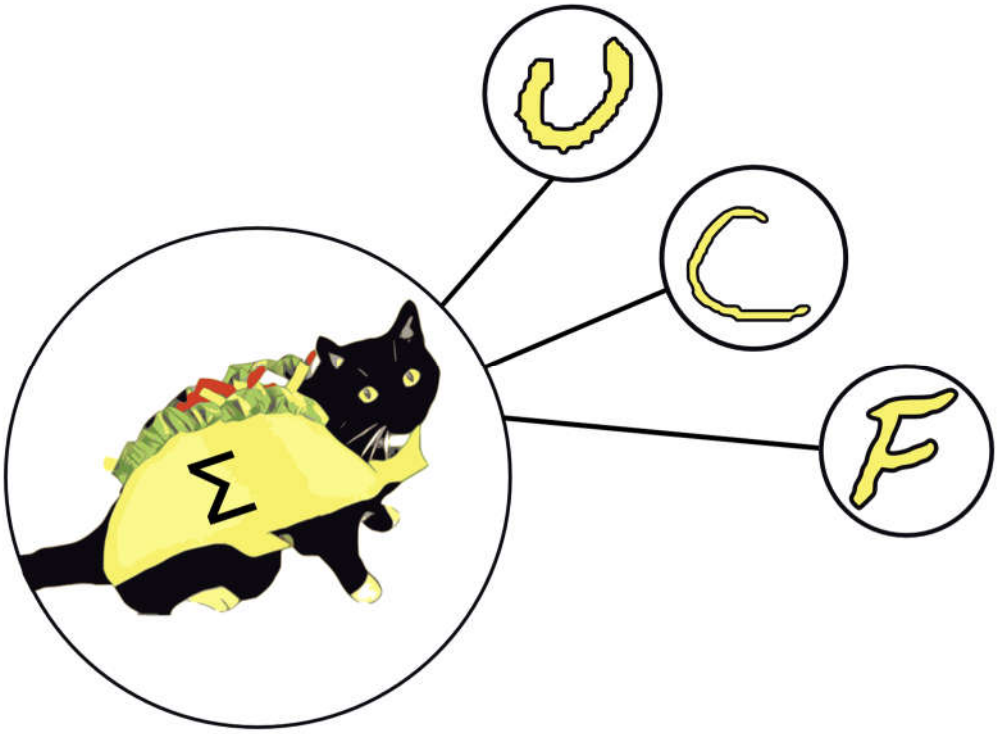
TACOCAT:

Trainable **A**cceleration of **C**lassification **O**perations via **C**ommonly **A**vailable **T**echnology



Group 31 Authors:

German Romero Castro
Electrical Engineering

Luke Minks
Electrical Engineering

Deven Morone
Electrical Engineering

Justin Sapp
Computer Engineering

Mentor:

Dr. Chung Yong Chan

Fall 2019

1 Table of Contents

1 Executive Summary

Over the past decade, major advances have been made in machine learning technology. However, the most common implementations of machine learning algorithms are currently software-based systems that perform a large number of mathematical operations digitally, and due to the computational expense of these operations, tasks such as voice-to-text transcription cannot be performed on inexpensive mobile devices but must instead be sent out to central servers for processing. Hardware neural network implementations, capable of massively parallel analog computations, may soon offer an inexpensive way to accelerate these operations on mobile platforms and other resource-constrained devices.

The main objective of the Trainable Acceleration of Classification Operations via CMOS Analog Technology (TACOCAT) project was to create a hardware-based implementation of a neural network that could be used for machine learning and classification/recognition tasks. While the emerging devices required to build a very large-scale neural network may not be ready for production until several years from now, we believe that we gained valuable experience in neuromorphic hardware design by building a small-scale, yet highly capable, neural network circuit using commercially available parts.

Our discussion of the design process begins with the background and motivations that led our team to begin working on TACOCAT. We then describe the project objectives for TACOCAT and the requirement specifications that it was expected to adhere to.

In the following sections of this document, we provide a review the research that our group conducted prior to beginning design work on TACOCAT. Different technologies that are relevant to the design and fabrication of hardware-based neural networks are investigated, some of which are being applied to actively marketed products, while others are mainly described in academic research literature. We also discuss the design process for a small-scale prototype that was designed and constructed as a proof of concept prior to the creation of the full-scale TACOCAT design. This small-scale prototype was a necessary milestone to reach before we went on to constructing our final network.

Later, we explain the design processes for the full-scale version of the neural network, focusing on device and circuit-level design of the neuromorphic hardware. This includes descriptions of the synapse circuit, the neuron's summing amplifier stage, the non-linear activation-function circuit, activation-function output buffers, and the comparator circuits that are used to display the result of the circuit's classification output. We also describe the design of all of the project's firmware and software components in detail.

Following those descriptions, we discuss the fabrication and testing processes that we employed to build the full-scale TACOCAT prototype. We also describe administrative details of the project, such as budgeting, management, project personnel, and major milestones along the development timeline.

In the final section of the document, we provide a summary of the project and some concluding remarks about the design process for TACOCAT.

2 Project Description

The following sections describe the background and motivation for the TACOCAT project, the project's objectives and requirement specifications, and a high-level overview of the project's design process including a summary of the different project tasks that were assigned to each team member.

2.1 Project Background and Motivation

In the beginning part of the 21st century, the roles that machine learning technology plays in our everyday lives have increased substantially. Machine learning-based systems are used for human speech recognition, handwriting transcription, facial recognition, real-time control of complex electrical and mechanical systems, autonomous vehicle navigation, and many other applications. We interact with some of these technologies directly, such as the facial recognition systems that mobile device owners might use to unlock their phones or tablets. Other machine learning systems, such as those that control fuel injection trim parameters in some automotive engines, go unnoticed by most users.

We have also begun to see that these technologies can be used in ways that might be harmful to society. Whereas facial recognition technology offers convenience to mobile device users, it also could offer a convenient pathway towards intrusive and omnipresent surveillance by authoritarian governments who would wish to constantly track and record the whereabouts of its citizens. Similarly, whereas autonomous vehicles could one day offer increased safety, accessibility, and efficiency in personal transportation applications, this same technology could be used in military applications to create automated attack vehicles. While weaponized autonomous vehicles might be used benevolently in defensive roles, the potential for malicious use is also very real. We are almost certain to face many challenges in the remainder of this century as we decide how to best apply these powerful new technologies.

Currently, most machine learning-related processing tasks have high computational overheads. Especially for optimization and recognition-based tasks, the training of neural networks and processing of data through those same networks requires a very large amount of multiplication and addition operations. Artificial neural networks very often have millions of artificial synaptic connections, and each one of these connections requires a multiplication operation to be performed between the synapse's weight value and its input value in order to carry out calculations using the network. Additionally, at each artificial neuron node (where groups of synapse connections terminate), all of the incoming synaptic input-value/weight products must be summed. For mobile devices and typical consumer-grade PC equipment, these operations can be very time consuming. To address this problem, cloud infrastructure is often employed so that these edge devices can transmit their input data to centralized servers that are able to quickly process the data using high-speed, highly-parallelized vector computations, and the results are then sent back to the edge device.

While this cloud-computing approach is a practical solution, it comes with some drawbacks. One important caveat is that edge-device users must have a high-speed data connection in order to communicate with cloud servers and receive their results with an acceptable amount of latency. Additionally, communication operations require energy and

bandwidth that are often limited in the short-term for mobile users by constraints such as battery capacity and monthly data transmission limits. Finally, the need for central processing to perform recognition tasks, particularly for image and speech data, presents significant concerns regarding privacy. Even if assurances are given that any data processed via cloud services will be confidential, the average user has virtually no way to ensure that these guarantees will be honored. On a larger scale, the fact that users must have access to centralized resources in order to have their data processed can create leverage for corporate or government entities to restrict access to these computing resources, possibly shifting the balance of power away from individuals and towards centralized corporations and state-run organizations.

However, new technologies may soon offer a solution to these problems. While traditional CMOS-based computer architectures require large amount of processing and memory resources to perform machine learning operations in a reasonable amount of time, specialized hardware architectures that incorporate emerging non-volatile memory devices are showing promising results in performing the same calculations using much less energy and much smaller physical footprints while still delivering very low computational latency.

While most publicly-disclosed instances of these hardware-based neural networks are found in academic research works, the fundamental emerging technology that will be required to produce this next generation of neuromorphic circuits, namely very-large-scale integrated non-volatile resistive memory devices, are already being brought to market in niche memory products. In a joint venture, Intel and Micron have commercialized a resistive memory technology known as 3D XPoint. Adesto Technologies is currently marketing a Conductive Bridging RAM (CBRAM) technology that also implements integrated non-volatile variable-resistance devices. Other examples of firms that are actively marketing non-volatile resistive memory products include Fujitsu, Panasonic, and Crossbar Technologies. Although these RAM-oriented products do not offer an interface that would allow the resistive devices within to be accessed in a way that they could be used directly in artificial neural networks, their commercial availability is a positive indicator for the state of practical non-volatile memory device fabrication processes.

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 TACOCAT project was inspired by the idea that a group of undergraduate students in Electrical and Computer Engineering 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 design patterns and also provide valuable experience in an up-and-coming field of research and development.

TACOCAT employs a surprisingly small number of artificial neurons and synaptic connections in order to perform somewhat specialized, yet highly complex recognition

tasks, and we hope that it will contribute in some way towards a technology that could add convenience, privacy, liberty, and efficiency to the everyday lives of people around the world.

2.2 Project Objectives

TACOCAT's neural network is designed as a Multi-Layer Perceptron (MLP), which uses several layers of artificial "neurons" each with multiple "synapse" inputs. A scalar numeric weight value is assigned to each synaptic input, and the neuron's output is determined by multiplying each synapse's input value by its weight value, taking the sum of those products, and applying that sum value to some non-linear activation function. Common activation functions include the logistic sigmoid function, hyperbolic tangent, and rectified linear unit.

A fully-connected network of these neurons and synapses can be "trained" using a set of input data samples that are paired with labels indicating the network's expected output. Inputs are applied to the synapses of the first neuron layer, and outputs are sampled from the neurons in the final layer. We use an algorithm known as gradient descent optimization to implement training by backpropagation, where each layer's outputs, starting with the final layer and moving backward, are compared to the expected output values, and that layer's synaptic weights can then be adjusted based on the activation function's derivative in an attempt to minimize the output error percentage. This process is repeated for the entire network over multiple "training epochs" until some minimum level of error is attained.

In most neuromorphic circuits, synaptic weights are represented by variable resistances. Emerging non-volatile memory technologies such as memristors/resistive RAM, phase change memory, and magnetic tunnel junctions may soon offer nanometer-scale, low-cost devices that can store these synaptic weights, but most current research works on neuromorphic hardware are based on custom-fabricated VLSI devices that are not available to the general public.

Instead of using novel devices, one of our design objectives was to use common digital potentiometers (with onboard memory) to implement non-volatile synaptic weight values in our circuit. While these devices may be too large and expensive to use in a circuit with millions of synapses, they should be useable in a circuit on the scale of several hundred synapses. The neurons' summing and non-linear activation functions are implemented using operational amplifiers.

The main functional objective for the TACOCAT project relates to its ability to recognize input data and identify one of the discrete classes that each data sample belongs to. For a data set of different line patterns, these classes could be vertical, horizontal, or diagonal. They could also be straight and curved. The classes that are used to describe the data are determined prior to training this type of neural network.

While we planned to build a small-scale prototype network that is capable of classifying different line orientations, as mentioned in the example above, the goal for the full-scale TACOCAT network is to recognize a limited set of handwritten characters. The data samples of handwritten character images are available in datasets provided by the U.S. National Institute of Standards and Technology (NIST). TACOCAT will not have a sufficient number of synapses and neurons to accurately recognize a large number of

different character classes, but the project will instead be focused on the objective of recognizing a small subset of character classes with reasonably high accuracy. We plan to use the characters ‘U’, ‘C’, and ‘F’ as the classes for our sample data.

Given a sufficiently large number of data samples, the probability of randomly guessing the classification for any given input sample would be roughly 33.3%. Our goal for TACOCAT is to reach at a level of at least 50.0% for our prediction accuracy, and optimistically, we hope that we can attain levels of accuracy higher than 75.0%.

We also have an educational objective for this project, which is to become familiar with the design process for both neuromorphic hardware and machine-learning software models. We expect that we will achieve the hardware aspects of this goal through the processes of researching, designing, and testing the elemental circuits involved in hardware-based neural networks. For the software aspects of our educational objective, we expect to gain an understanding of the fundamentals of machine-learning models by designing, writing, and testing our own code for the implementation of a multi-layer perceptron. By avoiding the use of existing high-level machine learning frameworks and instead producing all of the core software model code from scratch, we should gain a command of the fundamental aspects of designing and implementing a neural network model in software.

2.3 Requirement Specifications

In Table 2.1 and Table 2.2, sets of absolute maximum and absolute minimum requirement specifications for the TACOCAT project are listed. The absolute maximum specifications serve as hard limits on certain aspects of the final design, including physical and electrical characteristics. Alternatively, the absolute minimum specifications describe the minimum levels of performance and system capacity that will be considered satisfactory for the design.

2.3.1 Absolute Maximum Specifications

Because the physical characteristics are being specified at a very early stage in the design process, we have tried to assume very pessimistic values for the absolute maximum specifications regarding physical size and weight. We arrived at these figures by approximately doubling our expected measurements for these characteristics, while making sure that these cautiously pessimistic estimates would still describe a prototype that would be practical to build and safe to operate.

Weight	5 kg
Footprint area	1 m x 1 m
Supply Voltage	18 Volts rail-to-rail
Power Consumption	10 Watts
External Temperature	50° C

Table 2.1: Absolute maximum physical and electrical characteristics

2.3.2 Absolute Minimum Specifications

For the absolute minimum specifications, we were able to look at the results of software-based simulations to see what characteristics would be required in order to implement some appreciable level of handwritten character recognition capability. Simulations indicated that the neuron counts listed in Table 2.2 should be adequate to allow a small network to distinguish between two different handwritten letters, although it should be mentioned that at this size, while the network will most likely have decent prediction accuracy with easily distinguishable sets of input letters, such as 'X' and 'O', it would most likely struggle to classify letters with similar features, such as 'E' and 'F'. Estimates for throughput and latency are very pessimistic, but even as worst-case estimates, these results would be acceptable for testing and demonstration purposes.

Number of input neurons	25
Number of output neurons	2
Input data resolution	1 bit
Output latency	500 ms
Throughput	1 recognition operation per second
Accuracy	50%

Table 2.2: Absolute minimum operational and performance characteristics

2.3.3 House of Quality

The House of Quality diagram shown in Table 2.3 describes the trade-offs that are involved between different aspects of the project design.

One of the key points illustrated in these diagrams is that a larger number of neurons (and consequentially many more synapses) are often required in order to attain higher levels of prediction accuracy and flexibility/generality. While this principle holds true in software-based neural networks, it is less likely to be a major cost-driver in that domain. On the other hand, because hardware-based neural networks can conduct large numbers of mathematical operations in parallel, increased neuron counts are less likely to have a noticeable effect on output latency times.

2.3.4 Additional Specifications

The following specifications are also included in the TACOCAT design specifications:

- All voltage supply rails should be isolated from the mains supply voltages
- All ground rails should be tied to earth ground
- All printed circuit boards should be securely mounted to a panel or chassis using bolts or screws
- Prediction outputs should be displayed to the user via dedicated visual indicator devices such as LEDs that are activated by the neural network's output neuron voltages, without any additional software processing
- The user interface hardware should be mounted in a manner such that the user will not be exposed to any sharp edges, exposed electrical wiring, or components with high surface temperatures
- The user interface should be accessible for users who are color-blind

		Neuron Count	Size	Cost	Power	Training Time
		+	-	-	-	-
Speed	+	↓	↑			↑↑
Accuracy	+	↑↑	↓↓	↓		↑
Size	-	↓↓		↑↑	↑	
Cost	-	↓↓	↑↑			↓
Flexibility	+	↑				↑↑
Requirement Targets		25+	< 1m x 1m	< $275	< 10W	< 1 hour

Table 2.3: House of Quality

2.4 Design Overview and Assignment of Responsibilities

The neural network block diagram shown in Figure 2.1 represents the main hardware components of the TACOCAT project. The 3 block types that it is comprised of (labeled “A”, “B”, and “C”) represent the 3 functional hardware component groups that are used to construct the TACOCAT neural network. These components must be able to multiply, accumulate, and introduce non-linearity inside the network so that the network weights can be trained properly using machine learning algorithms. The blocks of this network diagram

will be made up of digital and analog circuitry that will execute these functions while being trained to correctly recognize user input in the form of handwritten-character image samples.

The first block, A, is made up of a line driver circuit. This circuit will provide buffered and inverted source voltages to the next block in the sequence. The voltages that this circuit provides will be fed to block B, which is the circuit that will supply the synaptic weights that the input source voltages will be multiplied by. These synaptic weights will be adjusted, at the system level, by an on-board MCU that controls the training process.

Finally, block C of the diagram contains the summation and nonlinear activation-function circuits used in the neural network. The first component of this block must add all of the outputs of the B blocks-from the previous stage of the diagram, thus completing the accumulation portion of the multiply-and-accumulate requirement of the network. This will be achieved using an inverting summing amplifier circuit. Afterwards, these accumulated weighted inputs will be subjected to a non-linear activation function using another analog circuit to implement the activation function requirement for the TACOCAT network. This non-linear activation function will be implemented using another operational amplifier circuit with an external rectifier-based clipping circuit. A brief summary of the blocks and components utilized in the intermediate neural network design can be seen in Table 2.4.

Note that this neural network layout consists of a 4-pixel recognition circuit representing an intermediate network with a 2-by-2-pixel square input image pattern. Our goal for this smaller scale network is to configure it using a standard training algorithm to distinguish classes of input patterns between horizontal, diagonal, or vertical lines in 4-pixel user inputs. Once this network is fully implemented and tested, we will look to expand the size of the network to a 5x5, 25-pixel network that will be able to recognize the difference between handwritten user input characters. This should be achievable, as we expect the only major difference to be a larger number of components and circuit boards necessary to construct the network. It is expected that by increasing the size of the neural network, especially by increasing the size of the network's input layer, that the range and efficacy of the network classification abilities should improve.

The current design of the 4-pixel network's input layer receives all of the input signals of the sensed image as parallel inputs to the line driver circuits of the synapse in the input layer. We would expect this design pattern to become much more difficult to implement as the size of the input layer increases proportionately to the squared width of the input image, so a different method to process these inputs will most likely become necessary once a larger number of sensed pixels are considered. A detailed re-design of the input layer for the final neural network architecture is explored in section 5.6 of this document.

There are a number of other parts that will be required to produce the final demonstration version of this hardware-implemented neural network design. This project seeks to take a digitally processed array of image pixel data as the input to the neural network. After being trained, the demonstration of the efficacy and accuracy of the trained neural network that has been designed, will be a test of character recognition.

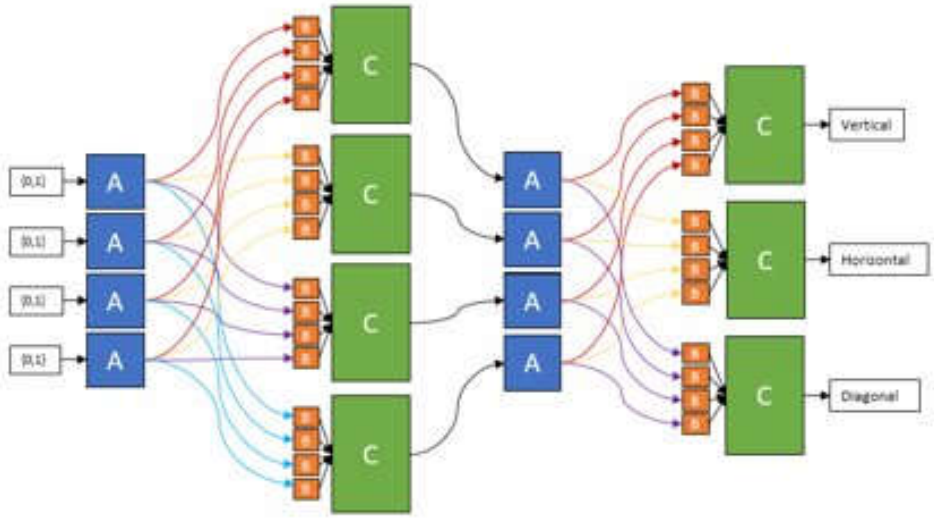


Figure 2.1: Intermediate Neural Network Block Diagram

Block Name	Function	Neural Network Component
A	Line Driver	Synapse
B	Adjustable Weight	Synaptic Weight
C	Multiply and Accumulate	Neuron

Table 2.4: Intermediate Neural Network Block Diagram Summary

The distribution of labor and responsibilities for completing the tasks and components of the final demonstration necessary to achieve the task described above, can be seen in Table 2.6, which also coincides with the block diagram of TACOCAT's system.

Our finalized project will be comprised of six main components as illustrated by the block diagram in Figure 2.2. A commercial power supply will be utilized to provide power to all electrical components. This should be appropriate since we will meet the project's hardware expectations with what has been explained in the paragraph above.

The MCU we choose will be responsible for the low-level aspects of the training and adjustment of our network's circuit. Data sets will be provided to the MCU such that the network can be trained to recognize the user input that the network receives. User input will be provided by a touchscreen interface with which the user can write an alphabetical character, draw a vertical, horizontal, or diagonal line, or create whatever other input images might be relevant to a particular network configuration. This input will be processed and transmitted to our neural network as digital pixel data through the MCU.

