized, yet highly complex recognition tasks, and will hopefully contribute in some way towards the development of efficient, secure data processing on resource-constrained devices.

II. System Overview

TACOCAT's multilayer perceptron architecture, depicted in Fig. 1, consists of 100 input neurons, 3 hidden-layer neurons, and 3 output neurons, which are fully connected via 309 hardware synapses. These neuromorphic circuits are implemented using a modular design consisting of one motherboard and three identical daughterboards.

Each daughterboard represents the 100-synapse array belonging to a single hidden-layer neuron. Weight adjustment and input signals are received from the motherboard, and the digital potentiometer and operational amplifier components on the daughterboard are arranged to simultaneously multiply 100 different pairs of input and weight values. The sum of these weighted values is calculated using a two-level analog adder tree consisting of ten-input summing amplifiers.

The motherboard is responsible for programming weight values to digital potentiometers, applying pixel input voltages to the daughterboard inputs, implementing neuron summing and activation functions, and providing accurate readings of neuron output voltages. The motherboard's control interface is provided by a 32-bit microcontroller that receives instructions from a PC-based software package via an I2C data link. A functional overview of the classification process is shown in Fig. 1, and a detailed system diagram is shown in Fig. 2. The hardware network runs on a dual-rail power supply, with a Vss equal to -1.65V and Vdd equal to 1.65V.

During training, the PC-based software package calculates ideal synaptic weights as floating-point decimal values, and these weights are converted into byte values ranging from 0 to 255 so that they can be programmed into the digital potentiometers in the hardware network. The software package takes input images from the Extended-MNIST (EMNIST) dataset [1] and converts them into 10x10-pixel black-and-white images, which are transmitted to the motherboard. The motherboard uses dedicated microcontroller output pins and an array of digital buffer ICs to apply corresponding voltage

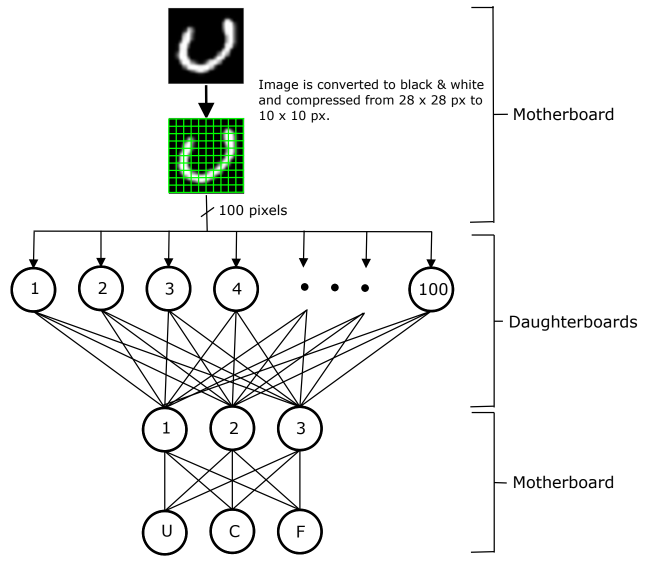


Image is converted to black & white and compressed from 28 x 28 px to 10 x 10 px.
Motherboard
100 pixels
1
2
3
4
100
Daughterboards
1
2
3
Motherboard
U
C
F

Fig. 1. Overview of the neural network's multilayer perceptron architecture. Each circular node represents a neuron, and each connecting line represents a synapse. The network as shown is trained to recognize the letters 'U', 'C', and 'F'.

levels to the daughterboard input lines, with black pixels represented by 0V and white pixels represented by 1.65V. The microcontroller on the motherboard then takes ADC voltage readings from all neuron outputs and transmits them to the PC, which uses these values to calculate updated synaptic weights, and this process is repeated until a specified minimum accuracy rate is achieved or a pre-determined number of training cycles has elapsed.

Prediction operations are similar to training, except weights are not updated, and only the output-layer neurons' voltages need to be measured. The output-layer neuron with the highest voltage level indicates the neural network's predicted classification for the input image.

III. System Components

This section is comprised of brief descriptions of key components used in the construction of the TACOCAT device.

A. Microcontroller

All hardware operations of the TACOCAT motherboard and daughterboards are controlled by an STM32H743 microcontroller, which contains an ARM Cortex-M7 core. This microcontroller was chosen because of the suitability of its general purpose input/output (GPIO) pin count, core clock speed, and physical package type for controlling the TACOCAT network. With up to 140 available GPIO pins, the STM32H743 has enough pins to drive 100 pixel input lines and still have pins available for communication and ADC functions. Its maximum clock rate of 480 MHz is fast enough to ensure minimal latency for training and recognition options.