Work Distribution	
Deven	
German	
Justin	
Luke	

Table 2.5: Responsibilities of Team Members

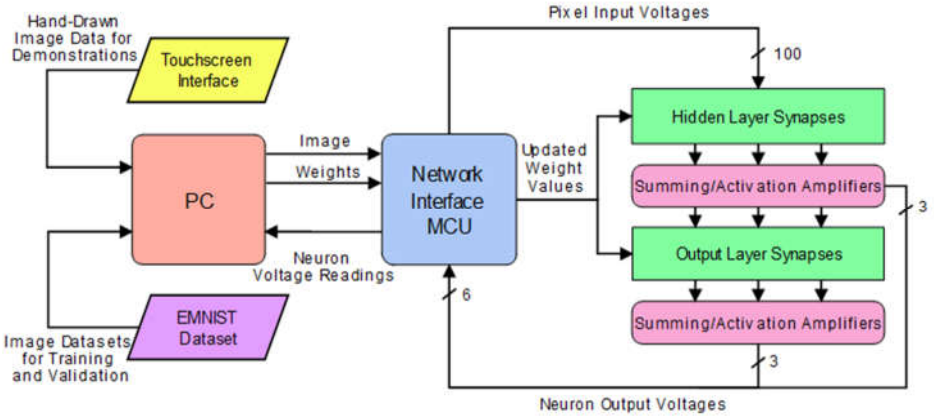


Figure 2.2: System Block Diagram

Once this pattern recognition or handwritten character recognition (depending on the scale of the network) is completed within the neural network hardware, the result will be displayed in an LED status array, where the user will be able to see the network's prediction for either the orientation of the line they have drawn or the handwritten character that they have entered.

Because a significant share of the hardware components of the network are heavily entangled with numerous other portions of the hardware, it is difficult to modularize the project. Since changing a single component can, in many cases, result in changes to almost every other module given above, the role assignments are predominantly administrative, not technical, since adjustments to a broad spectrum of features must often be made by a single person.

3 Research

Research is a crucial part in the development of any high level project. A logical starting point for TACOCAT project research is to survey existing software and hardware implementations of neural networks. Innovation sets new designs apart from existing works, but research is necessary to understand how the network will be implemented. It is important that we understand the software and hardware that is behind building an analog neural network.

3.1 Relevant Technologies

Each neuron in the network sums its inputs and produces the output dictated by its activation function. However, as this is not a binary network, meaningful computation cannot be achieved with unweighted inputs, as the only possible input and output values would be -1, 0, and 1, assuming ideal circuitry. This can be remedied by weighting the input of each neuron, which allows the input to be multiplied by an arbitrary value within a certain range. By weighting the network's inputs, each signal can be amplified, reduced, inverted, or nullified as desired. While a software-based neural network brute forces the signal weighting on each synapse, a simple algorithm can be used to read the voltages at each node of a physical network and adjust the weighting accordingly. As this network's signaling is based on voltage level, this weighting can be accomplished via voltage division. While voltage division is a straightforward enough concept, there are numerous ways to accomplish this goal which must be considered. The most obvious candidates are memristors, analog potentiometers, digital potentiometers, digital rheostats, and digital-to-analog (DAC) arrays.

3.1.1 Memristors

Memristors are solid-state variable resistance devices which do not have any digital component or moving parts. The most common way of implementing memristor technology is through use of titanium dioxide films with different levels of oxygen depletion; the oxygen vacancies can be shifted around by electric current, adjusting the resistance of the device. This can be achieved by "programming" the device with a higher voltage the device would otherwise operate at, as the lower operating voltages will not push the device out of its hysteresis and disturb the resistance setpoint. A diagram of the typical construction of a titanium dioxide memristor is shown below in Figure 3.1. Consequently, the memristors are entirely analog and do not require any direct digital control to program. Additionally, due to their potentially small size – implementable at a nanometer level – it is possible to achieve extremely high junction densities for use in integrated circuits. However, due to the infancy of the technology and inherent variability in manufacturing, memristors currently exhibit very high failure rates. While reliability is fairly good for functioning junctions, many remain stuck "open" or "closed" upon fabrication and cannot be adjusted, while many more are only partially usable and cannot operate across the expected range. Additionally, it is a fairly complicated endeavor to implement hardware to allow each individual memristors to be repeatedly and precisely reprogrammed, as the high voltage must be supplied without damaging hardware which is otherwise designed to

operate at lower voltages. Consequently, memristors are not suitable for this design, though they would be the only option if an integrated circuit implementation was desired.

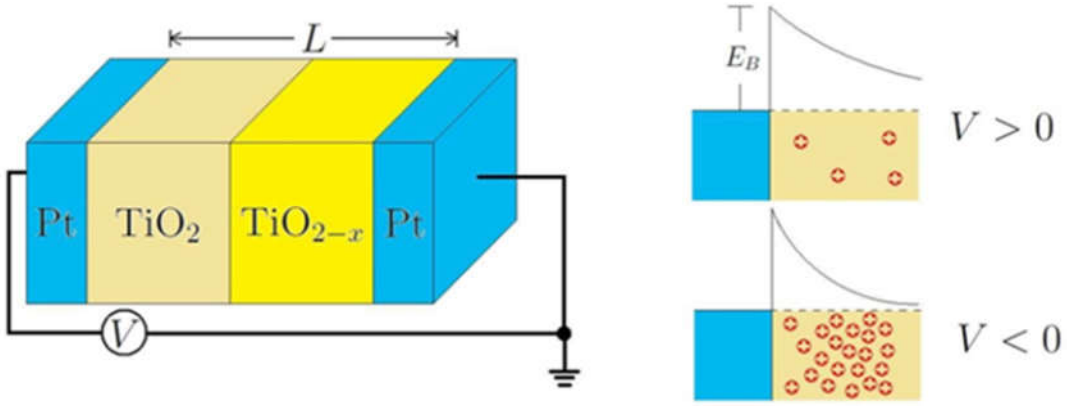


Figure 3.1: Titanium-dioxide Memristor Diagram

Titanium dioxide memristors may be common, but there are several types of memristors to be considered. There are two main different types of memristors, molecular and ionic thin film memristors, and spin and magnetic resistors. The molecular and ionic thin film memristors rely on unique properties of each type of material. The four types of memristors of the molecular and ionic thin film family are titanium dioxide memristors, polymeric/ionic memristors, resonant tunneling diode memristors, and manganite memristors. The titanium dioxide memristor shown in Figure 3.1 is usually used for modeling. The image used in figure 3.1 has reproduction permission requested in the image reproduction section of this document. Polymeric/ionic memristors use active doping of di-electric materials to create solid-state ionic charge carriers. Resonant tunneling diode memristors use specifically doped diode junctions as the breakdown layers between drain and source of CMOS components. While a manganite memristor consists of a bi-layer oxide film substrate that is dependent on manganite contrary to using titanium dioxide.

Magnetic Tunnel Junctions (MTJs) use the interaction between magnetic fields and electrons with different spin states to create junctions with high and low states of resistance. Because resistance in an MTJ depends on the polar orientation of a variable magnet, these resistance states are persistent even when the device is powered down. This can provide significant energy savings in a system that requires non-volatile storage. There are a number of different styles in which MTJs can be fabricated, including Spin Orbital Torque MTJs, Spin Hall Effect MTJs, and even application-specific designs such as the stochastically-activated P-Bit MTJ. The main drawback for MTJ use in synaptic weighting systems is that the ratio of an MTJ's highest resistance value to its lowest resistance value is much lower than other resistive memory technologies. Current MTJ technology is limited to resistance ratios in the 0-10 range of magnitudes. This order of magnitude allows for reliable storage of binary data values using typical sensing circuits to read the stored values, but it does not provide enough margin to store a wide range of analog values.

3.1.2 Analog Potentiometers

Potentiometers are, at present, the most obvious and widespread way to implement variable resistance for most applications. Analog potentiometers are one of the oldest types of

variable resistors and can be found in a wide range of contexts. Generally speaking, analog potentiometers have three terminals: the two ends of the potentiometer and the wiper. The wiper can be moved along the resistor between the two ends of the potentiometer, allowing an effective resistance anywhere between the relatively minimal resistance of the wiper and the full resistance of the potentiometer to be chosen. While the details of the implementation may vary, such as the precision with which the device can be adjusted and the mechanism of the wiper, the behavior is the same. In the case of this project, analog potentiometers are useful for basic prototyping and experimentation. However, they are not useful for actual network implementation due to the lack of usability in training. Because training often takes hundreds of epochs and requires precise adjustment of the weighting of each synapse, it could take days or weeks of manual adjustment of each potentiometer to arrive at a properly calibrated device, even for a very small test network. Additionally, because it is difficult to precisely adjust the potentiometers by hand, it is difficult to ever arrive at an accurately calibrated value, as even a few poorly calibrated synapses can result in both erroneous adjustment data from the training algorithm and incorrect network outputs. Consequently, analog potentiometers are unsuitable for this project beyond early prototyping and experimentation.

3.1.3 Digital Potentiometers

Digital potentiometers are, perhaps obviously, digital implementations of analog potentiometers. Unlike analog potentiometers, which can ideally be adjusted to infinitely precise values, digital potentiometers are implemented with discrete resistance ladders. Due to the use of discrete resistance components instead of a true analog potentiometer mechanism, there are a finite number of possible configurations – corresponding to the device's number of potentiometer taps – that the device can produce. This value is typically represented in binary using 7 or 8 bits. Given a value, the device automatically adjusts the switches on the resistor network to produce the appropriate total resistance. While some precision is lost, especially when using devices with lower tap counts, the overall accuracy of each device is far higher in a network application, as the resistance at each tap in each device does not change appreciably between adjustments. Additionally, the relatively low cost and small footprint of digital potentiometer chips, along with the widespread availability of devices with EEPROM and serial communication protocol functionality makes them an excellent choice for smaller-scale neural networks.

3.1.4 Digital Rheostats

Digital rheostats are very similar to digital potentiometers and are often sold in the same packaging and series of chips as regular digital potentiometers. The key difference between rheostats and potentiometers is that while potentiometers use three terminals, the rheostat only has two terminals, corresponding to a simple variable resistance instead of a potentiometer. While rheostats can be useful for high-power applications and for simple resistance trim adjustments, these advantages are not particularly applicable to this project. However, the key disadvantage of rheostats in this context is the devices' lack of a third terminal. Because the synapse weights of this network functions by using the wiper to choose between a positive reference voltage and a negative reference voltage according to the principle of resistive voltage division, two rheostats would be needed for each synaptic

weight. Rheostat devices also might not offer the same accuracy as true potentiometers due to the lack of the inherent proportionality exhibited by the wiper mechanism in potentiometer devices. Thus, rheostats are less suitable for this project in comparison to three-terminal potentiometers.

3.1.5 Digital-to-Analog Converters

Digital-to-analog (DAC) converters operate very similarly to digital potentiometers, using a discrete resistor array of one form or another to allow the desired resistance to be selected. The primary difference between digital potentiometers and DAC arrays is the presence of an op amp on the output of the DAC array. This output op-amp buffers the output voltage of the device, providing extremely precise output voltages and preventing the output voltage from acting as a function of the output current, as may occur with digital potentiometers. However, DACs suffer from a major handicap related to their basic function. DACs divide their provided reference value into a predetermined number of steps (256, 1024, 4096, etc.), but do so relative to the voltage connected to the ground of the device. Because the voltage output of the previous neuron can potentially be negative, the DAC may either be exposed to a reference voltage lower than its ground voltage or incorrectly divide a positive voltage relative to the wrong ground level. While DACs would be useful in this case for adjusting the initial inputs to the network, the only possible way to implement DACs in an entire network would be to employ ADC conversion at each neuron's output and then use the DAC to select an output voltage on each synapse, defeating the purpose of an analog network. Thus, while DACs are extremely useful in certain applications, their utility is limited in this project.

3.1.6 Hardware vs. Software Neural Network

Artificial neural networks can be implemented using a software approach or a hardware approach. Using software to implement the network is generally a simpler approach, while hardware design tends to be a longer, more difficult process. Software neural networks can be modeled in many different programming languages, including Python, C, C++, and Java as common examples. Software packages for constructing neural networks and running simulations are also widely available. Examples of these programs include Neural Designer, GMDH Shell, Neuroph, Darknet, DeepLearningKit, and many more. It is much simpler to design and implement a neural network using software due to the ease of code revision in comparison to physical circuit modification, but creating a neural network implemented with hardware has key benefits.

Artificial neural networks implemented by hardware on a large scale may soon be faster and more conveniently manufactured than software-based networks. Typical hardware-based approaches to implementing a neural network consist of digital or analog circuitry that realizes the main functions of a neuron and synapse. The synapse needs to sense the inputs to the neuron by using some sort of input-sensing interface. Next, this input data needs a way to be weighted. Some examples of weight-applying hardware include memristor devices that change their resistance states based on the voltage applied to them, digital potentiometers that change their wiper position based on digital signals, and phase-

change memory devices that change resistance based on the physical state of a resistive material.

The output of the input-weighting-device is fed into a hardware artificial neuron. This hardware neuron must be capable of accumulating all weighted synaptic inputs by some sort of summing amplifier, digital adder, or similar accumulation device. Additionally, in order to solve a wide range of problems using typical neural-network architectures, the hardware neuron must be able to apply non-linearity, which is available through a wide range of analog non-linear devices.

Smaller arbitrary hardware-based implementations already consume much less power and develop results quicker while maintaining higher accuracy. Impressive results have been achieved with existing hardware neural networks, but even better results are expected in the future, particularly as ongoing development non-volatile memory devices allows more efficient implementations of synaptic weighting and input-summing processes. In this regard, the design of new hardware-based artificial neural networks is an exciting frontier in the field of machine-learning research and development.

3.2 Neural Network Architecture

Neural networks are one of the primary tools that are used to achieve machine learning and deep learning in all sorts of engineering applications. As their name would suggest, “neural” networks seek to replicate the way that human brains learn and recognize patterns through interpreting and adapting to sensory data. Artificial Neural Networks (ANNs), however, are currently being designed and implemented with a specific application in mind, such as object or pattern recognition and data classification.

An ANN consists of multiple artificial neurons, which are configured to be grouped into “layers”. These layers have interconnection between nodes in other layers which include synaptic weights that can be adjusted for the network’s adaptation to a specific application through the process of backpropagation. At each artificial-neuron node in a given layer, the combination of input from the data with a set of synaptic-weights coefficients, that either attenuate or amplify that input, serve to assign a significance with respect to the machine learning algorithm used to program it. These weighted inputs are then accumulated by summing them and passed through the node’s assigned “activation function”, which is a pre-defined relation between neuron inputs and neuron output that is used to determine if the input signal should be represented in the neuron’s output in order for it to be processed further down the network and affect the network’s final-layer output. This relationship between inputs and outputs in an artificial neuron can be seen in Figure 3.2.

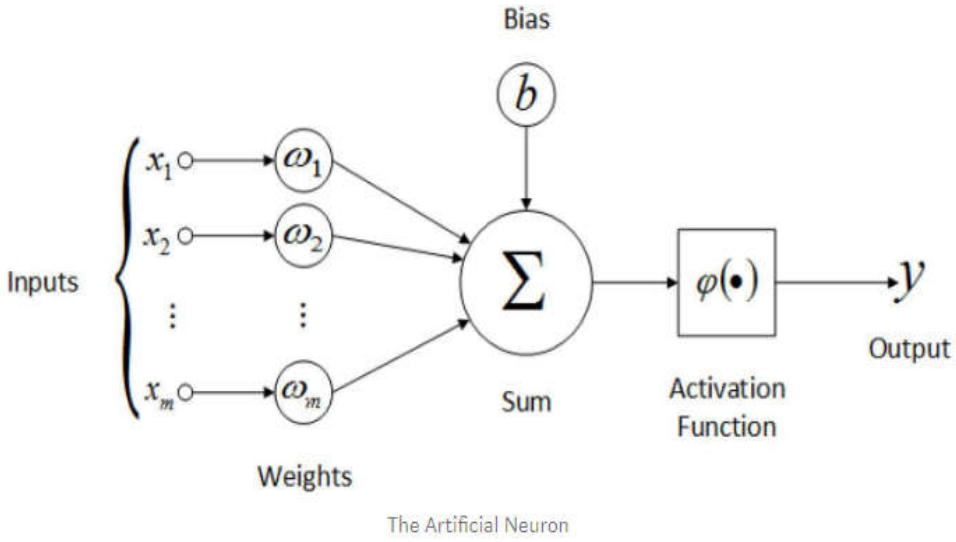


Figure 3.2 Artificial Neuron Layout. Permission requested from https://towardsdatascience.com

Using this basic artificial neuron structure, several neural network architectures can be constructed to achieve the applications desired. The suitability of a network for performing particular tasks depends heavily on the activation functions used in the neurons, the interconnections between neurons and layers, and the machine-learning algorithms used to train the network.

Interconnections in a neural network are the pathways that allow a network's processing elements, artificial neurons, to connect to one another. In their simplest form, these interconnections form at least two of the layers previously mentioned, one being the input layer and one being the output layer. A hidden layer is a third kind of layer, which may not be included in all types of neural networks, that act as a "black box", inaccessible by the users interfacing with the overall system.

3.2.1 Multi-layer Perceptron

The Multi-Layer Perceptron (MLP) is a class of artificial neural network with several unique characteristics and requirements. First, the network consists of at least 3 layers of neurons. These neurons are made up of the inputs and weights, as well as a non-linear activation function. A sigmoid function, such as the hyperbolic tangent or the logistic function (shown in Figure 3.3), has traditionally been used for neuron activation in MLPs. Both typical hyperbolic tangent functions are easily differentiable, which eases the process of backpropagation with gradient descent when training the network. The activation function and its derivative used in the training algorithms are included in (3.1).

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.1}$$

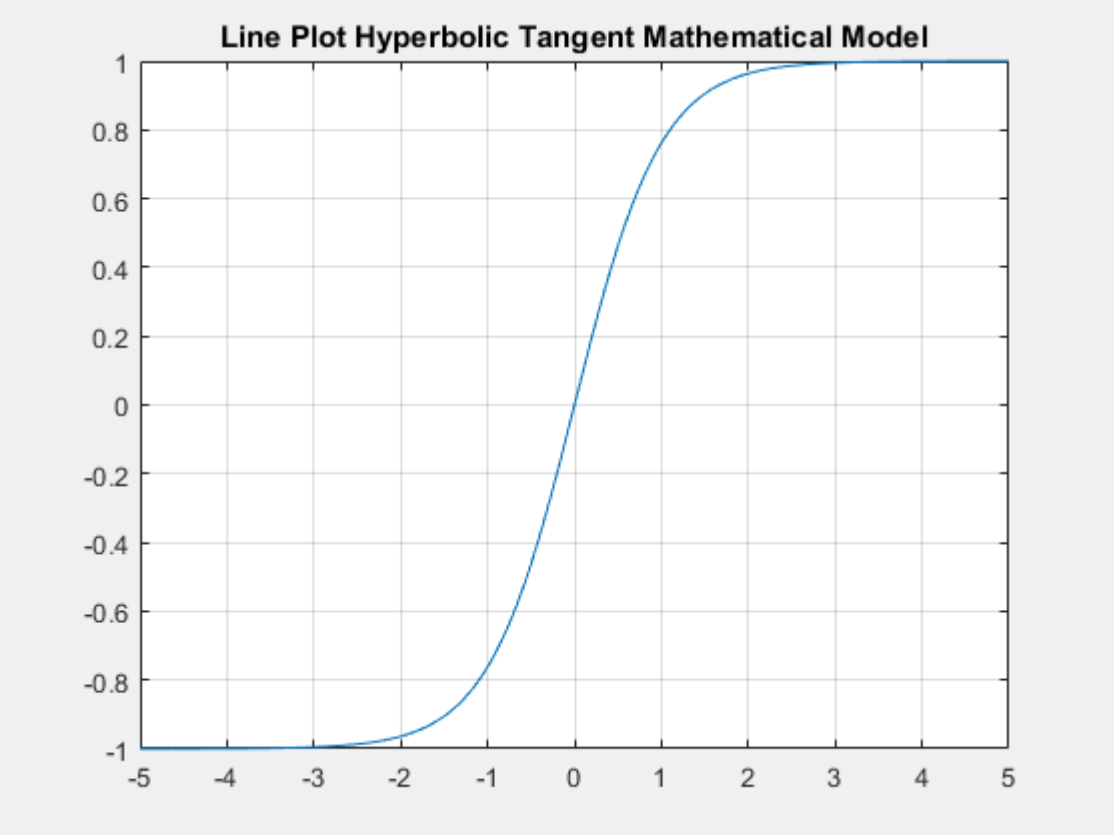


Figure 3.3 Plot of Logistic Hyperbolic Tangent Transfer Function

The MLP must be made up of a minumum of 3 layers of neurons, comprising one input layer, one output layer, and at least one hidden layer in order to solve non-linear, complex problems (Figure 3.4).

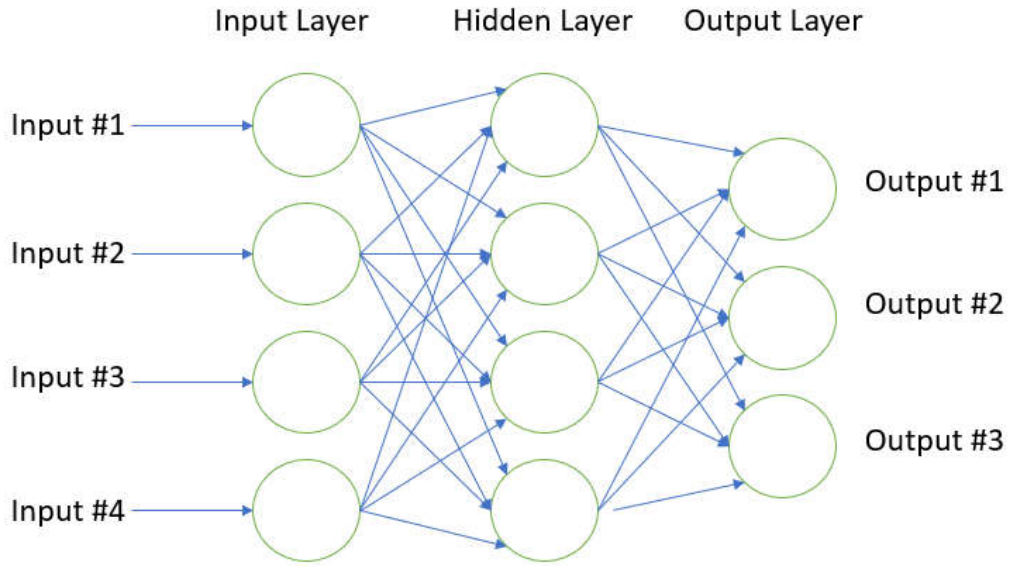


Figure 3.4 Proposed MLP 4-Pixel Input Network Architecture

3.2.2 Convolutional Neural Network

Convolutional neural networks (CNNs) are often utilized for processing input data sets consisting of photographic images. Typically, the CNN consists of the traditional input and output layers found in other neural network architectures, however, since its primary goal is to assign significance to various aspects and objects of an input image and differentiate one of these from another, it must consist of more than the usual number of hidden layers when compared to a regular MLP.