B. Digital Potentiometers

In the TACOCAT network, synaptic weights are implemented using digital potentiometers. The digitally-programmable variable resistance levels provided by these devices allow for a reliable and effective way to train the network using control signals provided from the MCU. When choosing an appropriate digital potentiometer, the most important factors to consider were the communication protocol, the number of potentiometers per chip, and the number of taps per potentiometer.

The Analog Devices AD5204 digital potentiometer was chosen as it provides 4 programmable potentiometers per chip and supports daisy-chained SPI communication, which both facilitated a simpler PCB design process. The AD5204 also has 256 taps per digital potentiometer, which met the requirements for digital potentiometer resolution that were determined from SPICE simulations.

C. Operational Amplifiers

Summing amplifiers with carefully-determined gain values and intentionally non-linear responses are used to achieve the necessary multiplication and summing operations of the artificial neurons in the TACOCAT network, and the main component of each summing amplifier is an operational amplifier stage.

The Microchip MCP6274 quad operational amplifier chip was chosen for the implementation of both the summing/activation amplifiers found in the neuron circuit and for the inverting signal buffers at the beginning of each synapse path. The MCP6274 has high input impedance, which enables reliable differential and common mode operations, and its very low input bias current of 1 pA helps to maintain a reliable output signal with low offset voltages.

D. Octal Buffer

The machine-learning algorithm that is used to train the TACOCAT network is based on ideal pixel input values of 0.0 and 1.0, and its ideal neuron outputs range from -1.0 to 1.0. These properties dictate that if an ideal value of 1.0 is represented by Vdd, and -1.0 is represented by Vss, then an ideal value of 0.0 must be represented by a voltage that is precisely half-way between Vdd and Vss, which equates to 0 V in the TACOCAT hardware network. The STM32 microcontroller is operated at Vss to Vdd (-1.65V to 1.65V) supply levels in order to simplify communication with the digital potentiometers and maintain the ability to measure all voltages with the onboard ADC peripherals, but this creates the complication of producing pixel-input voltage levels that do not correspond to either of the STM32's digital output levels. While the STM32H743 has multiple DAC channels, it does not have enough analog output lines to produce 100 pixel-input voltage levels simultaneously.

We solved this problem by placing a buffer circuit between each input driver pin on the STM32 and its corresponding input pins on the daughterboards. By running a digital buffer IC at 0V to 1.65V supply levels and adding a resistor between