The hidden layers of a CNN primarily consist of their namesake “convolutional” layers. These layers apply filters (also called kernels) that convolve the width, height, and input volume of the previous layer’s output and compute the dot product between the characteristic weights of the layer and the input to produce a 2D map of that filter. These produce a feedforward process responsible for subsampling the input image to facilitate the process of key feature mapping and object recognition of the CNN. Input images to the CNN are in the format of length, width, and channel dimensions before feeding forward into pooling and kernel layers of the fully connected network (Figure 3.5).

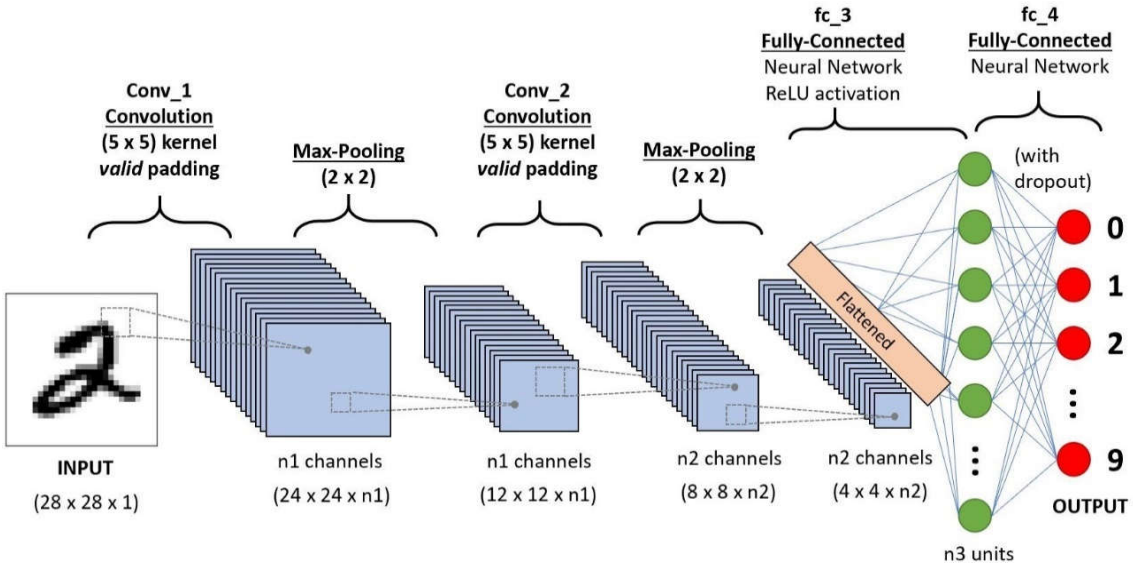


Figure 3.5 Example of CNN Sequence for Recognizing Handwritten Digits. Reproduction permission requested from https://towardsdatascience.com

Combining design principles from both the CNN and MLP network architectures could help realize the final form of the neural network architecture in this project. For the objective of producing a network with 25 or more inputs that is capable of classifying handwritten character image inputs, the image-processing abilities of convolutional layers is certainly desirable, but the complexity of implementing convolutional architectures in hardware may be too great, given the real-world constraints on the amount of time allowed for project development.

3.2.3 Spiking Neural Network

An emerging trend in artificial neural network architecture is the development of Spiking Neural Networks, which are composed of artificial neurons that are designed to emulate biological neurons by performing computations based on incoming binary spike information. These neural networks differ most significantly from other types of artificial neural networks used for machine learning in the fact that they depend on what's known as "spikes", or discrete triggering events, to spike and reset the potential of a single neuron in the network. The activation method for neuron in these networks often follows a leaky integrate-and-fire model, which dictates the creation of an output spike upon the accumulation of a certain number of input spikes within a time period. Spiking neural network architectures are naturally well-suited to implementations that use purely analog circuitry. Although these systems can be based on analog hardware, they are expected to be able to encode digital communications and not lose signal fidelity, with the networks functionality being dependent on receiving the "spiking" input that surpasses a certain threshold and fires the consequent neuron layers. An example 3-layer input-spike to hidden layer to output-spike SNN architecture can be seen in Figure 3.6.

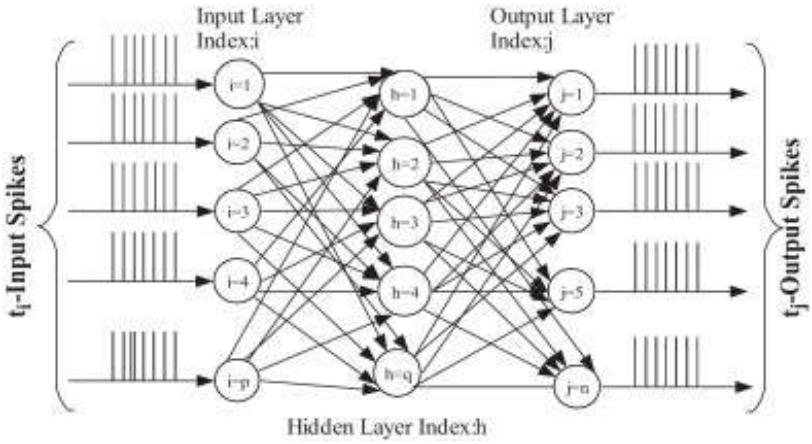


Figure 3.6 Diagram of Multilayer SNN. Reproduction permission requested from https://towardsdatascience.com

Some advantages that spiking network offer compared to other neural network architectures include lower energy usage and increased parallelizability due to neuron spike interactions being localized to the system that they are integrated within.

Spiking Neural Networks (SNNs) are best suited for time-space dependent and event-based information obtained from neuromorphic sensors, thus, a SNN-based design may not be the best suited architecture for the goals we wish to achieve with this project. Machine learning programming and training protocols for our implemented neural network will need to be manually programmed before the network can perform freely. Its computations and overall performance will mostly depend on how well the training algorithm adjusted the weights according to the task required. Given the project's goal of creating a network that can differentiate between hand-written characters, the input data set is composed of bitmapped images. Without potentially large amounts of additional research and development effort, it could be very difficult to translate image-based input data into the spike timing-based input data that a SNN would require, which limits the suitability of an SNN architecture for this project.

3.3 Existing Hardware-based MLP Designs

Several other Hardware-based Multi-Layer Perceptron designs exist in academia research. Some of these designs offer an alternative to fully software simulated implementations of artificial neural networks, which have the possible downside of not providing a real-time response and learning for neural networks made up of many neurons and synapses. This means that a hardware implemented neural network with parallelizable processing and close-to-real-time response capabilities offer a cheaper and faster alternative for more commercial devices' applications, i.e. facial recognition, speech recognition, etc.

Our project aims to prove this concept by implementing the core components of an MLP Neural Network using analog circuitry. First, the synapses are implemented using operational amplifier circuits to buffer and invert the inputs to the neuron. Since weights to the input to the neuron need to be adjusted by the training algorithm, digital potentiometers are used for this purpose, by having both the inverted and buffered input to each end of the potentiometer. The position of the wiper along the entire "resistor"

determines the weighted input into the next summing amplifier, which will accumulate all the weighted inputs to the neuron. Once this is done, the last operational amplifier circuit of the neuron will be used to implement the activation function of the neural network (Hyperbolic Tangent Function), by rectifying the summing amplifier's output using a diode rectifier bridge configuration on the feedback of the operational amplifier.

Key differences between our design and others found in academia research can be found by looking at the analog circuitry used to implement each of the significant parts of the network (Neuron, Synapse, and Activation Function).

3.3.1 Memristor MLP Based Designs

Memristor-based designs have been used in many research experiments when having their analog voltage-dependent-resistance be the weight that will be applied to the inputs of each neuron (Mikhailov et. al.) was desired.

Many benefits of using Memristor-based designs exist when implementing them as the adjustable weights applied to the inputs in the synapse of a neuron inside our neural network architecture. The physical characteristics of Memristors allow for the resistance to be adjusted depending on the "Set" and "Reset" voltages that cause the memristor to change states and alter the resistance perceived by the current passing through the device. The range of pulse voltages and state-change Voltage-Current plot for memristor devices similar to those used by Mikhailov et. al. can be seen in Figure 3.7.

Artificial neurons in this paper's designed experiment utilized complementary memristors paired with a multiplexing chip ADF436 to provide analog multiplexed inverted and buffered input voltages to the neurons, provided from the Atmel AVR-Microcontroller that was used to program and train the network. This weighted input from the memristor devices are fed into a 4-stage operational amplifier circuit. The first 2 operational amplifier circuits are current amplifiers that provide constant rail voltage to the cascaded layers of the network, as well as summing the input layer inputs. The next two operational amplifier circuit topologies serve as the non-linear activation function of the network by taking the accumulated inputs to the neuron and applying a rectifying diode bridge to clip the maximum and minimum peaks. The very last operation amplifier stage is a simple amplified version of the hyperbolic tangent activation stage operational amplifier. A block diagram of the circuit schematic used in this experiment is provided in Figure 3.8.

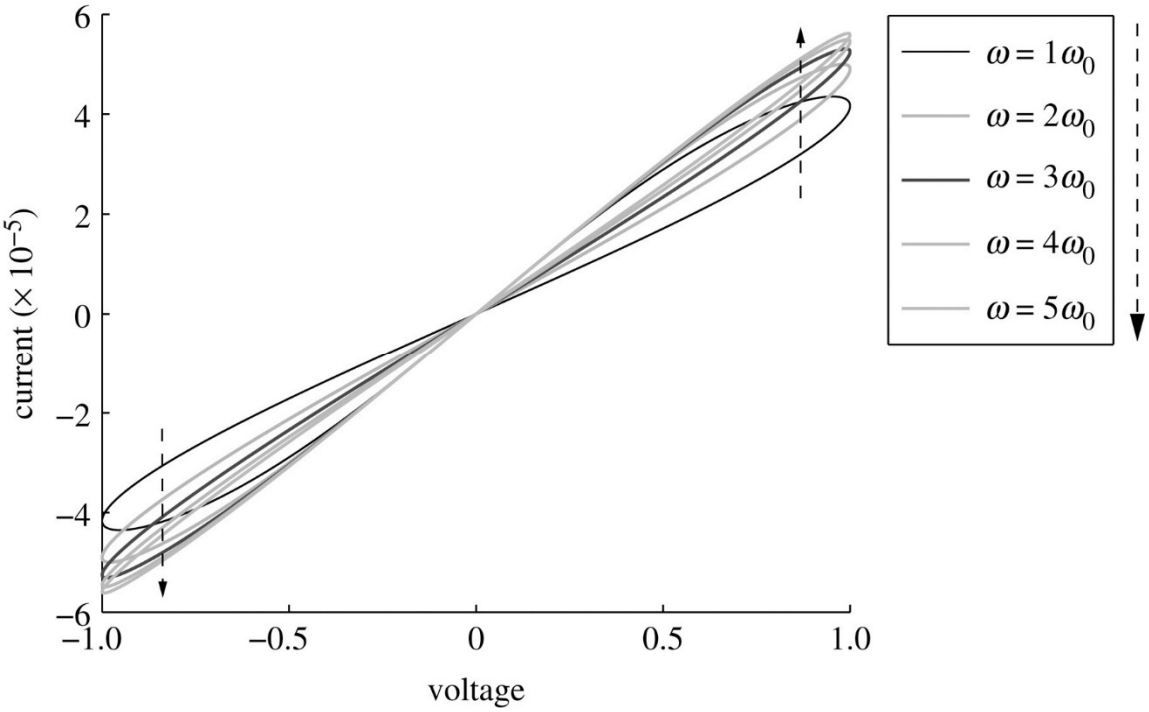


Figure 3.7: Memristor Characteristics Based on Voltage and Current Pulses at several frequencies. Reproduction permission requested from https://royalsocietypublishing.org/

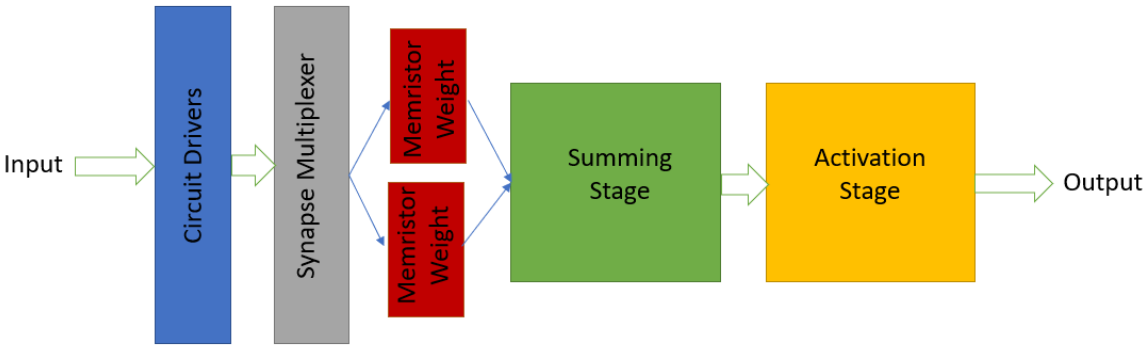


Figure 3.8: Block Diagram of Circuit Schematic used for Memristor based MLP Circuit Design

From the prospective neural network design described in Section 3.4, it can be seen that this circuit design shares many common characteristics of their respective circuit topologies. Our network's synapse shares the use of inverted and buffered inputs to the first operational amplifier stage of our neuron; however, we are replacing the memristor devices with digital potentiometers to adjust the weights of the inputs.

Similar neuron designs are also shared. Although, we don't have an addition current amplifier operational amplifier circuit to provide the voltage line as seen in the circuit diagram of Fig 3.7. In comparison, the rail voltages of the prototype 4:4:4 network are provided from the LM317 and LM337 linear voltage regulator pairs and supplied to the

V_{SS} and V_{DD} rails of the potentiometers and the Teensy 3.5. Finally, both neurons share the same rectified feedback operational amplifier to apply the non-linear activation function of the Hyperbolic Tangent.

3.3.2 FPGA Hardware-Based Designs

Another common hardware implementation of MLP include Field Programmable Gate Array (FPGA) designs of the artificial neural network. One of the implementations of these designs is a digital architecture for the realization of multiple layer feedforward networks using digital logic blocks, in software like VHDL and Xilinx, to achieve this as a configurable system implementable for specific functions (Tisan et al. 2006).

Fully customizable control logic blocks are configured and designed to control the neurons of each layer of the network. Control blocks are distinguished from one another as the design of each layer calls for an adjusted logic block flow. The input layer logic block composition will differ from the logic blocks in the hidden layers and differ from those in the output layer of the network. This digital hardware implementation of a MLP has the ability of learning-on-chip as all the hardware necessary to adjust weights for training the network and reconfiguration of interconnections between neurons in previous or subsequent layers are found directly on the FPGA. It has very high reconfigurability and has the benefit of being able to operate under real-time constraints, making it possible to implement spiking neural network MLP architectures

The entire digital architecture of this type of implementation is split up into certain groups of digital logic blocks that are essential to realizing this design on an FPGA. These groups include the following:

- Control logic block
- Processing block
- Error check block
- Calculus block of hidden layer weights
- Calculus block of output layer weights

The first type of block, the control logic block, is designed to control the neurons of the neural network, giving signals to multiply and accumulate the weight adjusted inputs to the network, as well as finally telling the memory blocks to compute the total accumulated sum of the neuron's inputs. The control logic block also serves to control memory and act as a data buffer for sensory input data to the neuron.

Next, the processing block is designed as the main component of the digital hardware neural network design. It is created to incorporate the artificial neuron functions of summing inputs and applying non-linearity, as well as housing the learning algorithm on-chip.

Neurons are modeled using a block made up of logic blocks that provide memory, one for data sampling, one that multiplies and accumulates, and finally a unit that applies the activation function. All these individual units achieve the main neuron designed compared to this project's design that purely consists of analog circuitry with additional digital circuitry to implement these functions.

Lastly, the calculus blocks for both the hidden layers and the calculus layers use digital logic blocks that calculate the weights applied to the inputs of the neurons in these layers. Their parameters to calculate the weights are modifiable through programming the FPGA when training the network or programming the desired training algorithm for the neural network.

All these separate digital circuit blocks that are used to realize this FPGA artificial neural network implementation with one hidden layer as seen in Figure 3.9.

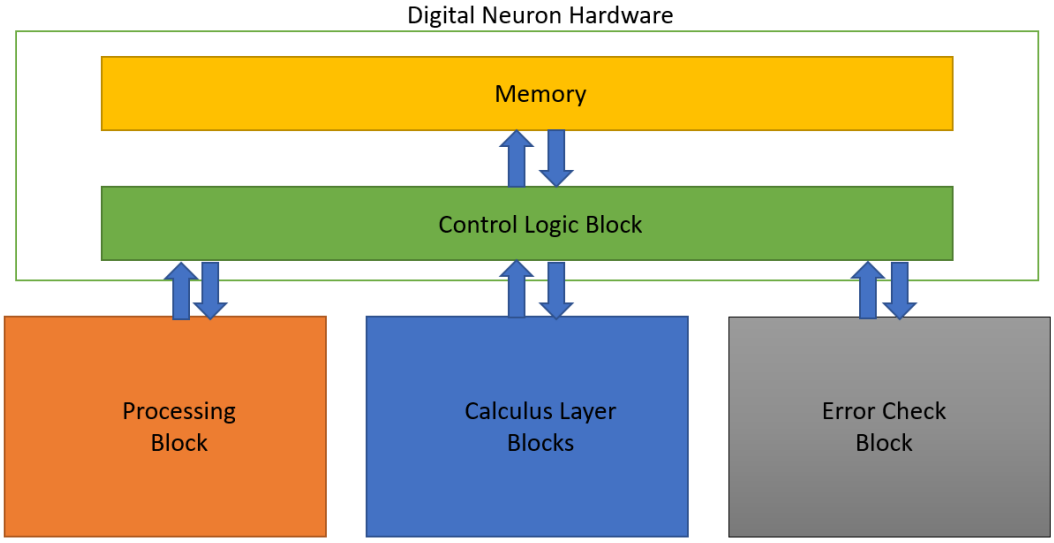


Figure 3.9: FPGA Digital Circuit Blocks utilized for single layer MLP Neural Network Architecture.

There are considerable benefits to implementing these neural networks in Field Programmable Gate Arrays. Their highly parallel architecture, customizability, and very high-power efficiency and low consumption are desirable traits. However, due to the time constraints and high difficulty in programming the entire network on FPGA, the final design architecture will be an analog circuitry MLP neural network architecture.

3.4 Component Selection

Needless to say, component selection is an important aspect of any project. It can be the difference between success and failure. After designing a schematic to perform a certain way in testing software, the actual physical hardware needs to be chosen based on the electrical tendencies of the circuit. Each component must be carefully and intentionally chosen to avoid failure of the neuron circuits. This is one of the most important parts of the entire creative, and design process.

The next step in the design process is to choose the physical components for each stage of the circuit mentioned in the previous section. The TL084 operational amplifier was temporarily chosen to carry out all of the amplification stages, as they are easy to obtain with low cost. They have generally low noise output, and a high slew rate while working

well with low currents. These operational amplifiers are used in each amplification stage of the entire system. They are seen in the buffer stage, the inverting amplifier stage, the hyperbolic tangent response amplifier stage and neuron output stage. The next components that need to be chosen are the digital potentiometers. The digital potentiometers that are currently being used are the MCP42010-E/P integrated circuits. These digital potentiometers have 256 taps which provide a digital wiper value range from 0 to 255 as they are 8-bit devices that can sweep up to 10kΩ of resistance. These chips have several different resistance values offered, including 50kΩ and100kΩ. We temporarily decided on the 10kΩ digital potentiometers because it suits the power consumption of the network. The pin out for the digital potentiometers we chose can be found in the appendix in Figure X.X. These MCP42010-E/P chips have two separate wiper outputs letting us utilize two potentiometers per chip. The digital potentiometer chips can use different types of communication as well, as SPI communication will be implemented. Communication between the Teensy 3.6 Development Board and the MCP42010-E/P digital potentiometer chips is what will control the wiper positions of the digital potentiometers as they will be adjusting the weight of each individual synapse input.

The last component decided on are which diodes to go with for the output clipping of the hyperbolic tangent amplifier stage. The 1n4001 diodes are being used in prototyping because they provide a desired turn on voltage for the required output clipping of the activation stage. The rest of the components throughout this circuit are all just typical resistors and jumper wires that many distributors provide. There was no need to go with any bulky power rated resistors as we are working with low enough current (50uA to 250mA) throughout the entire circuit. In the subsections below, there lies a further analysis of all components considered which shows some insight to how decisions were made when finalizing component selection. The subsections below will give an in depth analysis on how our decisions on component selection were made.

3.4.1 Operational Amplifier Considerations

Amplifier selection is one of the most crucial processes for this design, as everything from input buffering to activation functions is handled by the network's amplifiers. Because the amplifiers in this project are used in a number of ways, the selected amplifiers must be able to function adequately in all of these roles with minimal supporting hardware or concessions. The operational amplifiers chosen to construct the network must meet certain parameters that will be discussed below.

One of the most obvious considerations is the chosen amplifier's output range. The signals being handled in this project run between -1.65 V and 1.65 V, so it is a requirement that the chosen amplifier be able to output across a range of at least 3.3 V. While most amplifiers' power supplies run at a minimum of 3.3 V, a significant portion of them are not rail-to-rail, and many more perform poorly at the upper and lower ends of their operating range, exhibiting behavior such as "latching" their outputs to the corresponding voltage rail, effectively freezing the output in place even if the input's magnitude drops. There are two methods of addressing this problem: find a suitable rail-to-rail amplifier which does not exhibit undesirable behavior when outputting at or near the rail voltages, or use an amplifier with a much larger range than the necessary 3.3 V.

Each method has its benefits and drawbacks. The benefits of a 3.3 V rail-to-rail amplifier are dramatic simplification of the voltage rails for the network, as the digital hardware in the network runs at ±1.65 V, similar to the network signals. However, it can be a fairly expensive endeavor to fully employ such amplifiers, as such precision and functionality is not without matching cost. Given a reasonably priced and suitably functional candidate, however, it is well worth the cost and effort to eliminate excess voltage levels from the network. However, using an amplifier at higher voltage levels, such as Texas Instruments' TL084, is a valid approach; the TL084 is very cheap, at around $.15 per amplifier on the quad-amplifier chips, and is able to function in all of the applications required for this project, such as input buffering and rapid settling due to its high slew rate. As implied, though, the TL084 is neither rail-to-rail nor 3.3 V; the minimum operating voltages are ±5 V, or 10 V rail-to-rail, with the maximum output at roughly ±3 V under these conditions. However, this is approximately twice the necessary range, which allows undesirable behavior near the maximum and minimum output voltages to be ignored.

As mentioned, another device characteristic to consider is the versatility and offset characteristics of the amplifier. Since the chosen amplifier must be able to function as a unity gain buffer, inverting amplifier, summing amplifier, and activation function neuron, it is crucial that it be able to function well in each of these contexts. Because some amplifiers are not fully stable in unity-gain buffer applications, care must be given to avoid such parts. Similarly, the input bias and voltage offset characteristics must be paid attention to. In smaller networks, such as the 4-pixel test network for this project, only four inputs are summed by the summing amplifier stage of each neuron. With the 10 kΩ feed-in resistors used for the summing amplifier stage of each neuron, the current through each resistor should typically be in the microamp range, and 135 μA at most. When dealing with small signals, these currents may be only a few μA; an amplifier with an input bias current of 1 μA, for example, would cause substantial signal distortion and ruin the neuron's output. As such, JFET amplifiers are effectively mandatory for this project, as typical BJT amplifiers draw too much current to be worth the risk.

Another factor to pay adequate attention to is the number of amplifiers per chip. Since a minimum of two amplifiers are needed per input to each layer for buffering in addition to two amplifiers per neuron for the summing and activation stages, it is highly preferable to use amplifier chips with at least two amplifiers. A reasonable upper limit must be observed, however; cross-talk between signals, while negligible for many applications, must be considered for this project. Additionally, while routing can be greatly simplified by reducing the number of amplifier chips, it becomes more complicated once more if the number of chips drops too low, as significant numbers of resistors must be placed in close proximity to and connected to each amplifier. The ideal number seems, by estimation, to be around 2 to 4 amplifiers per chip, though this is dependent on the device package and the dimensions of the network.

An additional consideration is the slew rate of the amplifier. While the slew rate is not particularly meaningful during standard operation of a trained network, as the signals will overwhelmingly be DC, it is a limiting factor when training the network. Because training is effectively constrained by how fast the digital devices on the network can update,

amplifiers must have an adequate slew rate to allow sufficient time for signals to update and settle. Because there are potentially up to a dozen or even more amplifiers feeding into each other sequentially, this problem is significantly more apparent than in other applications. If training at 10,000 samples per second was desired, for instance, a slew rate of .01 V/µs would not be acceptable, as the signal would not even have time to rise or fall if it were over 1 V, let alone produce a stable value for the entirety of the ADC collection period. Because multiple ADC readings must be made after the network is given sufficient time to stabilize, a safe estimate is that the slew rate be at least 100 times faster than required for the signal to rise. For 10,000 training samples per second, a slew rate of around 1.5 V/µs or higher would be adequate.

Finally, the current driving ability of the amplifier must be carefully considered. As mentioned previously, the maximum current under ideal conditions through each feed-in resistor of the neurons' summing stages is 135 µA; to supply 10 neurons in the following layer, each amplifier must be able to supply at least 1.5 mA once the current draw of the corresponding buffer or activation stage is considered. In reality, the current supplied by each amplifier should be significantly higher than this value, as voltage drops and overheating may occur when devices are pushed near their limits.

3.4.2 Operational Amplifier Selection

For each amplification stage throughout the neural network circuit, we need an operational amplifier. There are many operational amplifiers available to choose from, but we need to choose the component with intention. An ideal operational amplifier for constructing the artificial neural network has a high slew rate, low power consumption and low noise. The operational amplifiers to be considered for selection are the TL084 operational amplifier, the MCP6274 operational amplifier, the TL974IN operational amplifier and the MCP6294IPWR operational amplifier.

The TL084 operational amplifier is a standard and commonly used integrated circuit. It has a high slew rate at about 13-V/µs which will give us a quick response in our output based on our change in the input voltage. This device is also designed to maintain low input and feedback currents, and it requires a minimum of plus and minus 5 V DC rail to rail voltage to operate. This will work for constructing the network, but we will need to come up with a solution for power distribution since the rail to rail voltage is greater than that of its neighboring components, the digital potentiometers and microcontroller in charge of controlling said components. It would be more convenient to find an operational amplifier with rail to rail voltage requirements that are equivalent to the other integrated circuits being implemented.

The MCP6274 operational amplifier is to be considered as well. When looking at the datasheet it seems like a good choice for our project as well. It has a significantly lower slew rate than the TL084 operational amplifier, but may still work well in the circuit. The slew rate is 0.9-V/µs compared to the TL084's 13-V/µs. Tests will need to be done comparing the quickness in output response based on our change in synapse input voltage to see if this is an ideal choice. The main benefit of using this integrated circuit opposed to

the TL084 operational amplifier is that it requires less voltage to power the device. The rail to rail voltage required to operate this device is a minimum of plus and minus 1 V DC to a maximum of plus and minus 6 V DC. This will let us run the power for all of the integrated circuits, digital potentiometers and microcontrollers, off of the same voltage bus. This would ridden the problem of having to use a voltage regulator to maintain proper power distribution throughout the network. This would also give a more simple approach to designing the printable circuit board since there would be only two voltage layers besides the ground layer. The main benefit of using the MCP6274 operational amplifier compared to the TL084 operational amplifier is that it requires much less current and voltage to power on the device in to its operational state.

The TL974IN operational amplifier is the third operational amplifier that is being considered to implement in the neural network circuits. After reviewing the datasheet, it is plain to see that its operating voltage requirements fit within our required specifications. The TL974IN operational amplifier has an operating range of plus and minus 1.35 V DC to plus and minus 6 V DC. This is a great range of voltages because within this range are convenient voltage levels for powering the other integrated circuits on board. The most valuable quality of implementing these operational amplifiers opposed to the others mentioned is that it still maintains a relatively high slew rate with an extremely low noise level on the output. The slew rate is typically 5-V/µs which is relatively high, not as high as the TL084 operational amplifiers slew rate, but should be enough to get the job done correctly. The noise level is typically only 4 nV/√Hz, which should provide us with precise and accurate output responses. This operational amplifier meets all the conditions to be chosen for, and implemented in to the neural network circuits.

The MCP6294IPWR operational amplifier is the last device to be considered for our amplification stages throughout the neural network circuit. The MCP6294IPWR operational amplifier has a typical slew rate of 6.5-V/µs. This slew rate should also be within the bounds needed for the circuit to operate properly. This integrated circuit maintains a low noise output level as well, sitting at 8.7 nV/√Hz, which is a relatively low noise level. The rail to rail operating voltage levels for the MCP629IPWR are plus and minus 1.2 V DC to plus and minus 2.75 V DC. The low input bias current of 1 pA is also to be considered since it is thirty times less than that of the TL084 operational amplifier. These parameters provide a very low power consumption device compared to the TL084 operational amplifier, but needs about the same as the MCP6274 and TL974IN operational amplifier's required operation voltage.

All four of these operational amplifiers will be compared during the testing of each. It is not difficult to choose the best component when just considering the information offered on the datasheets, but testing each device is important. There are always issues that can appear when testing actuality against theory. The price comparison of each component is another factor to be considered, and can be seen clearly in table x.x, which shows the distributor, and price of each component mentioned. The TL974IN operational amplifier seems to be the best choice for the circuit at hand since it has a relatively high slew rate and low enough power consumption to fit in with our other integrated circuit components.

Table 3.1: Operational Amplifier Price Comparison

Device	Distributor	Price
TL084	Texas Instruments	$1.10
MCP6274	Microchip	$1.01
TL974I	Texas Instruments	$0.89
MCP6294IPWR	Texas Instruments	$0.78

Price is not a huge concern for selection of this component, as the difference between the lowest and highest price is 32 cents. This will add up to about a $60 total difference in cost between the cheapest and most expensive operational amplifiers when considering the quantity needed to construct the final neural network. Conveniently, the cost per component is within a relatively low cost margin and there will be no issue funding these components.

3.4.3 Potentiometer Considerations

We are considering a number of factors during potentiometer selection, including the communication protocol used, number of potentiometers per chip, number of taps per potentiometer, and total resistance value of the potentiometers. As mentioned previously, the SPI communication protocol is fairly non-negotiable when selecting chips due to the impracticality of using any other commonly available protocol. Results to date have indicated that 256-tap potentiometers are necessary, as significant difficulty has been encountered when attempting to obtain convergence in networks simulated with 128-tap potentiometers. The number of potentiometers per chip should be as high as possible, as this dramatically reduces both the programming time of the network and the number of components, traces, and vias required on the final PCB. However, cost becomes somewhat prohibitive as the potentiometer count increases, and it is often difficult to find 4 or more potentiometers on a chip without sacrificing some other characteristic, such as affordability or tap count. Finally, the total resistance of the potentiometer should be minimized to a reasonable degree to avoid signal distortion. Because each potentiometer is functioning as a voltage divider between a positive and negative voltage on each end of the device with the wiper selecting the dividing point, it is easily possible for the buffering, inverting, and summing amplifier hardware to pull a non-negligible amount of current from either the positive or negative end of the potentiometer, introducing significant aliasing in the actual output voltage of the potentiometer and causing a breakdown of the ideally linear behavior of voltage as a function of wiper position. However, it is possible to use larger values for the potentiometers by buffering and inverting the output of each potentiometer with op amps with very low input bias currents, minimizing the current draw from the potentiometers and ensuing voltage skew, though this is a fairly hardware-intensive solution.

3.4.4 Potentiometer Selection

Since SPI communication and a high tap count are effectively mandatory, there is not a significant amount of room left for potentiometer selection. While elimination of the

EEPROM (or equivalent) requirement would broaden the range, it is more of a last resort decision than a mere tradeoff. Despite these constraints, a few viable candidates were found which fall within the desired boundaries given for the network's potentiometers.

The first candidate is Microchip's MCP44XX series. For early prototyping and testing, the MCP4441 variant of this device series was used. The MCP4441 is a 10 kΩ, quad-potentiometer I2C chip with 129 taps and onboard EEPROM; a variant with 257 taps, the MCP4461, is also available in this series. The operating range of the EEPROM variants is 2.7 to 5.5 V, with limited operation down to 1.8 V possible. The devices are available in 20-pin TSSOP and 4x4 QFN packages at a reasonable price of about $1.90, or around $.48 per potentiometer, for the MCP4461. The relative ease of placement of the TSSOP package allows for easy PCB integration, but the lack of a through-hole variant necessitates the use of breakout boards to perform breadboard testing. The device's use in this project is severely handicapped by its use of the I2C protocol, since the parallel nature of I2C devices is not practical for even small networks. Consequently, its use was limited to use in transitioning the early neuron prototypes from analog to digital potentiometers, and was not used beyond this point.

The SPI equivalent series of these devices is the MCP43XX series. Due to the satisfactory behavior observed during early testing with the MCP4441 and the similarity of the two series, the MCP43XX series was examined as a candidate for surface-mount PCB implementations. At an identical price to the MCP4461, the MCP4361 sports a price of around $.48 per 256-tap potentiometer, even better than the price of many 128-tap potentiometers available for purchase. Additionally, the fact that the device is a quad-potentiometer chip instead of a dual-potentiometer chip like most of the other candidates is advantageous when dealing with PCB routing and device layout, as it halves the number of chips required for the network's synapses. While the MCP4361 is only produced in 20-pin TSSOP and 4x4 QFN packages, precluding easy breadboard testing, the availability of the TSSOP package makes it a very attractive choice for later surface mount PCB design. However, a critical shortcoming of the MCP43XX series is that it does not support daisy chaining. Since the ability to daisy chain potentiometers dramatically simplifies the board routing and device layout, it is a fairly serious endeavor to control dozens or hundreds of devices without the ability to operate them serially.

The next potentiometer candidate is Texas Instruments' TPL0202. The TPL0202 is a 10 kΩ dual-potentiometer SPI chip with 256 taps and onboard EEPROM. The device has an operating range of 2.7 to 5.5 V and is produced in a 4x4 QFN package. Aside from the sole availability in a QFN package, which is difficult to solder, this device meets the technical requirements of this project. One of the primary non-technical drawbacks to this device is that, due to its sole availability in a dual-potentiometer package, the cost per potentiometer is relatively high at around $.92, almost double that of the MCP4461. The 4x4 QFN package also complicates routing and soldering, and the doubled number of chips significantly increases the overall footprint and design complexity of the network. Additionally, since QFN packages are difficult to solder without a screen, it is difficult to even attach the device to a breakout board for breadboard performance testing. Finally, as

with the MCP43XX series, daisy chaining is not supported by the TPL0202, greatly limiting its usefulness. As a result, the TPL0202 was not chosen for this project.

Another candidate is the MCP42010, a 10 kΩ dual-potentiometer device produced by Microchip. The MCP42010 is a 256 tap SPI chip produced in 14-pin PDIP, SOIC, and TSSOP packages. A significant difference between the MCP42010 and the other candidates is the lack of EEPROM on the device; while this is not a disqualifying factor, it is a significant complication as it would force the network to be reprogrammed after each power cycle. Additionally, the relatively high price of around $1.00 per potentiometer is one of the highest prices of the potential candidates, which is a significant hurdle considering hundreds of potentiometers would be required for even a relatively small network. However, the availability of the device in a PDIP package makes it an attractive choice for breadboard testing and for early through-hole PCB design, as in the 4-4-3 test network. Importantly, the MCP42XXX series supports SPI daisy chaining, greatly simplifying the routing and external hardware and logic required to operate the network. Consequently, the MCP42010's PDIP variant was chosen for use in the through-hole PCBs of the 4-4-3 test network, and the TSSOP variant is a good choice for later surface mount implementations.

Finally, Analog Devices' AD5204 is being considered. The AD5204 is a 256 tap quad-potentiometer SPI chip which is available in 10 kΩ, 50 kΩ, and 100 kΩ varieties. The device is produced in SOIC, TSSOP, and LFCSP packages, complicating breadboard testing, though the TSSOP is an ideal choice for surface mount layouts. While the resistance tolerances are 30% versus the standard 20% for most digital potentiometers, this can be minimized as discussed in the "Potentiometer Considerations" subsection. Because the AD5204 contains 4 potentiometers per chip instead of 2 and supports daisy chaining, it requires half of the overall footprint of the MCP42010 at approximately the same cost per potentiometer. Because this simplifies routing and reduces the overall size of the network, the AD5204 is a likely choice for the final network's potentiometers.

Because there is a significant amount of overlap among the potentiometer candidates and a fairly wide range of factors to consider, it is helpful to summarize both the technical and logistical considerations of the parts to simplify the decision process and make comparisons easier. These summaries are provided in Table 3.2 and Table 3.3.

Device/series	Manufacturer	Resistances	Tolerance	Tap counts
MCP44XX	Microchip	5 kΩ, 10 kΩ, 50 kΩ, 100 kΩ	±20%	129, 257
MCP43XX	Microchip	5 kΩ, 10 kΩ, 50 kΩ, 100 kΩ	±20%	129, 257
MCP42XXX	Microchip	10 kΩ, 50 kΩ, 100 kΩ	±20%	256
TPL0202	Texas Instruments	10 kΩ	±20%	256

AD5204	Analog Devices	10 kΩ, 50 kΩ, 100 kΩ	±30%	256

Table 3.2: Potentiometer Technical Consideration Summary

Device/series	Pots. per chip	Price per pot.	Protocol	SPI daisy chaining?
MCP44XX	4	$.30	I2C	N/A
MCP43XX	4	$.40	SPI	No
MCP42XXX	2	$.86	SPI	Yes
TPL0202	2	$.78	SPI	No

Table 3.3: Potentiometer Logistical Consideration Summary

The range of values for each device series is provided, where applicable; prices per potentiometer are based on present prices from Mouser assuming an order of 100 potentiometers. One of the most striking differences is in price between the MCP43XX/44XX series and the other candidate devices, at a difference of more than 2-to-1; however, neither device is capable of SPI daisy chaining, indicating there is clearly a premium on the shift register architecture necessary to allow for daisy chaining. While cost is a fairly high priority considering the The difference between the top two candidates, the MCP42XXX series and the AD5204, is minimal; a price difference of only 1 cent separates the two. However, the higher potentiometer count per chip for the AD5204 more than offsets the higher tolerances, the effects of which can be minimized. Consequently, the AD5204 is clearly the best choice once all factors are brought into consideration.

3.4.5 Digital Potentiometer Controller

The controller for our digital potentiometers, and all other components that need digital data inputs to function, has a few requirements to be considered ideal. It needs to have a floating point hardware unit, at least 7 analog inputs with atleast 12 bit ADC (analog to digital converter) resolution, and SPI (serial peripheral interface) and I2C (inter-integrated circuit) communication is preferred. It will also require a minimum of 10 GPIO pins, and a C compiler to handle the programming. Some microcontrollers that we are considering are the Teensy 3.5 Development Board, the TIVA C Series TM4C1294 Launchpad, the STM32 Nucleo-64 development board with STM32F030R8 MCU and the STM32H743 MCU. We will be closely analyzing each device in the following paragraphs of this section.

The Teensy 3.5 Development Board will be a solid choice in theory, as it meets all of the necessary requirements to control our network. It has worked successfully for basic prototyping of the overall network as it was readily available to us during prototyping of the smaller scale network. The Teensy 3.5 Development Board has a 120 MHz ARM Cortex-M4 with a floating point hardware unit, digital input/output pins with 5 volts DC tolerance, and twenty five analog inputs with two 13-bit resolution ADCs. The CPU on board will support C programming language. For communication, the board includes three serial peripheral interface ports and three inter-integrated circuit ports. The development board also requires 3.3 to 6 volts DC to operate, which fits within the bounds of the supply voltage requirements for the operational amplifiers and digital potentiometers being used. With a total of 62 digital input/output pins, there will be more than enough pins to work with if this device is to be implemented. The Teensy 3.5 Development Board is

manufactured by PJRC, and can be obtained on digikey.com. The Teensy 3.5 Development Board costs $31.25 which is affordable and within our budget.

The TIVA C Series TM4C1294 Launchpad is the second device we are considering to control the entirety of our network. The Launchpad has a 120MHz 32-bit ARM Cortex-M4 CPU on board that will support C programming language. Other features include eleven GPIO pins, serial peripheral interface and inter-integrated circuit communication, eight analog input pins, and two 12-bit resolution analog to digital converters. The device can use a supply voltage between 3.3 and 5 volts DC to operate which fits within the supply voltage bounds of all operational amplifiers and digital potentiometers the network is using. This is a convenience that is not necessary, but will optimize power distribution of the network. The TIVA C Series TM4C1294 Launchpad is manufactured by Texas Instruments and can be ordered online from ti.com. The TIVA C Series TM4C1294 Launchpad costs $19.99, which is also within our budget, but is cheaper than the Teensy 3.5 Development Board. The approximately twelve dollar difference is not significant enough to value the cost of the TIVA C Series TM4C1294 Launchpad over the Teensy 3.5 Development Board so both controllers are still to be considered.

Another microcontroller being considered is the STM32 Nucleo-64 development board with STM32F030R8 MCU. On the board is an ARM Cortex M4 32-bit STM32F401RET6 microcontroller, which will suffice as a CPU as it is compatible with C programming language. The STM32 Nucleo-64 development board also has a floating point hardware unit and six analog inputs. Seven analog inputs is preferable, but the STM32 Nucleo-64 development board is compatible with Arduino boards which can be used to extend input availability. The controller also has three serial peripheral interface ports and three inter-integrated circuit ports which satisfy our digital communication needs. Another included feature is the 12-bit resolution analog to digital converter. The STM32 Nucleo-64 development board requires 3.3 to 5 volts DC to operate which fits the supply voltage bounds of the other active components that make up the network. The STM32 Nucleo-64 development board is manufactured by STMicroelectronics can be obtained via online order from st.com. The price of the STM32 Nucleo-64 development board is a mere $13 when compared to that of the Teensy 3.5 Development Board and the TIVA C Series TM4C1294 Launchpad, but the price is still not our main concern.

The final microcontroller we are considering to mangage all hardware operations of the TACOCAT network, is the STM32H743 microcontroller with an ARM Cortex-M7 core. This microcontroller is ideal because of the suitability of its GPIO pin count, core clock speed, and physical package type for controlling the TACOCAT network. For a 100-input neural network, an MCU with a minimum of 120 GPIO pins and a sufficiently high clock speed is required. With its 140 available GPIO pins, the STM32H743 has a sufficient pin count to both drive 100 pixel input lines and handle communication and ADC functions; additionally, its maximum clock rate of 480 MHz is fast enough to ensure minimal latency for training and recognition options. The STM32H743 is available in a quad flat-pack package, which makes it easier to work with than other large microcontrollers which are commonly available in ball grid array packages. The comparison of each component can be seen below in Table 3.4.