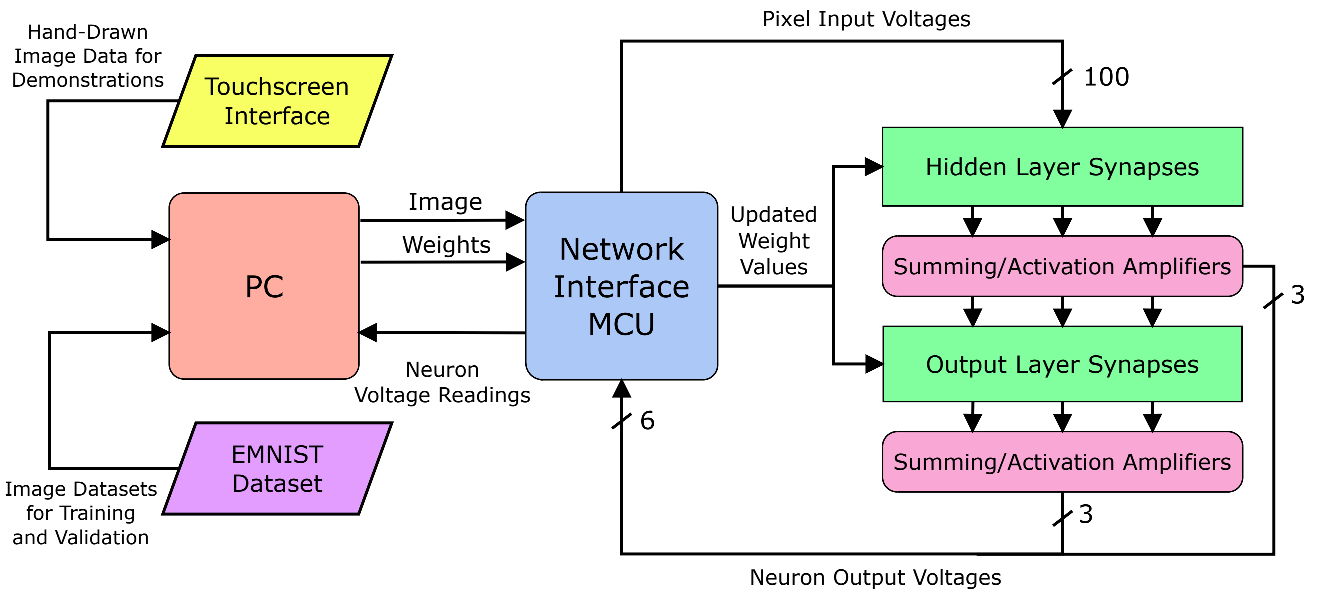


Fig. 2. System Block Diagram.

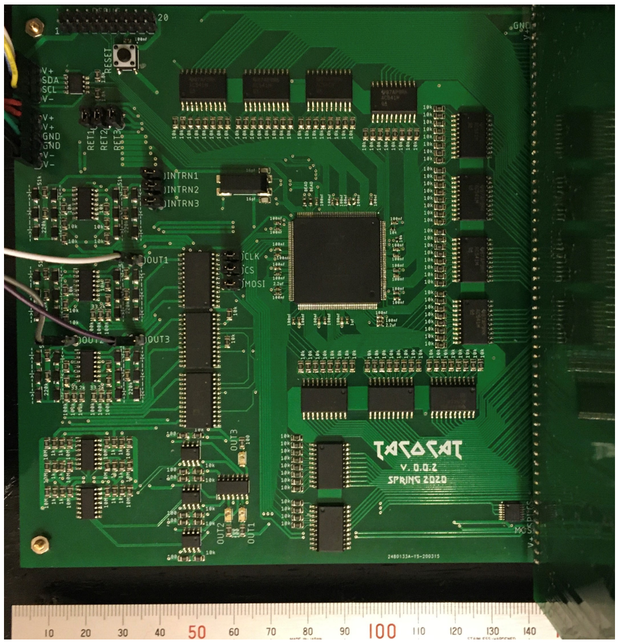


Fig. 3. Overhead view of the motherboard. Daughterboards are mounted on the right-hand side of the motherboard.

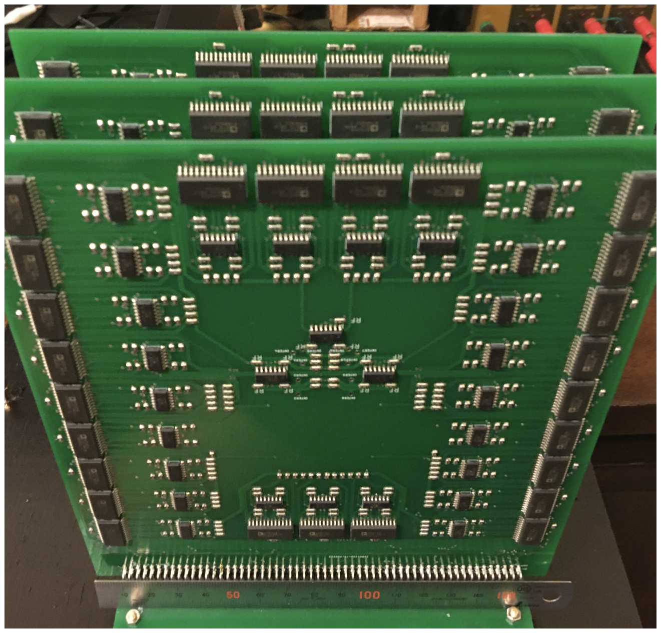


Fig. 4. Three vertically-mounted daughterboards. Each daughterboard is connected to the motherboard using 116-pin headers and sockets in a double-row configuration.

the 0V supply rail and the buffer input, we can produce a 0V buffer output level by setting the STM32 pin to Hi-Z input mode. For a drive value of Vdd, the STM32 driver pin is reconfigured as a digital output with a high value.

We chose the Texas Instruments AC541M octal buffer IC for this purpose. With the AC541M's 8 buffers per chip, 100 buffers can be obtained using a reasonable number of ICs. Also, the AC541M is capable of normal operation with supply voltage rails at 0V and 1.65V.

IV. Hardware Design

The following subsections briefly explain the design process for the TACOCAT neural network hardware.