Device	Manufacturer	Price
Teensy 3.5 Development Board	PJRC	$31.25
TIVA C Series TM4C1294 Launchpad	Texas Instruments	$19.99
STM32 Nucleo-64 Development Board	STMicroelectronics	$13.00
STM32H743	STMicroelectronics	$12.29

Table 3.4: Comparison of Digital Potentiometer Controller Devices

For essential prototyping of our intermediate network, we are implementing a Teensy 3.5 Development Board to communicate with the potentiometers using serial peripheral interface (SPI) communication. This controller provides us with all of the necessary digital outputs for communication while only using 3.3 volts DC to power the device. These necessary digital outputs include: a high frequency clock signal, serial communication output and chip select output. The pin out for the Teensy 3.5 Development Board can be found in the appendix in Figure X.X. A picture of the physical device can be seen below in Figure 3.11.

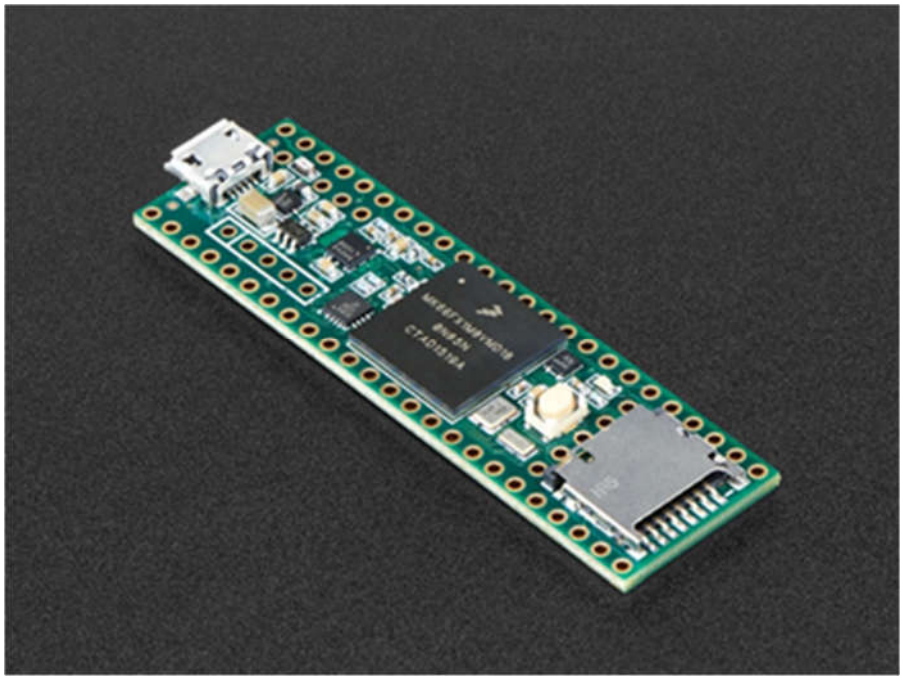


Figure 3.11: Teensy 3.5 Development Board

3.4.6 Communication Protocols

The long-running universal asynchronous receiver-transmitter (UART) standard has been of some interest for embedded communications for this project due to its ubiquity and simplicity. However, while it may be useful for controlling an output indicator or for bidirectional communication with the network with an external source, it is unlikely to be

useful for communications among the digital devices of the network itself, as these chips are generally designed for I2C or SPI communication. Thus, UART communication is of limited utility for this project outside of controlling certain peripherals.

Inter-Integrated Circuit (I2C) serial communication is a widespread and extremely useful serial communication protocol for embedded systems and device control. Unlike UART, which communicates via a single serial line, I2C communication buses contain two lines: a clock line and a data line. As multiple devices are connected to the same data line, each command begins with an address so that the correct device responds to the given command. As multiple devices can be connected in parallel to the same serial data line with multiplexing inherent in the protocol, I2C is an attractive choice for communications with the potentiometers and other devices in the network. However, as each device requires its own slave address, there is a relatively low upper limit on the number of devices which can be connected to a single data line. Because of the scale of this project, this necessarily indicates that either additional external multiplexing is necessary on the line or that multiple data lines must be used. Because most of the potentiometer chips observed only have two address bits, allowing a total of 4 devices to be operated simultaneously on one data line, this could potentially mean dozens of separate I2C buses that would need to be separately activated when necessary, greatly complicating both the training algorithm and the external hardware for the circuit. Thus, I2C is not a practical choice for controlling large arrays of devices that do not have a convenient method for assigning unique slave-device addresses.

Serial-Peripheral Interface (SPI) serial communication is similar to I2C communication, but with a few key differences. SPI communication uses three wires: a chip select line, a clock line, and a data line. Chips are activated when their chip select pin is pulled low, indicating that they should begin reading from the data line. However, unlike with I2C devices, every device has both a serial input and a serial output pin. If the chip select of the devices are all connected to the same chip select line from the microcontroller and the serial output of each device is connected to the serial input pin of the next device in line, it is possible to “daisy-chain” a significant number of devices together, causing them to function as an extremely large shift register. Each time a write command is given to the network, each device writes its contents to the next device in line; this process allows a single microcontroller to potentially program dozens of chips with a single data line and no additional external hardware for multiplexing.

Additionally, because each device is only responsible for driving the device after it, there is no significant risk of data attenuation as the size of the network grows. As long as the clock and chip select signals can be maintained, the upper limit of daisy-chaining is fairly arbitrary. Because a single controller can simply output the write commands and data in the order they appear in the network, from farthest to nearest, there is no significant penalty to speed compared to another approach such as multiplexed I2C communication. Thus, SPI is by far the ideal choice for controlling a large array of digital potentiometers due to its low footprint and reliable operation.

A sample SPI master-slave configuration, similar that which will be used for communication with the digital potentiometer controller and the potentiometers is shown

in Figure 12. It should be noted that this is a parallel configuration; in a daisy-chain configuration, the SDOx pin of each slave would be connected to the SDIx pin of the following slave, with only a single I/O connection from the SPI master to the first slave's SDIx pin.

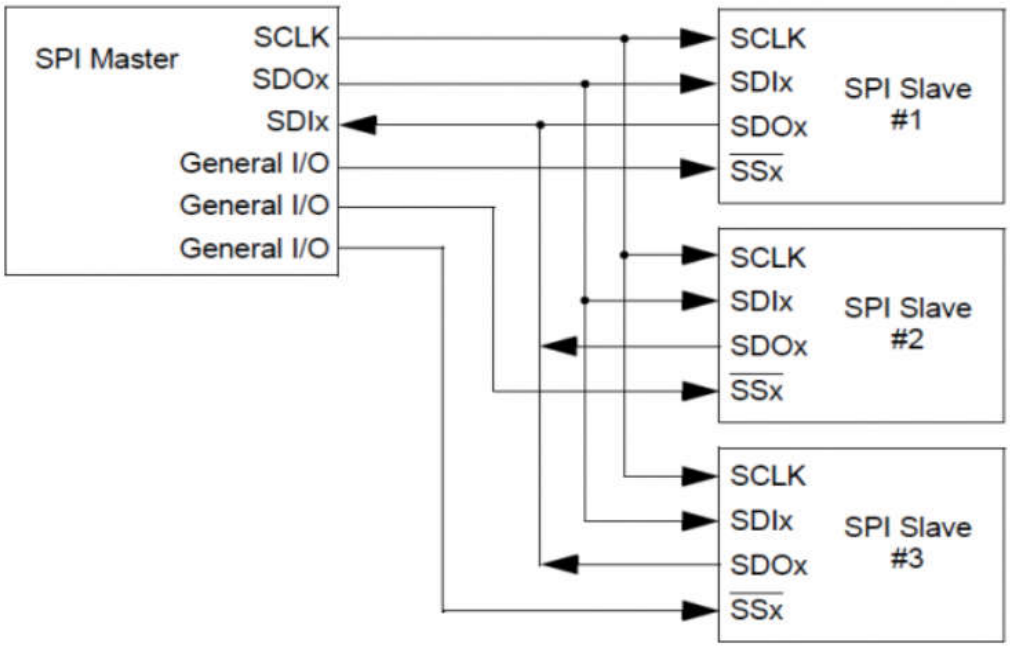


Figure 3.10 SPI Master Slave Configuration Sample. Reproduction Permission requested from https://electrosome.com/

3.4.7 Voltage Isolation for External Communication

Electronic communication networks are based on the transmission of information as an electrical signal that is propagated through a circuit. In many small-scale communication networks, the different devices that are attached to the network will at least have their power supplies tied to a common ground, which facilitates the creation of network circuits.

For TACOCAT's communication network, the microcontroller device that controls the low-level procedures for training and prediction operations operates on the same split-rail power supply as the analog neural network itself. While the microcontroller and other digital components in the neural network such as the digital potentiometer control circuits share a common ground connection, the microcontroller is also expected to communicate with a PC or mobile device that sends user-input data and may receive results or meta-data for performance analysis. This "user-interface device" is unlikely to be powered by the same split-rail power supply as the neural network, which has a nominal maximum voltage of 3.3V between the positive and negative rails.

One way to establish communication links between devices with different ground levels is to connect two device's network circuits together through a pair of matching nodes that are separated by an insulating layer, but able to communicate with each other using a signal

transmission medium that does not require an electrical connection. Assuming that I2C or an I2C-derived protocol will be used for external communication, the link needs to provide at least one bi-directional channel (for serial data) and one unidirectional channel (for the serial clock). Because the I2C slave device might employ clock-stretching, we will only consider devices that support bi-directional communication for the clock signal as well as the data signal.

Optocoupler devices, containing a light-emitting device and a light-dependent resistance, have been used to perform this type of isolation in the past, but modern digital isolator designs have largely taken their place. These digital isolators use a variety of electromagnetic waves or fields to transmit data without a completed electrical circuit. Table 3.5 shows a comparison of several different digital isolators that are marketed specifically for use with I2C-based communication links.

Device	Manufacturer	Min/Max Supply Voltage	Maximum Clock Rate	Isolation Technology	Price
ADUM1250	Analog Devices	3.00 - 5.5 V	1 MHz	Magnetic Transformer	$5.55
MAX14933	Maxim	2.25 - 5.5 V	1.7 MHz	Not Stated	$3.00
ISO1540	Texas Instruments	3.00 - 5.5 V	1 MHz	Capacitive	$4.66
SI8400	Silicon Labs	3.00 - 5.5 V	1.7 MHz	Radio Frequency	$3.09

Table 3.5: Comparison of Digital Isolator Devices

The datasheets for all of the devices listed in the comparison table recommend adding a small capacitor across the power-supply rails on each side of the chip, but no other additional components are typically required. Considering that the expected clock rate for external communications is not expected to exceed 1 MHz, all of the devices that were considered should meet the specifications for maximum clock rate. All devices should also be compliant with the expected supply voltages of 3.3V on each side of the IC.

There may be other issues related to performance, reliability, and efficiency that differentiate the devices, but they are not readily apparent from looking at the device datasheets. Fortunately, because the different devices can easily be substituted for one another, prototype testing can be conducted conveniently with several different device models to see if there are any tradeoffs between price and operating characteristics.

3.4.8 Power Distribution and Regulation

After selecting our microcontroller used for communication with the digital potentiometers, it occurred to us that we should apply a voltage regulator to protect our microcontroller and the circuit itself. Based on which operational amplifier is used, we may have to distribute supply power differently. If the MCP6274 operational amplifier, TL974IN operational amplifier, or MCP6294IPWR operational amplifier are going to be used, then the power should be able to run off the same voltage lines that power the digital potentiometers and microcontroller. Since the required supply voltage is within the bounds of the supply voltage required for the digital potentiometers and microcontroller this would

be convenient. The TL084 operational amplifiers require a minimum of plus and minus 5 volts DC, which does not match with the voltage required to power the digital potentiometers and microcontroller. In this case, a dual power supply voltage regulator must be implemented to take the –1.66 Volts DC and +1.66 Volts DC lines and boost them to – 5 Volts DC and + 5 Volts DC. This is an option, but can create issues when designing the printed circuit board of a singular neuron circuit. Instead of using three voltage planes, plus and minus 1.66 Volts DC and ground, there would be five. The addition of the plus and minus 5 Volts DC creates two more voltage bus layers that will be needed.

Since TL084 operational amplifiers are readily available to us, our group decided to do some breadboard prototyping with these integrated circuits in the mean time. Since a dual power supply voltage regulator was essential for proper power distribution, the LM317 & LM337 precision voltage regulator was decided upon for prototyping. This voltage regulator fits our desired input and output voltage ranges, and is highly affordable. The voltage regulator is being used as a dual power supply to regulate our bipolar power configuration. It can be seen in Figure 3.11. A –1.66 V DC to +1.66V DC voltage range will be applied to the digital potentiometers and the Teensy Development Board to power the devices. The final components that need power to operate are the TL084 operational amplifiers mentioned above. For these amplifiers, we will be using -5 V DC to +5 V DC rail power supplies. Our overall power distribution is routed as follows. Starting with one -5 V DC bus and one +5 V DC bus from a DC Voltage power supply, wires will be jumped to the plus and minus rail terminals of each operational amplifier. Then, running the +5 V DC and -5 V DC in to our dual power supply from the DC voltage power supply. This converts the plus and minus 5 V DC in to a respective +1.66 V DC and - 1.66 V DC. These plus and minus 1.66 volts DC voltages are then run in to their own bus which we can distribute power to the Teensy 3.5 Board and all of the digital potentiometers from.

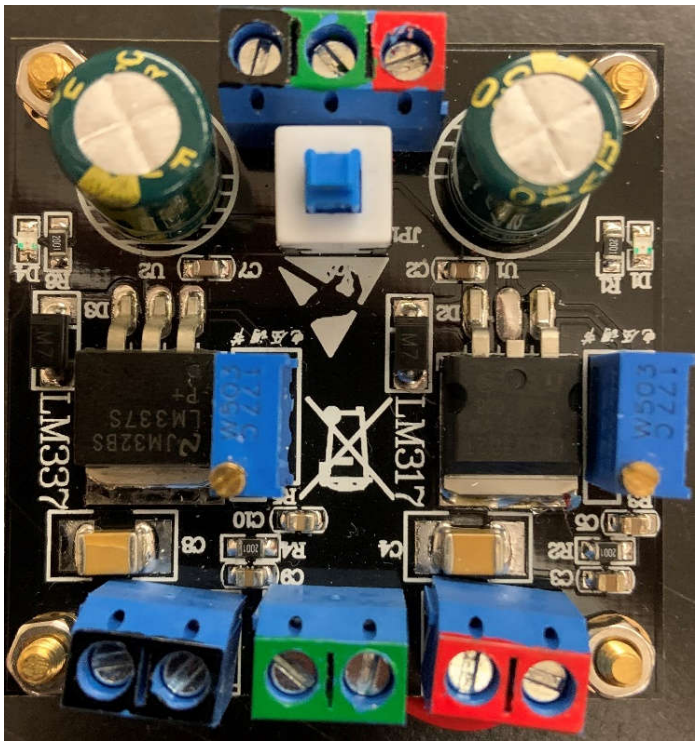


Figure 3.11: LM317 & LM337 Precision Voltage Regulator

In the future, different operational amplifiers may be implemented, which will require us to reroute our power distribution in a simpler fashion. In this case, the amplifiers will be chosen to match the rail to rail voltage of the digital potentiometers and Teensy 3.5 Development Board. This will make the printed circuit board design much simpler since there will not be two different voltage planes to be considered. After testing with the TL084 and acquiring promising results, we must make sure that introducing new operational amplifiers to not degrade previously obtained results. If possible, it would be ideal to implement operational amplifiers with lesser supply voltage requirements while still providing the same promising results of our system.

3.4.9 Programming Languages

The codebase for this project was divided into sections that are run on two separate platforms for different purposes: one is a simulation/training and user-interface program group that runs on a minimal PC platform, and the other is firmware that runs on a ARM microcontroller platform.

Python was chosen as the programming language for the simulation/training software due to several of its qualities that also make it popular as a language for other research in machine learning. Python has strong support for vector/matrix math operations, which is provided by the NumPy library. Python also supports functional programming constructs such as list comprehensions and lambda functions that simplify the manipulation of large data sets.

C/C++ was chosen as the programming language group for the implementation of firmware functionality due to its efficiency and reliability. The option of compiling Python code into

binary files that could be executed on a microcontroller, via the MicroPython project's code library, was considered, heavy reliance on a toolchain that is still in the development stage seemed like too big of a risk to take in the firmware design plan.

The C/C++ toolchain for ARM microcontrollers is mature and comes with few limitations relative to the C/C++ toolchains for common personal-computer CPU architectures.

3.4.10 Handwritten Character Data Set

Recognition of handwritten numeric characters in the Modified NIST (MNIST) data set is a common benchmark for machine-learning algorithms. This data set, described in LeCun et al., consists of image samples of handwritten numeric characters from '0' to '9' that were provided by the U.S. National Institute of Standards and Technology. Roughly half of the samples in the data set were gathered from U.S. Census Bureau employees, and the remainder of the data set was collected from samples submitted by high school students.

The MNIST set is divided into two distinct subsets, one intended for training and another for testing. Images were cropped from the original data at a size of 20 x 20 pixels, and these images were centered in 28 x 28-pixel squares. The images contain 8-bit pixel values representing grayscale levels ranging from 0 to 255.

In this work, we chose to use the Extended MNIST (EMNIST) data set, which is a close relative of MNIST, is described in Cohen et al. This dataset follows the same sample formatting and file structures as MNIST, and while MNIST contains only numeral characters, EMNIST contains numeral and alphabetic characters. Like MNIST, EMNIST is also divided into distinct training and testing subsets.

3.4.11 Diode Component Selection

Inside the artificial neuron circuit configuration that we are designing for the MLP neural network, the last activation function stage operational amplifier will need to implement the hyperbolic tangent activation function. We are achieving this by utilizing a rectifier diode bridge on the feedback of the operational amplifier. Diodes are used for waveform manipulation in the way of voltage rectifiers, clamping and clipping circuits. The purpose of the diodes in this circuit will be to rectify the weighted and summed inputs from the summing amplifier circuit and clip the extremes of the output waveform.

To choose an appropriate diode for this application, a range of voltage for which to clip is required. Since the range of output voltages for the multi-layer perceptron needed is -1.65V to +1.65V, a diode with a 2V turn-on voltage would be enough if the circuit was designed with diodes both in forward and reverse current directions on the feedback loop of the activation function operational amplifier. There are several options for types of diodes to use for this purpose i.e. Zener diodes, rectifier diodes, germanium or silicon diodes, PN diodes, Schottky diodes, etc.

The first possible type of diode to choose from, Zener diodes. These types of diodes are typically used for reliable voltage reference applications. They also have a higher turn on voltage than a regular diode would, somewhere in the range of 5V, which would be too large for the activation function stage of our neural network circuit design. Furthermore, to turn on the diode would require it to be reverse biased. This would be inconsequential to

the application, but it would require a different configuration than the one designed in the circuit schematic of the 2x2 network design.

Schottky diodes would be another option of diodes from which to choose from for the signal rectification of the activation function operation amplifier circuit. However, Schottky diodes tend to have a much lower turn-on voltage than that of PN junction diodes. They have forward voltage somewhere in the range of 0.15V to 0.4V as opposed to a typical 0.6-0.7V forward voltage of a PN junction diode. This would mean that many would be required to be placed in series for the rectifier bridge of the activation function operation amplifier circuit and would be less efficient than finding a more appropriate singular diode with a larger forward voltage for this project's application.

Small signal diodes offer perhaps the smallest package size while offering a fast switching rectification with low leakage and high reliability. Specifically, the 1N4148 small signal diode from ON Semiconductor could be used for the rectifying diode bridge of the activation function circuit design. It would offer an affordable option with a 1V forward voltage, despite it having a low acceptable current of 200mA. At around $0.10 per unit, it's a good contender for this application as it would over a reasonable 0.2W maximum power dissipation, and fast signal switching speed. However, rectifier diodes from Vishay Semiconductor of the 1N400X family would offer a more affordable price per unit, with a higher power dissipation capability, while having a lower signal switching speed.

The final and most appropriate diode type for the activation function circuit is a rectifier PN junction diode. This is because of its use of only allowing current to conduct in one direction, and typical forward voltage of around 1V. For the activation function circuit, a 1V forward voltage rectifier diode would call for the need of at least 2 identical diodes to be connected in series in both forward and reverse directions on the feedback loop of the operational amplifier to achieve positive and negative voltage rectification.

Many rectifier diodes with 1V forward voltage exist in the electronics market, though they have different maximum forward-bias currents. For this project's purposes, a 1-ampere rated rectifier diode would be more than enough considering the low current draw of the network. The 1N400x family of 1-ampere general-purpose silicon rectifier diodes are commonly used for AC signal rectification for applications similar to the clipping performed in this project. Compared to the 3-ampere family counterpart, the 1N540x, they are less expensive due to their lower heat dissipation requirements and more appropriate for the much lower expected currents in the activation function's analog circuit.

Finally, the 1N4001 rectifier diode was chosen for its price trade-off compared to the 3A counterpart, and its ease of integration with this project's activation function circuit. Despite it having a lower switching speed than the small signal diodes of the 1N4148 variety, it is a specification that is traded off for a higher power dissipation ability, since that is a more concerning specification. A summary of all diodes considered for the activation function operational amplifier circuit can be found in Table 3.6.

Device	Manufacturer	Forward Voltage	Forward Current	Diode Type	Starting Price per Unit
1N4001	Vishay Semiconductors	1.1V	1.0A	Rectifier	$0.07
1N5400	Vishay Semiconductors	1.2V	3.0A	Rectifier	$0.10
1N5819	Vishay Semiconductors	0.55V	1A	Schottky Rectifier	$0.37
1N4733A	Micro Commercial Components	5.1V	200mA	Zener	$0.32
1N4148	ON Semiconductor	1.0V	300mA	Signal Fast-Switching	$0.10

Table 3.6: Diode Component Selection Technical/Price Comparison

3.4.12 Shift Register Component Selection

Shift register IC will be used to provide the input data to the input layer of the final MLP neural network circuit design. This is being done to avoid having a large number of parallel traces being run from the microcontroller that processes the input image data, to the synapse circuits of the neurons from the input layer. To overcome this problem, serial-input serial-output shift registers will be used.

Serial-input (SI) shift registers benefit from the fact that they can be connected to peripheral circuits via the serial communication protocols used. In the case of the final circuit design, SPI serial bus and clock signal from the microcontroller chosen will be used to feed the input pixel data to the necessary CMOS hex inverter and buffer chips, which will in turn set the voltage levels of the PB0 and PA0 pins of the digital potentiometers. To integrate the SI Shift registers, when considering the 25-input final network design, at least 2 16-bit SI registers or 4 8-bit SI shift registers must be utilized.

There are some technical specifications to be taken into account when choosing the right shift register for this application. The first of which, is the number of bits desired for the shift register's storage register. A larger number of bits allowed in the storage register will call for a lesser number of chips for the input layer of the neural network, as the number of chips needed is directly related to the number of input pixels obtained from the input image. Therefore, careful price consideration for the shift register chip needs to be taken as a lower total number of chips needed would benefit the cost of the design if it can be done using a 16-bit registers over 8-bit registers.

As previously mentioned, the SI shift register IC is the main shift register type being considered for this application. This is due to the parallel-input shift register types being obsolete for the input layer design of the final neural network circuit. The purpose of using the shift registers for this design is to avoid having a number of parallel input traces to the input layer of the network equal to the number of inputs from the pixel data. If the input is of large pixel quantity, it would be more efficient use of PCB space to allow the serial bus

of the SPI interface to feed the input data to the input layer of the network. Having a parallel-input shift register would defeat this purpose as it would encounter the same problem with the number of traces. Furthermore, having parallel-input shift register would mean that the same number of traces would be needed as not having obtained the input data serially would in the first place, making this option obsolete. Thus, the SI shift register type is optimal for the input layer.

Next, the communication protocols available to the shift register chip need to be considered. Ideally, a shift register with SPI communication capabilities would be the choice for the input layer of the final circuit design, over one with I^2C communication. This is due to the rest of the circuit design utilizing SPI communication buses instead of I^2C buses, as the communication protocol used by the digital potentiometers of choice for the weights of the synapses use SPI communication. Thus, the data in the serial data bus will be providing the serial input bits to the chosen shift register IC.

Supply voltage ranges for the shift register is perhaps the most important technical consideration for choosing the right IC for input layer of the final neural network design. The voltage range needed for the inputs to the digital potentiometers will be directly affected by the logic high and logic low levels used by the shift register chip. Several shift register IC's use high and low logic levels that are proportional to the supply voltage it is provided with. This means that carefully choosing one with the right built in hysteresis between high and low logic levels is needed to find one that will fit the input voltage range needed of -1.65V to +1.65V to the digital potentiometers.

Certain shift register options in the market have the benefit of including a chip select pin, which can disable the shift register clock and the storage register clock, placing the serial data read in a high impedance state. The serial data input and read pin provides the ability to read the stored data in the storage register of the chip, which allows for ease of troubleshooting the device, since reading the data stored can be done in a recirculating loop. These types of shift registers have four basic modes of operation:

- Hold (no operation performed)
- Write (via serial input)
- Read (via serial output)
- Load (via the serial input data, stored in parallel to the register)

This capability is a reason to be cautious, however, as chips with this configuration can result in a false clocking of the shift register data via the chip select line, if it goes logic-low. Typical maximum clock rate should also be taken into account, as the serial data clock rate would be driven by the clock edges of the microcontroller used to process the input pixel data. A detailed summary of technical considerations for candidate SISO shift register chips from different manufacturers can be seen in Table 3.7.

It should be noted that the low and high voltage levels of the devices chosen above are measured with respect to the ground terminal of the network the device is to be included in, meaning that the high and low logic levels can be adjusted by supplying the ground and V_{CC} terminals of the shift register chips with an appropriate voltage level range to match the range needed for the inputs to the digital potentiometer weights of the input layer.

Device	Manufacturer	Low Level Output / High Level Output	Storage Register Size	Nominal Supply Voltage	Starting Price per Unit
TPIC2810	Texas Instruments	$0.3V_{CC}$ $0.5V_{CC}$	8-bit	3V ~ 5.5V	$1.56
MC74HC165A	ON Semiconductor	0.1V - 0.4V 1.9V - 5.9V	8-bit	2V ~ 6V	$0.099
SN74LS165AD	Texas Instrument	0.2V - 0.4V 2.4V - 3.4V	16-bit	5V	$0.74
74HC165D	Nexperia	0.1V - 0.4V 5.2V - 5.81V	8-bit	2V ~ 6V	$0.37
74HC595	Texas Instruments	0.002V – 0.4V 1.9V – 5.8V	8-bit	2V ~ 6V	$0.15

Table 3.7: Shift Register Component Selection Technical/Price Comparison

3.4.13 CMOX Hex Inverter Component Selection

The modified design for the input layer of final neural network design, containing the SISO shift register chip, calls for a way to invert the input digital signals to the PB0 and PA0 pins of the digital potentiometer, such that an appropriate range of weight voltages can be achieved. To complete this task, there needs to be a complementary voltage to both aforementioned pins of the digital potentiometer. The use of a CMOS hex inverter will allow for the complement of the corresponding digital logic voltage level input to the neurons of the input layer.

CMOS hex inverters consist of 6 input logic-level input voltages and 6 logic-level inverted output voltages. There are several important technical considerations for choosing an appropriate CMOS hex inverter for this application, first of which, is the supply voltage. To keep a simple and compact PCB and final neural network circuit design, it is desirable to keep the supply voltage of the peripheral circuit IC's in the input layer of the neural network to run off of the same supply voltage line (V_{CC}) as the other components of the network. This would mean that it would be optimal to try to choose a hex inverter chip with a supply voltage in the range of about 5V, if it is to be kept at the same rail voltages as the operation amplifiers in the network, or 3.3V if it is to be kept at the same voltage level as the digital potentiometers of the network.

CMOS inverter chips are sometimes sold as CMOS gate packages that don't always include hex inverters (6 inverter circuits). These are not the ideal IC to select for the application of this circuit, as it would mean an inefficient use of PCB space. A larger quantity of hex inverter chips would be needed to complete the task of inverting all the digital logic-level inputs to the synapses of the input layer. CMOS hex inverters are ensured to include 6 pairs of input and complementary (inverted) output voltage at logic-level high when the input is triggered.

The minimum edge rate of the hex inverter chips would be an important technical specification to consider if the fast switching voltage speed was needed. For example, the

74AC04 hex inverter chip from Fairchild Semiconductor offers a minimum input edge rate of ~125mV/ns. At the expected edge clock rate of 20MHz, the minimum edge rate of the chip would certainly be met, and thus, it isn't going to be a limiting factor for choosing an appropriate hex inverter for the input layer, as the serial data would meet the requirement.

A survey of available hex inverters is provided in Table 3.8.

Device	Manufacturer	High Level Output Voltage	Nominal Supply Voltage	Starting Price per Unit
74LS04	Renesas	0.4V – 2.7V	4.75V – 5.25V	$0.69
CD4069UBE	Texas Instruments	4.95V – 15V	3 – 18V	$0.28
74HCT04	Texas Instruments	0.1V – 4.4V	4.5 – 5.5V	$0.44
74AC04	Fairchild Semiconductor	2.9V – 4.86V	-0.5V – 7V	$0.19

Table 3.8: CMOS Hex Inverter Technical Considerations Summary

The high output voltage level range of the hex inverter chips is perhaps the most significant technical consideration for choosing an appropriate component for the synapse circuit. Since the voltage levels on the PB0 and PA0 pins of the digital potentiometers in the weighted synapses of the input layer have a minimum voltage level requirement of -1.65V to +1.65V, the high level output voltage range capability of the hex inverter chip chosen must take this into account. Upon analysis of the data shown in Table 3.8, the Texas Instruments 74HCT04 Hex Inverter IC seems like the most appropriate choice for the input layer design of the final neural network. This is due to the nominal supply voltage range being the same as the intended supply voltage of the operational amplifiers used for the neurons, as well as the high level output voltage range having sufficient enough leeway to accommodate for the voltage range needed by the digital potentiometers of the synapses.

3.4.14 Touchscreen Interface Selection

The way that our neural network will receive an input to recognize, is through a touchscreen input. Users will be able to write a letter on the touchscreen device and the neural network will be able to decipher what they wrote. This touchscreen interface will be connected to the Raspberry Pi, which will be converting the image in to a ten by ten-pixel array that represents the character the user intended the machine to guess. These inputs will then be fed in to the network for processing. Some touchscreens that are being considered for selection are the Raspberry Pi 10.1-inch LCD (B) Touchscreen Display Capacitive Touch Screen Monitor, the Raspberry Pi 7-inch HD IPS Capacitive Touch Screen Display, the 4-inch Raspberry Pi LCD Resistive Touchscreen Monitor TFT LCD, and the 5-inch Raspberry Pi LCD Touchscreen Monitor TFT.

The Raspberry Pi 10.1-inch LCD (B) Touchscreen Display has a maximum resolution of 1280x800 and is a touchscreen that is officially supported by Raspberry Pi. This ten inch touchscreen interface seems to be a perfect size for our project, but the larger size comes with a larger cost. This interface will definitely feel the best when using, and look the best

overall compare to the other options. Each touchscreen interfaces price will be taken in to consideration before purchasing.

The Raspberry Pi 7-inch HD IPS Capacitive Touchscreen Display has a 1024x600 pixel maximum resolution. This will more than satisfy the needs for our neural network. It also has a USB interface for communication which will handle data and power signal flow. The seven inch touchscreen seems to also be a considerable choice for a touchscreen that is supported by Raspberry Pi, and this size should be fine for what we are trying to implement. The price of this touchscreen is less than that of the ten inch touchscreen interface while still being big enough to be comfortably drawn on.

The four-inch Raspberry Pi LCD Resistive Touch Screen Monitor TFT LCD is the third touchscreen interface to be considered for our project. This device has a resolution of only 480x320 when compared to the other two larger touchscreen interfaces, but this is still more than enough to get the job done. Since we are only considering a ten by ten pixel input, this should not be a concern. This is by far the cheapest component to select, but four inches is rather small, even tinier than most smart phones nowadays. This is a possibility, but probably not the best option when taking user comfort in to account.

The five-inch Raspberry Pi LCD Touchscreen Monitor TFT is the last touchscreen interface device being considered for use within our project. The five inch touchscreen display has a maximum resolution of 800x480 which will also provide us plenty of room to work with. The prices of each touchscreen interface and where to obtain them can be found below in Table 3.9. Each of the touchscreen devices are manufactured by Raspberry Pi and are official products.

Device (Size)	Supplier	Price
10 inch	Newegg.com	$209.99
7 inch	Digikey.com	$79.00
5 inch	Wish.com	$30.00
4 inch	Wish.com	$15.00

Table 3.9: Touchscreen Interface Comparison

It is clear to see that as inches are shaved off, the price is significantly lowered. The four- and five-inch touchscreen interfaces are on sale at the time of this table being created, which explains why they are such lower cost than that of the seven-inch. The seven-inch screen seems to be the best choice for a comfortable yet affordable touchscreen input device. We have not yet implemented the touchscreen interface during prototyping but plan to add it to the network as we construct the final network. Ideally, users will be able to draw a character within the English alphabet with a stylus, and the neural network will be able to output which letter has been drawn.

3.4.15 Additional Component Considerations

While most attention is rightfully directed towards the selection and vetting of components such as the operational amplifiers and digital potentiometers used in the network, the level of precision required for this device does not allow for broad assumptions of ideal behavior and minimal error. While the network's training algorithm is extremely forgiving with many sources of error, it may take unreasonably long or fail to converge with certain

training sets if error is too high. One of the largest sources of potential error is the inverting/non-inverting buffer pair for each input into each layer of the network. Assuming a worst-case scenario with 5% resistors, it is possible for the inverting buffer to distort the input signal by up to 10%, or 165 mV on the maximum signal of 1.65 V. By comparison, a 256-tap potentiometer can adjust a 1.65 V signal – or 3.3 V from positive to negative – by around 13 mV, or .4%. Thus, the maximum error even with 5% tolerance resistors is equivalent to around 25 potentiometer taps, or 10% of the entire range of the potentiometer. In a design with multiple layers, this error may accumulate to the point that the network may no longer be able to converge once other sources of error are included.

CMOS hex buffers are one component that could be considered for including in the input layer circuit of the final neural network design's synapses. Previously, in the intermediate network design, analog unity-gain operational amplifier buffers were used to reduce the loading effect of the input voltages to the layers of the neural network. Complementary CMOS hex buffers could be used in the input layer of the final neural network to provide the PB0 and PA0 pins of the digital potentiometers with the required voltage ranges for synapse weight adjustment in training. Since the SIPO shift register that will provide the logic-level input data voltages to the CMON inverters and buffers, the same logic level voltage could be applied to the corresponding pin of the digital potentiometer. Thus, it isn't absolutely necessary to include the CMOS hex buffer component, since the loading effect wouldn't be as great when using these digital components in the input layer. However, if CMOS line drivers with buffer/inverter pairs in the same IC are made available, they could prove a more efficient use of PCB space and could be considered for including in the input layer's circuit design.

3.5 Small-Scale Network Design

An intermediate network design is a must when considering a final design of the magnitude at hand. Being able to design a smaller network that performs the same end goal will be a tremendous help in designing the final network since we are taking a modular approach to its construction. The idea in mind is that the smaller scale network will be able to seamlessly become the larger network after some slight modifications to input size and a couple other of factors. The goal is to have a smaller scaled network that can easily transform in to the final neural network by connecting neuron circuits together in such a way to realize a larger network. Since we are taking a modular approach to the construction of our network, this should not be too difficult. What may cause difficulty is obtaining the same results that our smaller scaled network is obtaining while scaling up to an exponential increase in components.

3.5.1 Small-Scale Network Design and Functionality

Before the end goal of a neural network that can recognize a hand drawn character in a 5x5 pixel array can be accomplished, an intermediate step needs to be taken. The approach taken in this project is to construct a simple neural network that can recognize a hand-drawn pattern in a 2x2 pixel array. These patterns will consist of columns, rows, diagonal lines, single pixels, and three- or four-pixel patterns. These cases cover every possible combination of pixel inputs for a 2x2 network. These network dimensions were chosen because it is scaled down enough to make it feasible to build the constituent subsections on breadboards. As such, this intermediate network provides the opportunity to take

necessary testing measurements to ensure that the 5x5 network will operate smoothly before designing and order a printed circuit board to officially train and test the end goal functionality of the final network. At the same time, it is also large enough to introduce some problems that may become unmanageable when scaling the network up to the final 5x5 network, such as current and voltage fan-out, power distribution management problems, digital communication errors, and unwanted signal noise. Any inconsistencies between the intermediate and final network may be potential problems, so any significant problems must be addressed before scaling up to a larger network. Minimizing any inconsistencies is key to have the most seamless process possible when increasing to the complexity of the final network.

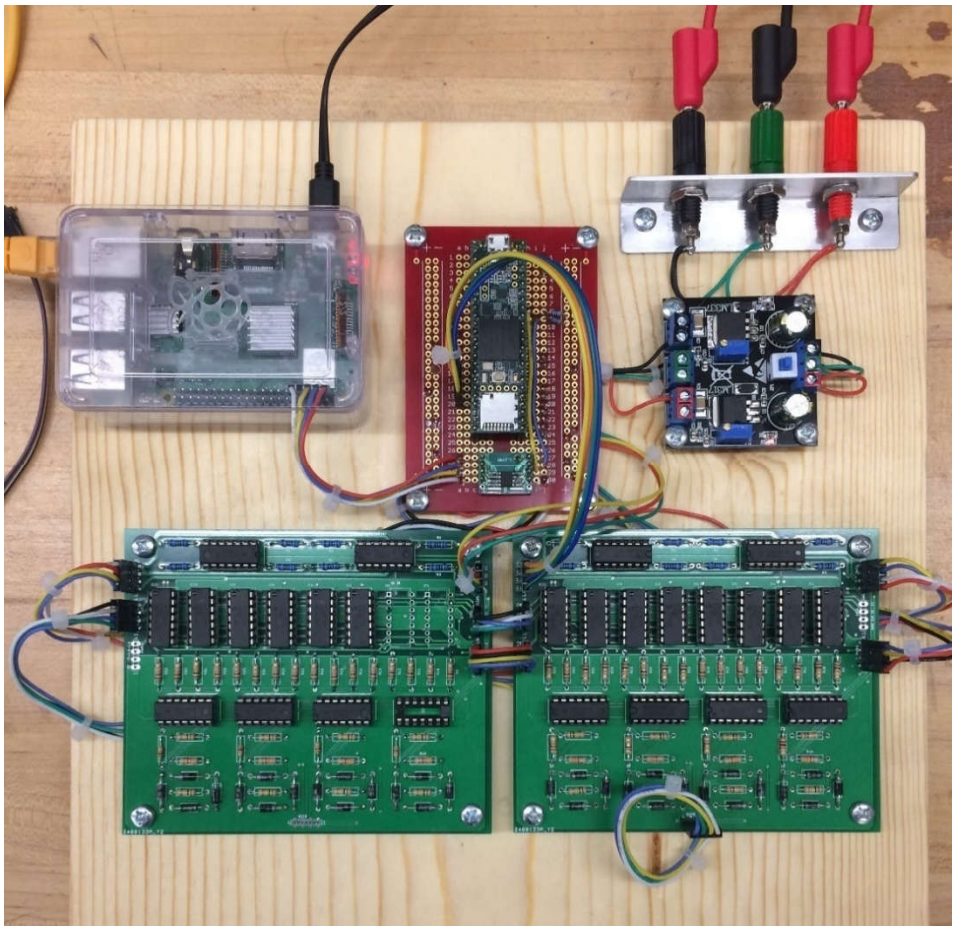


Figure 3.12: Small-scale prototype assembly.

The design is effectively a scaled down version of what the final network will look like. Since a modular approach is being taken to constructing the total network, this should not be too difficult. Each printed circuit board ordered will serve as one complete neuron circuit, and they will be configured together as necessary to realize the network. The full smaller scale prototype design can be seen mounted and configured in Figure 3.12. Depicted are two synapse-neuron circuit boards located in the bottom center, the Teensy 3.5 development board located in the upper middle, power supply input jacks that feed in to the precision voltage regulator in the top right, and the Raspberry Pi, which feeds inputs to the Teensy 3.5 development board, at top left.

3.5.2 Necessary Tests and Measurements

There are many necessary tests and measurements that need to be conducted before an artificial neural network is brought to life. Every parameter that can be reasonably controlled must be intentionally chosen and measured to a tee. Some required measurements include the current and voltage at each node throughout the circuit to allow for troubleshooting when errors or unanticipated behavior occur. Testing input versus output transfer functions and voltage levels is also crucial when managing a large network. For example, if there is a voltage drop between the input and output of the summing amplifier stage, that is a red flag that the neuron is not functioning properly. Precise control over the voltage and current levels in every portion of the device is crucial, as the network is sensitive to millivolts of noise. While static error can be trained around, dynamic, random, or intermittent noise issues will cause severe problems for the network and must be located and eliminated wherever possible. Additionally, since the network's training relies solely on the output voltage produced by each neuron, fast and accurate measurements must be repeatedly made with the ADC; inconsistent measurements will complicate training and may result in an erroneous solution for the network. Results and measurements obtained will be properly recorded and documented to ensure proper functionality as we continue to prototype.

3.5.3 Small Scale Network Training

Since producing a functional network on breadboards proved extremely difficult, the design was realized on PCBs. Since this allows for much shorter and stronger connections, this eliminates the signal loss issues common to the breadboard implementation and allows a single controller to control the entire network serially and make the necessary ADC readings for training. Since the SPICE simulation training and physical network training function in an extremely similar manner, it is not a particularly complicated endeavor to convert the SPICE training program into a suitable program for training the physical network, since the dimensions and overall training process are identical. The key difference is that the controller no longer needs to wait for each individual SPICE simulation to finish; the speed of the physical network is nearly instantaneous by comparison. Thus, within reason, a much higher training speed can be achieved. Even if the network struggles to converge and requires far more training than the simulations to reach a solution, the superior speed of the physical network more than offsets any difficulties.

3.5.4 Problems and Concerns for the Final Network

There are several concerns for the functionality of the final network to be constructed. Maintaining high accuracy results with twenty-five or more inputs is the goal to be accomplished. There are several different variables that can affect our results accuracy and overall functionality of the network. One of these variables would be the fan-out of the system. Fan-out is a term that describes the maximum number of inputs we can feed for one output. If our current as we pass from individual neuron circuits with a modular approach diminishes at too quickly of a rate, this will be a problem. When the larger network is being tested, measurements will be taken to ensure that our current is stabilized throughout the entire network. This is one of the most crucial concerns for our network from an analog hardware perspective. Another concern would be voltage dropping across long traces on the printed circuit boards. This can be a voltage drop up to 15mV, which is

the difference between several digital values on the wipers of the digital potentiometers in place. These values are highly important since they determine the weight that needs to be applied to each synapse. If the weight that's being applied to one synapse is not what it is intended to be, the results can lose tremendous amounts of accuracy. The printed circuit board needs to be designed with intentions to avoid long traces and without a clunky spread of the components. Components should be placed logically to have short traces connect power to each integrated circuit being used in the design. The integrated circuits that draw power are the TL084 operational amplifiers and the MCP42010-E/P digital potentiometers. This will be solved by cutting one of the PCB layers in to four different rails. Plus and minus 5 V DC as well as plus and minus 1.66 V DC. This will allow for logical placement of each integrated circuit to avoid long traces which can cause significant voltage drop.

Besides hardware limitations, another concern is proper implementation of training the network. While simulations performed to date have indicated the network will function properly, but there are several things that could go wrong. One constant concern is that, because the voltage losses across the network's traces and pathways will be non-negligible, the training algorithm may not be able handle signal aliasing beyond a certain threshold. Because the network relies on fairly precise analog voltage levels, relatively small inconsistencies in the network's voltage levels can produce a dramatically changed network output if the training algorithm does not respond properly. While the network is fairly flexible and is capable of training around these deficiencies when they are relatively small and consistent, blindly relying on training to deal with poor design is not an acceptable solution, especially when other sources of error must be factored in. Intuitively, there is a cumulative limit to the amount of error that the network training can handle before there is no longer a convergent solution to the training data.