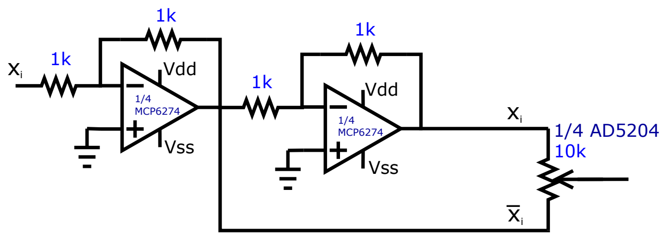


Fig. 5. Synapse circuit design.

A. *Synapse Circuit Design*

The synapse circuit, pictured in Fig. 5, is the first circuit encountered when following the signal path in an individual neuron circuit. At the beginning of the signal path, the synapse circuit applies the input signal to an inverting buffer stage, which feeds into an additional inverting buffer stage, creating a buffered pair of complementary (inverted and non-inverted) signals that are applied to the legs of the synapse's digital potentiometer.

The synaptic weight value, determined by the software-based training algorithm, is applied by adjusting the potentiometer's wiper position, creating an adjustable voltage divider between the inverted and non-inverted input voltages. The weighted input value is represented by the voltage on the potentiometer's wiper pin. This configuration allows the input signal voltage to be multiplied by any value between -1 and 1, with precision limited by the number of taps on the device.

Since the input is isolated from the neuron and synapse hardware via a pair of inverting buffers, the input signals are protected against excess current draw, which could cause distorted outputs or damage hardware in adjacent network layers and even lead to cascading failures in extreme cases.

The synapse circuit described above should provide a minimal component count while maintaining safeguards that protect attached circuits. A total of only four discrete components is necessary to buffer a single input and protect components in adjacent layers. Other implementations that were considered as alternatives either lacked input safeguards or introduced redundant sources of error to the network.

B. *Neuron Circuit Design*

In a multilayer perceptron, the artificial neuron takes the sum of the weighted input values that are received from its synapses and applies an activation function to that sum. In the TACOCAT network, the artificial neuron circuit is designed to implement weighted-input summing along with a hyperbolic tangent activation function.

For the hidden layer of the network, the hundred weighted input voltages are summed using a two-level tree of summing amplifiers comprised of ten 10-input summing amplifiers that feed into an additional 10-input summing amplifier, which produces the weighted input sum. This approach was chosen to avoid the challenges and potential sources of error involved with summing one hundred inputs using a single amplifier.

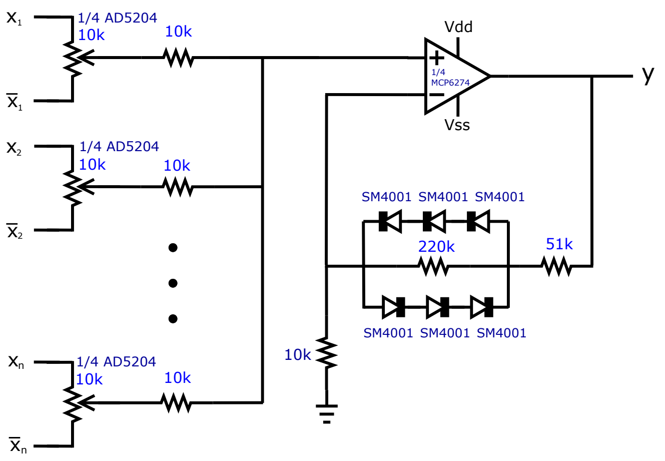


Fig. 6. Neuron circuit design.

Each summing amplifier in the tree is designed to have a gain of -.1 with respect to each of its inputs.

As the hidden layer of the network is only fed forward to three output neurons, this multi-stage tree approach is not required when summing the hidden layer outputs. However, the presence of a single inverting amplifier stage resulted in the neuron output of the hidden layer becoming inverted. This issue was addressed by inverting the connections to the potentiometers on the output layer's input synapses. Using this approach, it is possible to produce an overall gain of 1/n for each input to a given neuron, with n being the total number of inputs. With all potentiometer wipers set to their maximum values, this results in a unity-gain linear output response when. The linear responses of the daughterboards' 100-input adding trees can be seen in the plot of output voltage ADC readings vs. daughterboard input activation provided in Fig. 7.

To function properly, however, the neurons must have some non-linearity in their activation response. In a fashion similar to the artificial neuron design used in [2], a portion of the feedback circuit resistance for the neuron's amplifier stage (pictured in Fig. 6 is shunted using two opposing groups of series-connected diodes. Three SM4001 diodes are used in each series-connected group to create an output response that transitions from a linear region centered at 0V to a gradually increasing amount of compression which leads to clipping as the output voltage approaches either +1.2V or -1.2V. As seen in Fig. 8, the output waveform's characteristics are similar to the hyperbolic tangent function. By implementing summing and activation functions using a single amplifier stage, the TACOCAT neuron circuit provides appropriate transfer characteristics while maintaining a low discrete component count and minimal footprint.