3.6 Top-Level Design

A top-level design for the intermediate neural network needs to be made for testing and prototyping. The small-scale network's PCB layout and the circuit schematic of the artificial neurons and synaptic weights are included in the chapters below. This section of the document covers essential design steps for intermediate prototyping that will be conducted.

3.6.1 Singular Synapse-Neuron Circuit

Starting from scratch, our first goal was to build a singular completed synapse-neuron circuit, which consists of an "A block" (buffer stage), "4 B blocks" (digital potentiometers) and a "C block" (neuron output stage) from our block diagram. This circuit will commonly be referred to as a single neuron throughout the document. The completed circuit of one neuron consists of a unity gain buffer amplifier that will feed in to four different digital potentiometers. This part of the circuit is what we refer to as the synapse stage. The output of each digital potentiometer will then feed into an inverting summing amplifier that will sum the four voltages acquired from the output of the digital potentiometers. The input voltages will vary based on the wiper position of the digital potentiometers, which will change weights based on the input and desired response of the fully trained system. The

output of the summing amplifier stage will feed into the activation stage, which is another amplifier circuit that gives us a hyperbolic tangent response transfer function.

After finding encouraging results from both the hardware network simulations and SPICE simulations of the device, the decision was made to proceed with a hyperbolic tangent function response, which is amongst a number of commonly used non-linear activation functions for neural networks. With this summary of the design process and major components and subsections of the singular synapse-neuron circuit complete, the actual schematics and board design of the network can be discussed. The activation function also closely resembles the hyperbolic tangent function which is another commonly used non-linear activation function for neural networks. Using this hyperbolic tangent function may also be viable and will be looked in to as the network construction progresses.

First, a schematic of an individual synapse-neuron circuit has been provided below in Figure 3.9. In this case, the neuron has been designed with a total of 4 inputs, suitable for the 4-pixel test network used in this project.

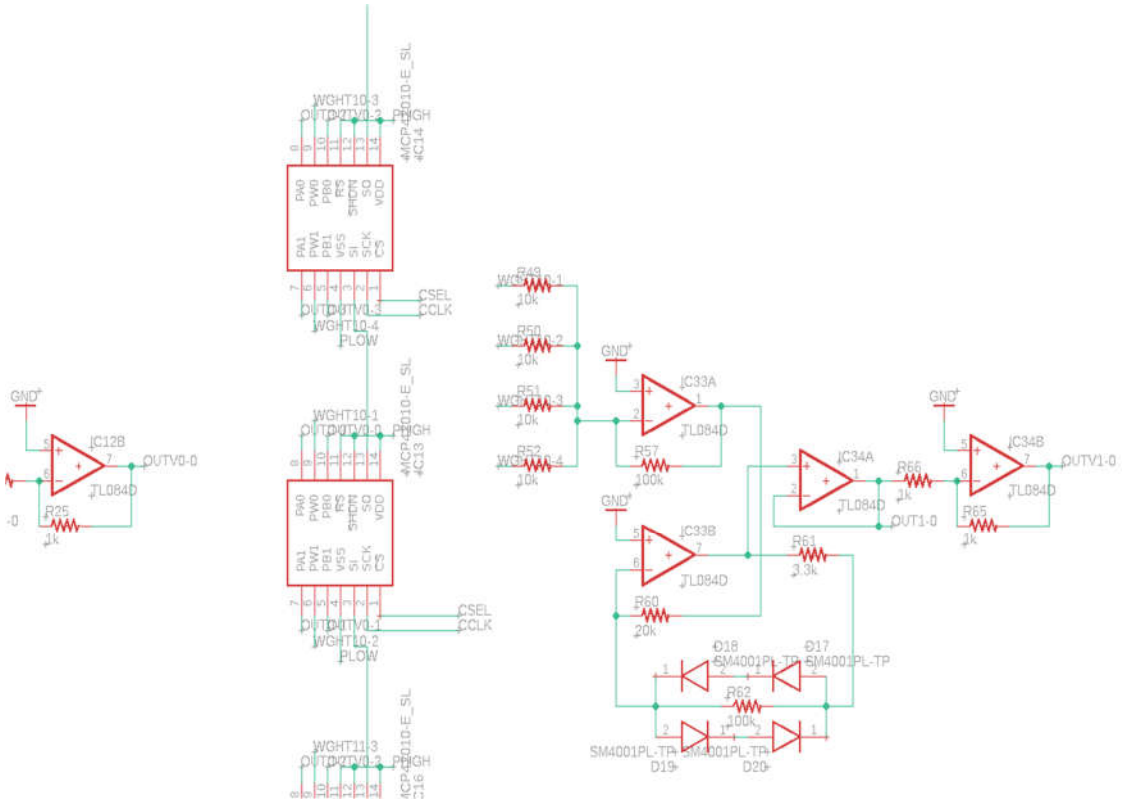


Figure 3.9: Individual Synapse-Neuron Circuit

3.6.2 Smaller Scale Implementation

Now that we can successfully construct one synapse-neuron circuit, it is time to combine several of these circuits to create our 2x2 four-pixel recognition neural network. This will be done by essentially creating two separate layers. One layer will consist of four synapse-neuron circuits and the other will consist of three synapse-neuron circuits. This will be referred to as the 4x3 layout. For the initial four pixel network used to test the components

and theory of the project, refer to Figure 3.13 for a schematic created in Eagle PCB Design Software.

While a 4-pixel network only allows for the classification of extremely simple shapes and figures, such as lines, dots, and Ls, the number of neurons and synapses in the network is still sufficient to produce valid feedback on the behavior of the individual components and the network as a whole. If it were practical to use a smaller network, this most likely would have been done; however, the network dimensions cannot reasonably be reduced below 4 inputs without effectively turning it into a simple switchboard for 1 or 2 pixels, at which point the design could simply be implemented with a few logic gates.

Since the PCBs are designed in a modular fashion, the number of layers for this network can easily be varied. The default layout being targeted is a 4x3 layout to observe the behavior of a multi-layer network without excess cost; layers can easily be added or removed to the design to observe the changes in network behavior, since the process is as simple as connecting or disconnecting one of the PCBs. This provides tons of flexibility when considering the potential size of our final network.

Figure 3.13 shows the layout of our four to three-layer intermediate network. You can see the four initial input buffers in the top left. These input buffers consist of two unity gain

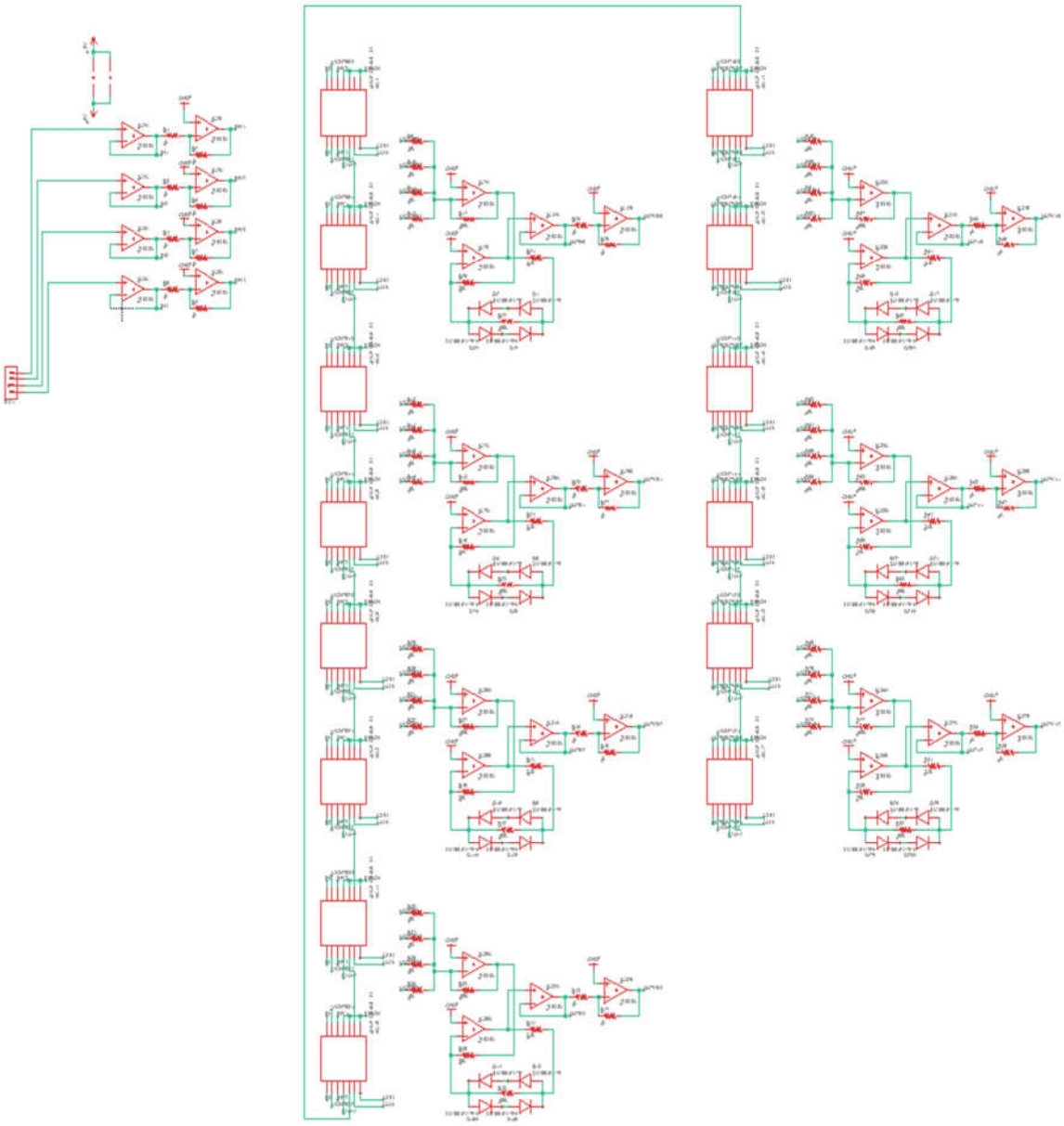


Figure 3.13: Four to Three Layer Neural Network

buffer amplifiers, one inverting buffer amplifiers and one non-inverting buffer amplifier. The outputs of the inverting buffer amplifier are fed in to the PB pins of the digital potentiometers, while the outputs of the non-inverting buffer amplifiers are fed in to the PA pins of the digital potentiometers. Setting these two voltage levels lets us control the voltage output of the wiper pin by changing the wiper position of the chip using serial communication from the Teensy 3.5 Development Board. The outputs of four wipers (two digital potentiometer chips) are then fed across 10kΩ resistors and summed at the negative input of our summing amplifier stage. This can be seen in the long middle column of the schematic that consists of four individual neuron circuits. The output of our summing amplifier stage is then fed into the activation stage which consists of the operational

amplifier circuits that have a clipping diode bridge in parallel with a 100kΩ feedback resistor on the output terminal. The neuron outputs are finally fed into another set of input buffers that will regulate the current output. This makes up the four-neuron layer of our intermediate network.

The four-neuron layer feeds in to a three-neuron layer which is shown in the right side column of the schematic provided. This is done by feeding the new input buffers into the respective PA and PB pins of the following layers digital potentiometers. After this connection is made, the same process described above is completed again, but with only three synapse-neuron circuits instead of four. After this circuit was assembled and desired output responses of each neuron were obtained, we just needed to train the circuit using software so that it can adjust the weights of each digital potentiometer's wiper output based on the given pixel inputs to give an accurate response of what was inputted. Now that we have a proper schematic resembling our breadboard prototyping, we will design and order printed circuit boards, solder components and continue to test the network. Being able to test with PCBs opposed to breadboard prototyping allows us to bypass tedious troubleshooting with the breadboard. We can now swap out components quickly to ensure consistency between circuit boards and functionality of the overall network.

3.6.3 Small-Scale PCB Design

The creation of the initial 4-pixel test network was the first foray into PCB design for this project. While breadboarding was practical when dealing with individual neurons and very small partial networks, it became unmanageable when scaled up to the size of even a small network. Because a PCB is unavoidable for the final network, the 4-pixel test network was an ideal opportunity to begin producing PCB layouts as practice for the eventual production of the final 25-pixel network's layout.

In accordance with the discussions and guidelines laid out in Section 6.5 and throughout Chapter 8, a board layout with 4 inputs and 4 outputs was selected. This allowed for an arbitrary number of layers to be added or removed from the network for testing purposes and allowed for the implementation of an "unclassifiable" output neuron in addition to the "diagonal", "horizontal", and "vertical" output neurons used to classify line directions in the 4-pixel input. As shown in Figure 3.14, the board features a compact design with two sets of pins and streamlined analog signal pathways.

Since this design is intended to be a transitional step between breadboard testing and a full-sized character recognition network, the decision was made to utilize through-hole parts for this step, though surface mount components will be used for the final network design. While this increases the footprint of the device by a factor of around three, it allows for easier component placement and handling and permits the re-use of parts from earlier prototyping stages. Since the purpose of these boards is primarily to eliminate the noise and interference issues that plagued breadboard prototyping, the ability to adjust, replace, or remove various components is key to this design. Through the use of DIP sockets, integrated chips with identical pinouts can readily be swapped out.

Because component swapping is so easy, it is extremely useful to observe the behavior and function of various components in a network application. While it is possible to test components by themselves or in a simple individual neuron, the true test of a device's

behavior is in a network application. For example, while an amplifier may appear to be fine when tested in a single neuron, propagation errors may occur when multiple neurons are run in series and parallel, as in a standard network. The ability to swap components out of an already-functioning network allows for rapid component testing and eliminates the need for guesswork or trial-and-error testing of components. This is especially key when dealing with fairly expensive parts, such as potentiometers; purchasing a hundred devices for a full-size network only to discover they do not function in a large network application is an unacceptable risk and an unnecessary budget violation.

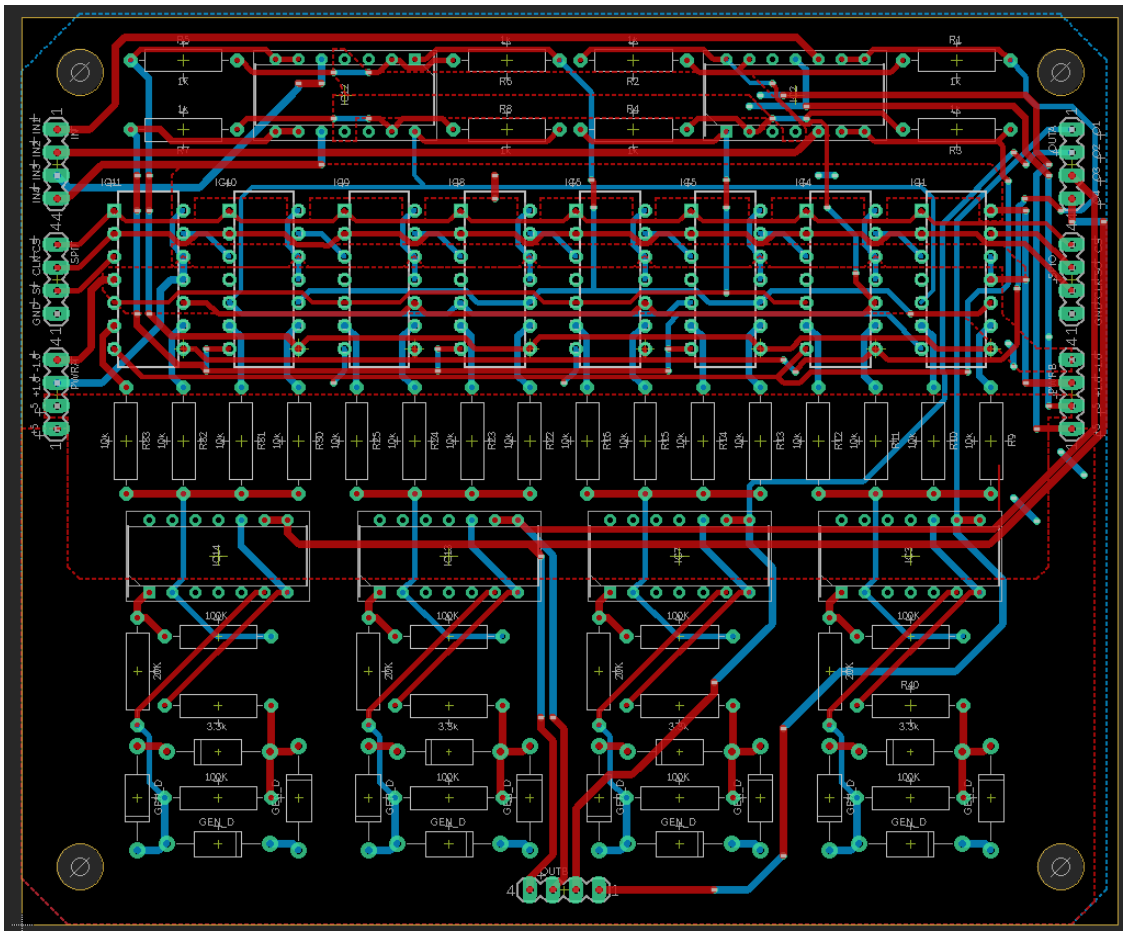


Figure 3.14: 4-pixel network layer PCB with voltage planes omitted

The EAGLE PCB layout we are utilizing for this initial test network is shown in Figure 3.14, with voltage planes (bordered by dashed red and blue lines) left unfilled for clarity. Voltage planes are used for the voltage rails of the circuit where possible, and trace widths are maximized within reason to minimize signal attenuation. Small voltage drops over long traces are another concern that will only show in practical application, and must be accounted for. Minimization of these trace lengths has been implemented in the design to avoid any significant voltage drops that could disrupt obtaining clear results. The four inputs to the circuit are provided via the top left row of four pins, and the four output signals can be read from either the top right row or bottom center row of pins. The remaining pins

are used to carry the voltage rails and SPI signals. In the third and final layer, the rightmost two potentiometers and rightmost neuron are omitted to produce a 4-input, 3-output layer. This design is subject to change as more tests will be completed to determine the designs functionality.

4 Related Standards and Real-World Design Constraints

This chapter describes how implementation details of the TACOCAT project are affected by two domains of real-world design rules: Related Standards and Real-World Design Constraints. All major projects, especially those with intent to be distributed,

4.1 Related Standards

This section describes technology standards that are relevant to the TACOCAT project. Since this device relies heavily on software-based theory and on digital communication, most of the applicable standards are related to software and firmware.

4.1.1 Serial Peripheral Interface (SPI)

Serial Peripheral Interface (SPI) is a widely used communication protocol in embedded systems. It was first developed by Motorola in the 1980s, and today it is a widely accepted standard that is not enforced by any major institution. An SPI communication network consists of a master device, that is the generator of a serial clock and slave-select signals, and at one or more slave devices that share these serial clock and data buses.

For TACOCAT, the microcontroller chosen for adjusting the weights of the synapses of the neural networks being trained, is the master device responsible for regulating the timing and transfer of serial data using this communication protocol. All the peripheral devices, like the digital potentiometers, serial shift registers, and hex inverters would be the slaves.

4.1.1.1 *Impact of SPI on Design*

The effect of SPI on this project is significant. While SPI, I2C, and other hardware communication protocols generally rely on parallel devices and device addressing, a special case of SPI does away with this characteristic. In certain devices, device RAM is implemented using a shift register along the SPI data pathway instead of a standard register. While the difference may seem subtle, this allows data to be moved serially from one device to another along the same pathway. Since each device only supplies the device in front of it, the length of an SPI daisy-chain is theoretically arbitrary, and only practically constrained by clock and chip select attenuation.

The ability to create arbitrarily long serial communication paths is the linchpin to the function of this project. Since network sizes vary by orders of magnitude, the number of digital devices can vary from less than a dozen to several hundred. Parallel signaling to hundreds of digital devices is an unreasonably difficult task, as external multiplexing hardware would be required to manage so many devices. Even with multiplexing, a significant number of additional controller pins would be required to operate the multiplexer. In contrast, the SPI daisy-chain approach requires a total of three data lines – clock, chip select, and data – regardless of the length of the chain. As a result, standardized control software can easily be created, and the number of pins required is constant regardless of network dimensions. Additionally, since there is no longer a need for dozens or hundreds of chip-select lines in parallel, routing becomes dramatically easier and allows for much more efficient component placement.

4.1.2 I2C-bus Communication

The Inter-Integrated-Circuit (I2C or I^2C) communication bus is a two-wire serial communication protocol that was developed by Philips Semiconductors (now NXP Semiconductors) and originally released in 1982 (NXP document UM10204). The protocol became a de-facto standard in the embedded systems industry, and the first formal specification document for the protocol and related hardware was published in 1992.

The standard is currently described in the 6th revision of the formal specification document that is published by NXP Semiconductors as document UM10204, "I^2C-bus specification and user manual." Among other topics, the specification document describes the standard specifications for the following aspects of I2C:

- Master/slave device organization for I2C busses, including multi-master systems
- Clock and data signal timing
- Communication procedures using affirmative and negative acknowledgements
- Device-address allocation and format
- Communication control mechanisms, including clock-stretching and multi-master arbitration

The standard specifications also include specific guidelines for maximum data transmission rates depending on the mode of operation for an I2C bus, which are shown in Table 4.1.

Mode	**Maximum Data Rate**
Standard	100 kbit/s
Fast	400 kbit/s
Fast+	1 Mbit/s
High-speed	3.4 Mbit/s
Ultra-fast	5 Mbit/s (unidirectional only)

Table 4.1: Maximum data rate by I2C operation mode

Previous I2C documents specified that I2C-bus technology was protected under patents held by Philips Semiconductors and that any manufacturers wishing to implement I2C protocols on an IC device were required to obtain a license from Philips (Philips document AN10216). However, this is no longer mentioned in the current I2C standard specifications.