V. Software Design

The software/firmware suite for the TACOCAT network is composed of three main components. A software-based machine-learning model handles data-loading and network training operations, and a separate firmware package controls

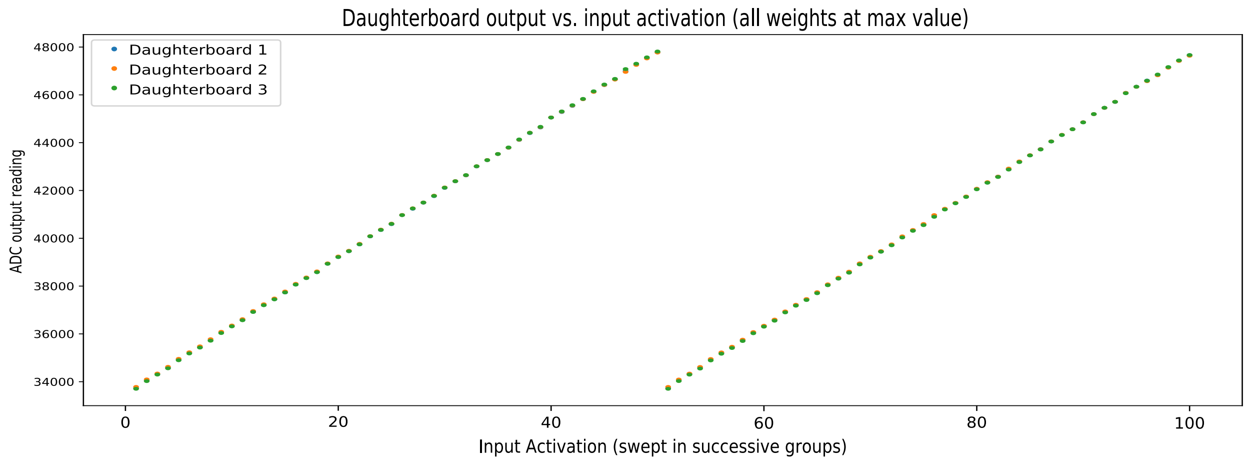
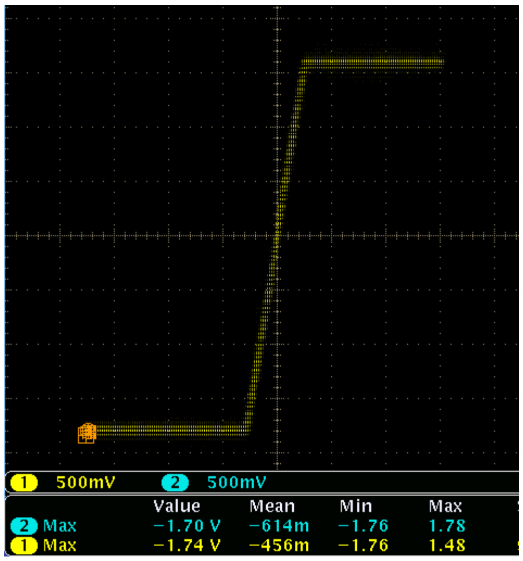


Fig. 7. Daughterboard summing amplifier test results.



```
 1   500mV     2   500mV
            Value     Mean     Min     Max
 2 Max     −1.70 V   −614m    −1.76    1.78
 1 Max     −1.74 V   −456m    −1.76    1.48
```

Fig. 8. Oscilloscope measurement of neuron activation function output voltage vs. input voltage.

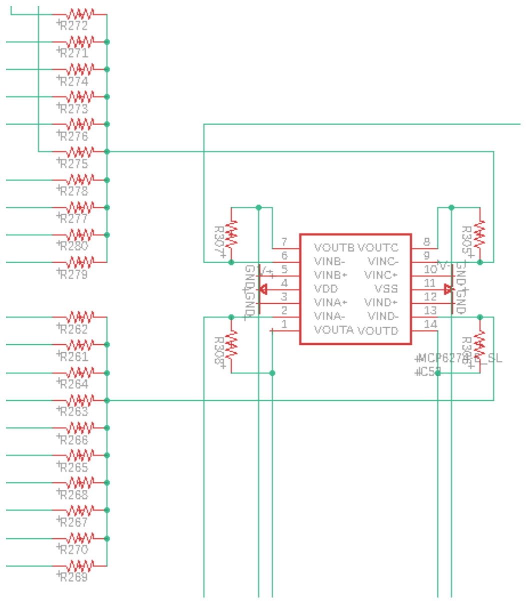


Fig. 9. Two of the 10-input summing amplifiers used in the daughterboard design.

the TACOCAT hardware and runs on the ARM microcontroller. Also, a graphical user interface module was implemented for public demonstration purposed. A diagram of the system is shown in Fig. 10.

A. Machine-learning Model

Instead of taking the very-high-level approach of using a ready-made machine-learning framework, we decided to compromise by writing our own customized model in Python, which is a high-level language that offers powerful libraries for tensor mathematics and data manipulation. At this level of abstraction, all mathematical operations for the software model had to be expressed in code, but the code itself is fairly brief and easy to understand.

The machine-learning model's software was divided into several Python modules: a dataloader module for loading training/validation data, a SPICE extension module for simulating neural network "think" operations in hardware that is interchangeable with a separate hardware-network communication module, and a core module that contains the construction, training, and validation logic for the neural network.

B. Dataloader

The dataloader module uses the Mlxtend library [3] to load training/validation data from files that adhere to the formatting used in the MNIST dataset. This module also performs interpolation operations to compress data images from the standard size of 28x28 pixels to a square size of a smaller specified width, and it performs a thresholding operation to convert pixels from grayscale to black and white, as shown in 11. It should be noted that this thresholding operation is implementing an input-layer activation function outside of the hardware network. This was considered as a possible area

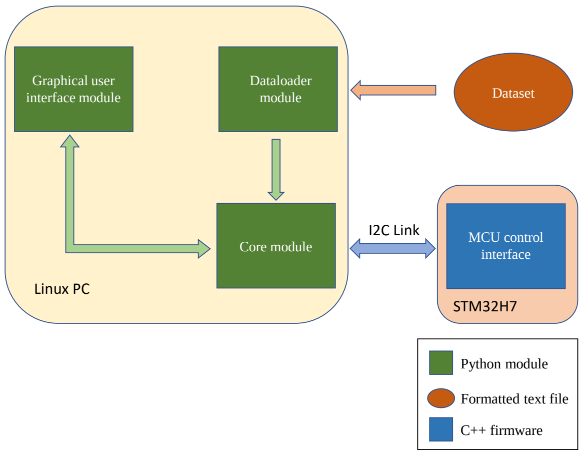


Fig. 10. Software/firmware block diagram.

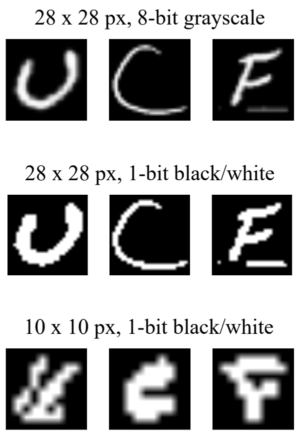

28 x 28 px, 8-bit grayscale
28 x 28 px, 1-bit black/white
10 x 10 px, 1-bit black/white

Fig. 11. EMNIST image downsampling example. Downsampling operations are performed in the software dataloader module.

of concern, as one of the main objectives of this project is to build a hardware-based neural network. However, because the equivalent activation function for the input layer would be a unit step function, the software operation to calculate the function's output is a simple comparison between the input value and the threshold. This function could also be implemented in hardware using a circuit as simple as a single transistor. If high accuracy were required, comparator circuits could also be used to implement the comparison operation in hardware.

C. *Simulation of Network Training*

Prior to the construction of the TACOCAT hardware network, simulations were conducted for a small-scale 4-pixel network using LTspice, which is a SPICE simulation software package provided free-of-charge by Analog Devices, Inc. In order to integrate SPICE simulations with the software simulation model, a Python module was created to generate SPICE netlists based on the ideal weight and input values described by the state of the software-model's network. The same module then runs LTspice simulations of these netlists and parses the output from LTspice into a data structure that can be returned to the core module. The core module then updates the state of the software-model network based on the simulation outputs, and the next training cycle begins.

By updating the SPICE hardware model's netlist after each training iteration and using the simulated output values of the hardware neuron models based on the updated weight values, the hardware network training is effectively emulated in the software simulation. The SPICE connection module was also programmed to consider parameters for digital-pot weight resolution and digital-pot tolerance. By conducting simulations with various digital-pot parameters, it is possible to estimate the levels of tolerance and resolution that are required to achieve convergence and reasonably low error rates in the neural-network training process. Simulation results indicated that with 7-bit pot resolution, the network training process might fail to converge to an optimized set of weight values. Training simulations for 8-bit potentiometers achieved convergence with good accuracy results.

D. *Core*

The core module contains definitions for a `NeuronLayer` class, which contains member variables that store metadata describing the neuron layer's number of synaptic inputs and its neuron count along with a reference to a stored array of synaptic input weights. The module also contains a `NeuralNetwork` class, which hold a collection of references to `NeuronLayer` objects along with methods for training and validating the neural network model and getting prediction results. This module also maintains collections of measurement data related to training, testing, and validation, and it includes methods that create graphs and charts to visualize this data.

The core module was designed to separate the implementation of the network components in the `NeuronLayer` class from the training and validation algorithms that belong to the `NeuralNetwork` class. By encapsulating the properties of these separate classes, the core module remains flexible enough to work with multiple different `NeuronLayer` implementations. In fact, the same training and validation algorithms are used for software-model based simulations, SPICE hardware-model based simulations, and the actual hardware implementation of the neural network (connected via I2C link to the MCU).

Python offers a broad range of interfacing libraries that also make this flexibility easy to maintain. Python's libraries for issuing operating-system commands and parsing text-file based inputs and outputs simplified the implementation of the SPICE-model connection, and Python also has a number of libraries that allow communication using lower-level protocols that are commonly supported by integrated circuits, such as I2C. Python's `pickle` module is used to save a serialized version of a trained neural network so that it can be reloaded at a later time and used for predictions or further training and data collection.

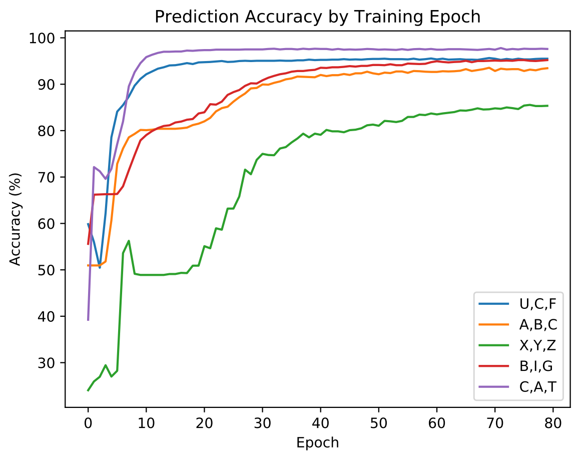


Fig. 12. Estimated prediction accuracy by training epoch for each dataset.

TABLE I
ACCURACY RATES FOR DIFFERENT DATASETS TRAINED USING IDENTICAL HYPERPARAMETERS.

Character Dataset	Prediction Accuracy (%)
UCF	95.17
ABC	92.78
XYZ	85.01
BIG	95.48
CAT	97.44

VI. Evaluation of Finished Prototype

A. Accuracy

The TACOCAT hardware network was trained to recognize the main target dataset that includes the letters 'U', 'C', and 'F', and it was also trained to recognize several other three-character datasets with the goal of determining the network's accuracy in a more general scope. The network was trained using mini-batches of 100 data images at a time with weight adjustments between each mini-batch. Starting weight values were the same for each training session, and weights were normalized, using a linear scaling method, to fall within a minimum/maximum range after each backpropagation adjustment.

Fig. 12 shows the training history for several different datasets that were each trained for 80 epochs using identical starting weights and hyperparameters. After 80 epochs of training were completed, each network was validated using a dataset composed of all of the EMNIST test data images that were available for each character (separate, mutually exclusive, sets are provided for training and for testing). Final validation results are shown in Table I.

B. User Interface

For demonstration of the hardware's functionality, users are able to enter handwritten letters via the touchscreen interface, shown in Fig. 13. The demonstration program allows users to choose from a list of synaptic weight values for any combination of three different handwritten characters that the system has previously been trained for. After the user selects a pre-trained network, they can draw their character input on the left side of the touchscreen and press the "OK" button below the input area, and the GUI shows both the downsampled version of their input image and the hardware network's neuron output voltages for each character class on the right hand side. The character class with the highest voltage is highlighted to indicate the network's prediction.

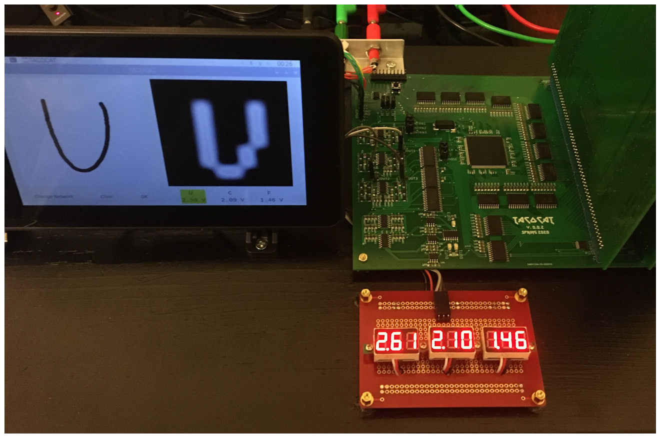


Fig. 13. Demonstration program showing prediction result for handwritten character from touchscreen input (left of motherboard). ADC voltage readings are shown for each character on the graphical display, and digital voltmeter readings taken directly from the circuit are shown on LED displays below the motherboard. From left to right, neuron output voltages correspond to the letters 'U', 'C', and 'F'.

C. Conformance to Requirement Specifications

The hardware neural network's operating characteristics were measured and compared to requirement specifications that were determined at the beginning of the design phase. Average power was calculated from supply voltage and current draw values that were measured during the network training process. Accuracy for the 'U', 'C', 'F' dataset was measured by running the complete set of EMNIST test images available for those classes (5,181 images in total) through the hardware network's recognition process. Throughput and latency were both measured by using the `timeit` module in Python 3.7 to determine the time required to complete repeated recognition operations. These throughput and latency values are conservative estimates of the hardware network's actual performance that also include processing latency in the PC, I2C communication latency between the PC and microcontroller, and the delay required for sequentially processed input-driver pin setup and ADC measurements in the microcontroller.

The TACOCAT hardware network met or exceeded all of these requirement specifications, as shown in Table II.

VII. Conclusion

In this work, artificial synapse and neuron designs were proposed. These designs were tested using SPICE simulations before being implemented in a hardware-based neural network

TABLE II
COMPARISON OF REQUIREMENT SPECIFICATIONS AND MEASURED PERFORMANCE.

Parameter	Unit	Requirement	Actual
Weight	kg	< 2.00	1.14
Footprint	m^2	< 1.00	0.13
Power Consumption	W	< 2.00	1.14
Latency	ms	< 10.00	1.14
Throughput	ops/s	> 100.00	836.00
Accuracy	%	> 75.00	95.17

using a multilayer perceptron architecture. For the primary dataset of interest, the network achieved 95.17% prediction accuracy after 80 training epochs.

Plans for further research include the use of the TACOCAT network hardware to simulate the effects of varying levels of process variation in memristor-based synaptic weight arrays. TACOCAT could also be used to simulate memristor stuck-at faults that might occur while the network is deployed in the field and prevent proper weight programming.

ACKNOWLEDGMENT

The authors wish to acknowledge the generous assistance and support of Dr. Chung Yong Chan. He is recognized for his excellent mentorship throughout the development of this project. His passion for teaching and guiding students to understand important concepts in the field of Electrical and Computer Engineering is to be admired, respected, and appreciated.

AUTHORS

Luke J. Minks is a baccalaureate student in electrical engineering at the University of Central Florida. His research interests are in analog and mixed-signal circuit design. Contact him at lminks@knights.ucf.edu.

Deven J. Morone is a baccalaureate student in electrical engineering at the University of Central Florida. His research interests are in analog and mixed-signal circuit design. Contact him at deven.jahred@gmail.com.

German Romero Castro is a baccalaureate student in electrical engineering at the University of Central Florida. His research interest is in low-voltage analog circuit design. Contact him at german_romerocastro@knights.ucf.edu.

Justin D. Sapp is a baccalaureate student in computer engineering at the University of Central Florida. His research interests are in neuromorphic computing and emerging nonvolatile memory devices. Contact him at jsapp@knights.ucf.edu.

REFERENCES

[1] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 2921–2926, 2017.

[2] A. N. Mikhaylov, O. A. Morozov, P. E. Ovchinnikov, I. N. Antonov, A. I. Belov, D. S. Korolev, M. N. Koryazhkina, A. N. Sharapov, E. G. Gryaznov, O. N. Gorshkov, and V. B. Kazantsev, "Towards hardware implementation of double-layer perceptron based on metal-oxide memristive nanostructures," *CoRR*, vol. abs/1711.01041, 2017. [Online]. Available: http://arxiv.org/abs/1711.01041

[3] S. Raschka, "MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack," *Journal of Open Source Software*, vol. 3, no. 24, p. 638, 2018.