4.1.3 Federal Regulations for Radio Frequency Devices

Standards for radio-frequency devices are specified in Title 47, Chapter I, Subchapter A, Part 15 of the U.S. Federal Code. While the TACOCAT project does not involve any intentional emission of radio-frequency signals, any device manufactured using TACOCAT technology would still need to conform to the standards for unintentional radiators of radio-frequency signals that are described in this part of the Federal Code, which is administered by the Federal Communications Commission. Title 47, Chapter I, Subchapter A of the Federal Code may also be referred to as the "FCC Rules."

4.1.3.1 *Device Classification*

According to section 15.3 of the code, TACOCAT technology would most likely be evaluated as a Class B digital device: "A digital device that is marketed for use in a residential environment notwithstanding use in commercial, business and industrial environments. Examples of such devices include, but are not limited to, personal computers, calculators, and similar electronic devices that are marketed for use by the general public." Depending on the application, TACOCAT technology might also be evaluated under the criteria for a Class A digital device: "A digital device that is marketed for use in a commercial, industrial or business environment, exclusive of a device which is marketed for use by the general public or is intended to be used in the home."

Additionally, TACOCAT would be considered an "unintentional radiator" of radio-frequency signals, which is described in 15.3(z) as a device that intentionally generates radio-frequency energy for its own internal purposes but does not intentionally broadcast radio-frequency signals. Radio-frequency signals seem to be described in Section 15.3(k) as "signals or pulses at a rate in excess of 9,000 pulses (cycles) per second."

4.1.3.2 *Information Provided to the User*

Section 15.105 of the code specifies that the device must include a statement indicating compliance with part 15 of the FCC Rules in the device's user manual. Boilerplate text is provided for both Class A and Class B digital devices. For Class A devices, the suggested statement is:

> Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

4.1.3.3 *Limits for RF Signals Conducted into AC Power Lines*

Section 15.107 of the code describes voltage limits for RF signal that is unintentionally conducted back into the AC power lines that Class A or B digital devices would be connected to. Different relative voltage levels are listed (in dBμV) depending on the frequency of the RF signal.

Table 4.2 shows the limits that are given for Class A and Class B digital devices in Section 15.107 of the FCC Rules. Limit ranges that are marked with an asterisk (*) reflect that the prescribed relative voltage levels decrease with the logarithm of the frequency.

While power supply design is not within the scope of the TACOCAT prototype development process that is described in this document, it is reasonable to assume that as the technology matures and is implemented commercially, these limits would need to be evaluated.

In its current design iteration, TACOCAT's MCU clock frequency of 120 MHz is well above the maximum frequency of 30 MHz that this section concerns. However, with an SPI communication network operating in the 4.0 – 5.7 MHz range and an I2C bus designed

to operate at 0.40 MHz, the prototype does have some systems that could present problems with RF signal leakage into the AC supply lines.

Digital Device Type	Frequency of emission (MHz)	Conducted limit (dBµV)	
		Quasi-peak	Average
Class A	0.15 - 0.50	79	66
	0.5 - 30.0	73	60
	0.15 - 0.50	79	66
Class B	0.15 - 0.50	66 to 56*	56 to 46*
	0.5 – 5.0	56	46
	5.0 – 30.0	60	50

Table 4.2: Limits for RF signal conducted into AC power lines. Adapted from tables provided in FCC Rules Section 15.107.

It is also specified in this section that devices operating solely on battery power are exempt from these measurements. It is unclear if devices that operate primarily on rechargeable batteries but can also be powered by a charging device (such as mobile phones or tablets) would also be exempt from these regulations.

4.1.3.4 *Intentional RF Radiation in Peripheral Devices*

The TACOCAT prototype is likely to incidentally include some peripheral communication devices that are designed to transmit and receive RF signals. A primary example would be that most of the mini-PC or mobile computing devices that would be candidates for user interface implementation will include wireless communication modules for wireless internet or Bluetooth protocols (typically operating at 2.4 GHz or 5 GHz).

In order to provide a reasonable level of assurance that these communication devices would comply with the specifications found in the FCC Rules, any mini-PC or mobile computing devices used for the TACOCAT user interface should include documentation stating their compliance with the FCC Rules. For commercial production, independent testing would still be required for the completed TACOCAT device, but the chance of peripheral components failing to comply with federal regulations would hopefully be minimized.

4.1.4 U.S. Food and Drug Administration: Proposed Regulatory Framework for Modifications to Artificial Intelligence/Machine Learning-Based Software as a Medical Device

As artificial intelligence and machine learning technologies have only recently made significant entries into the marketplace of safety-critical devices, there are few examples of formal standards regarding the safety of AI/ML-based devices. In a discussion paper released in 2019, titled *Proposed Regulatory Framework for Modifications to Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD) - Discussion Paper and Request for Feedback*, the U.S. Food and Drug Administration (FDA) has described a hypothetical system for evaluating the safety of proposed updates

to Artificial Intelligence/Machine Learning (AI/ML)-based medical devices, including software-based "devices" such as mobile computing device apps.

While the handwritten-character recognition function performed by the prototype TACOCAT device described in this document is unlikely to be useful in medical applications, due to the universal approximation potential of the MLP architecture, future iterations of the TACOCAT technology could conceivably be used for other recognition tasks, such as identification of skin cancer in a photograph image, real-time detection of cardiac arrest in EKG monitor signals from at-risk patients, or intelligent control of blood sugar in a wearable insulin pump device.

Some guidelines discussed for the proposed safety standards in the FDA paper are general enough that they could be used for safety standards in other safety-critical domains, such as autonomous vehicle control. It seems reasonable to expect that standardized safety guidelines and certification will soon play a very important role in the design of safety-critical AI/ML-based devices.

4.1.4.1 *Model for Best Practices in SaMD Development*

While the FDA's discussion about proposed safety standards for development of Software as a Medical Device (SaMD) focuses primarily on the development of guidelines for improvements to existing devices, it also outlines a holistic model that describes best practices for development based on the Total Product Life Cycle (TPLC) of AI/ML-based SaMDs. Parts of the model, depicted in Figure 4.1, are also general enough to be extended to other types of safety-critical AI/ML-based systems.

The FDA discussion points out that the TPLC approach is particularly important when working with artificial intelligence and machine learning technologies due to their abilities to adapt over time. Conventional software typically has a static code base that is only changed when developers release an updated version, but many machine learning algorithms are designed to adapt over time based on prior results. Regulatory bodies need to account for the possible future states of these algorithms and not just the state of the software at the time of its release.

The development model also maintains a holistic perspective by considering the development organization's "culture of quality and organizational excellence" in addition to auditing more concrete aspects of the organization such as its methods for management and maintenance of data sets, its training/tuning methods, and its validation and clinical evaluation processes.

As part of the TPLC approach, the model also accounts for the review and certification processes required for the release of new or updated technology, as well as tracking performance and evaluating changes in devices that have been released into production. Along with changes to the software algorithm, the device's effectiveness can change if it is exposed to new input data or if it is used for purposes that it was not originally intended to be used for. These changes are considered in greater detail in other sections of the FDA discussion document, especially in the context of deciding the extent to which product changes need to be regulated.

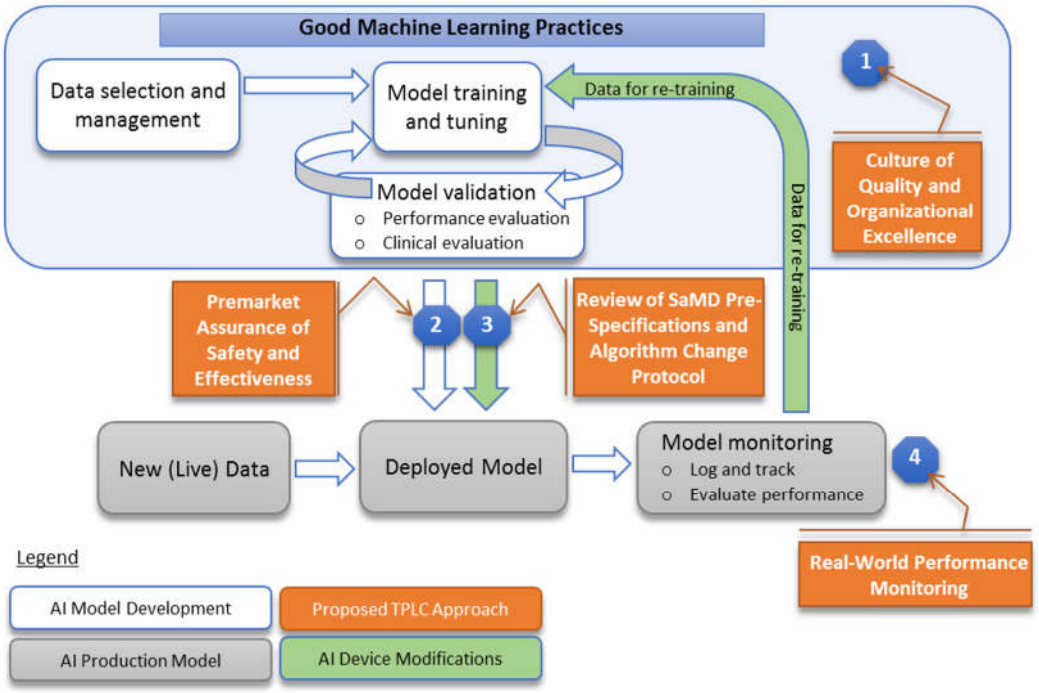


Figure 4.1: FDA's proposed model for AI/ML-based SaMD development. Reproduced with permission of U.S. Government from discussion paper at: https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-software-medical-device

4.1.4.2 *Assessment of Product Modifications*

The main purpose of the FDA discussion paper is to propose specifications for a new standard that could be used to regulate proposed changes to existing AI/ML-based SaMDs. The document discusses methods for classifying different types of proposed changes, evaluates the degree to which different changes should be scrutinized, and describes formal procedures for the approval of device modifications.

4.1.4.2.1 Classifications for Device Modification

The document proposes the classification of device modifications into three different categories:

- **Performance:** Device modifications for the purpose of increasing performance.
- **Inputs:** Modifications to the device's intended input data set.
- **Intended Use:** Modifications to the device's intended uses.

It is specifically noted that these categories are not mutually exclusive, but most proposed modifications can primarily be described using one of these classifications. The Performance classification encompasses modifications that only effect performance and do not change the way that the device is used. Modifications in the Inputs category are made

for the purpose of allowing the device to consider different kinds of inputs. These new inputs could relate to training and validation data or to operational input data. Modifications in the Intended Use category include the addition of completely new roles for an existing device (such as the addition of stroke detection to a device that is meant to detect cardiac arrest) or the degree to which devices are used in existing roles (such as converting an application that notifies doctors of possible tumor detection in a CT scan into an application that directly diagnoses tumors in CT scan images).

4.1.4.3 *Proposed Framework for FDA Approval of SaMD Modifications*

The discussion paper describes a series of "premarket certification" steps that manufacturers would be asked to complete prior to making changes to an existing AI/ML-based SaMD. At the core of the approval process is the requirement for manufacturers to submit a modification plan that gives a detailed explanation of the SaMD Pre-Specifications (SPS) and Algorithm Change Protocol (ACP). The SPS describes the proposed changes to the device in terms of the three classifications listed in the previous subsection. The ACP describes the specific methods that the manufacturer plans to use in order to change the SaMD in a safe, controlled manner. Figure 4.2 shows the decision flow for the approval process proposed in the discussion document.

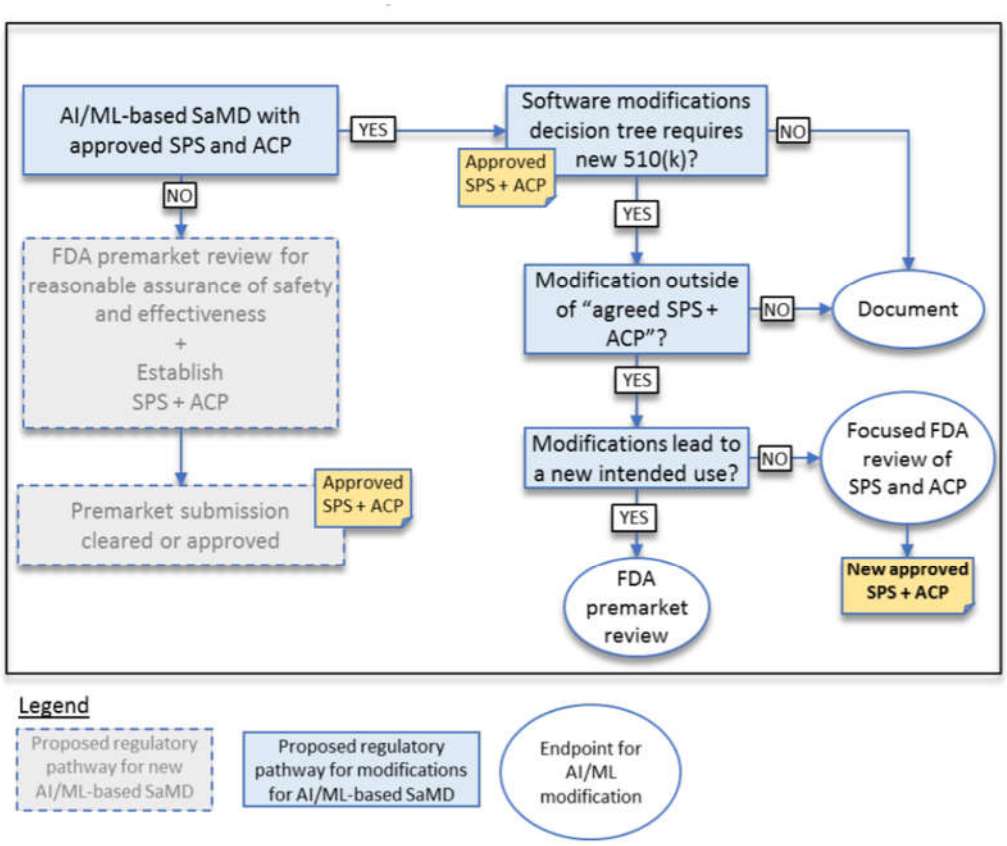


Figure 4.2: FDA's proposed approval process for SaMD modifications, to be considered in conjunction with discussion text. Reproduced with permission of U.S. Government from discussion paper at: https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-software-medical-device

The document provides detailed rationale for the evaluation of different SPS/ACP combinations, including the degree of scrutiny that required from various subgroups within the FDA and the amount of additional input/documentation that could be requested from the manufacturer.

4.1.4.4 *Relevance to TACOCAT Project*

While the TACOCAT prototype is not currently intended for any type of use as a medical device, the guidelines for safety certification standards proposed in the FDA discussion paper offer some key insights that can inform the TACOCAT design process. The most valuable insight is that the creation of effective AI/ML-based devices depends on the establishment of a development team that is rooted firmly in a "culture of quality and organizational excellence." The best way to ensure the effectiveness of the team's development methods is to build them on this solid cultural foundation.

In more practical terms, the FDA's discussion of proposed standards points out the primary objectives of the AI/ML-based development process: data selection/management, training/tuning, and validation. While any developer working in the AI/ML domain is bound to include these processes in the course of developing a new product, it is certainly helpful to have them clearly enumerated along with an explanation of how they should interact with one another during the development cycle.

Finally, while modifications to the TACOCAT prototype design are not likely to be subject to the approval of any official regulatory body concerning product safety, it is helpful to have a framework that can be used in order to consider the effects of any changes that might be made to the prototype during the course of its development. For example, while it might be tempting to modify the layout of a working prototype's input layer in order to expand the set of possible inputs, the risk of damaging a working prototype before it can be presented for evaluation by instructors could be a serious mistake for a Senior Design development group.

4.2 Real-World Design Constraints

There are constraints placed upon the team members and the overall development of TACOCAT that do not pertain to those resulting in the physical laws and objectives of the main project. Some of these constraints stem from real-world causes, which can be seen below in the economic, time, and safety constraints considered.

4.2.1 Economic Constraints

First, the economic constraints present in constructing a hardware implemented neural network using analog and digital circuitry as opposed to simulating it in software is discussed. Physical neural networks in the industry typically use nanostructures, like memristor arrays, to implement the adjustable weights of a neuron in the synapse of a network layer. These highly scalable devices, however useful for this application they may be, are incredibly expensive and range from $89 to $300 for a 1x16 discrete memristor device array. While one could argue for using these to achieve a smaller area design, the high cost of building the weights of each neuron's synapse with these devices will start to get expensive very quickly. With just the prototype board, 2 of the 16 discrete device arrays will be needed, which will cost about $600 for the prototype board. This would make

funding an issue when just getting started with the project. It is not a good idea to spend tons of money on a prototype when you don't know if it will function properly yet.

Even more concerning to the economic constraints of the project is the scale of the final implemented network. There is an exponential increase in the cost of the devices needed to realize a larger network. A 25-input neural network hardware MLP with the described design will already require a large quantity of digital potentiometers, which is the most expensive component needed to realize this design. If we were to increase the size of the network to a larger number of input-pixels to have a wider range of recognizable characters, the need for more digital potentiometers to weigh these input pixel data would drive the cost of the project up rapidly by about a factor of 5 if the project were scaled to obtain inputs for a 8x8 pixel image.

Lastly, the number of PCB's used to implement each "module" of the neural network will increase the cost of the total project. Thus far, a 4-input neuron module will be soldered per PCB, and then each of these will be interconnected using jumpers from breakout pins included on the boards. If the size of basic module used to physically realize and solder this network were increased, the cost of printing each of the PCB would increase as the total area needed would be greater.

4.2.2 Time Constraints

Time constraints that are placed on the TACOCAT project affect the overall project's specifications more perhaps than any other design constraint. All of the authors need to complete the milestones required by the ABET program in the University of Central Florida, which impose necessary timeline for completion of project milestones. This means that any added feature for this design that would otherwise largely extend the deadline for these milestones beyond the required date would cause any one of the authors to possibly delay graduation if there are any failure to meet these deadlines. This would cause not only a delay in proceeding to a career in their respective fields of study and interest, but it would also mean they would also jeopardize their academic performance in the form of their grade point average. While failure to meet these requirements is highly unlikely, the seriousness and possibility shouldn't be taken lightly and need to be kept in mind in the process of designing the scope of this project.

There are several ways that the risk of time constraints putting a halt to the overall development and completion of this project can be mitigated. First and most important of all, is proper and skilled guidance from experienced university faculty. This team of engineering students is under the mentorship of Dr. Chung Yong Chan throughout the development and execution of this project. His highly diversified and extensive technical background in guiding students through the senior design process will help prevent unnecessary mistakes and possible delays from occurring. His mentorship and guidance will help the authors set realistic milestones that will be held in high regard for completing the project in a timely manner while achieving all the required ABET and design specifications.

4.2.3 Safety Constraints

Safety is of paramount importance when handling electronic devices. While microelectronic circuits don't have the same potential dangers as projects with higher

power dissipating devices would have, safety precautions and concerns must be considered throughout the design process and while implementing the final design of TACOCAT.

Improper handling of electronic micro-chips and power supplies is a possible source of safety breach as the dissipation of heat is a possible source of bodily injury that either an observer or one of the authors could suffer if proper precautions are not taken. Thus, proper reading of data sheets of each components used must be done carefully to not bias power rails incorrectly and possibly cause the breakdown of devices to occur. Furthermore, proper handling of power supply units while prototyping TACOCAT need to be taken to ensure that no possibility of electrical shock could occur.

5 Final Network Design

The final network design is a scaled-up version of our intermediate design. The multilayer perceptron network takes one hundred inputs as opposed to four. This is taking it from a 2x2 pixel recognition network, to a 10x10 pixel recognition network. Scaling higher is a possibility but it is intelligent to work our way up. Once the four-pixel network was functional, the one-hundred-pixel network was tackled. If the one-hundred-pixel network would have been completed well before the due date of the project, a larger network could have been attempted. With this being said, a larger network could have been constructed only after proper testing has taken place to ensure its functionality. Desired simulation results are apparent, but there were many obstacles.

Simulation results prior to prototyping and design were enticing, but this didn't mean it would have been a seamless process when attempting to train the final network and get immediate success. Troubleshooting is an expected process with building any network, let alone a hardware neural network. Fan-out also was also an issue taken into consideration that needed to be tested as the network was being constructed. The final network was constructed of individual synapse-neuron circuits that consist of the stages described below. Each stage is important and was tested to ensure success. The stages make up a singular synapse-neuron circuit, and the synapse-neuron circuits are used to realize the overall network.

5.1 Synapse Circuit Design

The synapse circuit is the first circuit encountered when following the signal path in the schematic of a fully constructed individual neuron circuit. The synapse circuit handles each input to the neuron, buffering and providing complimentary input signals before supplying these signal pairs to the synapse's digital potentiometers. The potentiometers weight each of the inputs based on the values dictated by the training algorithm for the network. Weighting the inputs is achieved by using the potentiometers as voltage dividers and adjusting the wiper positions of the potentiometers, altering the voltage level at the wiper of each potentiometer. These output voltages are then fed into the summing amplifier stage.

The circuitry of the synapse stage is relatively simple. The decision was made to use simple inverting amplifier pairs to provide reliable current and voltage input into the digital potentiometers. While an inverting and non-inverting buffer pair could have been used in series to produce the inverted and non-inverted versions of the signal, inverting amplifiers require two resistors, while unity-gain non-inverting buffers require none. Even though the use of 2 resistors could be avoided by using a unity gain buffer in the synapse circuit, the use of the pair of unity gain inverting op amp circuits prevent a larger bias current into either stage, providing a more reliable voltage level for the digital potentiometer's synaptic weights.

The chosen buffer setup is depicted in Figure 5.1. Since the input is isolated from the neuron and synapse hardware, the input signals are protected against excess current draw, which may cause distortion or damage any supply hardware or previous network layers in extreme cases, as a single amplifier could otherwise potentially be responsible for supplying dozens of synapses.

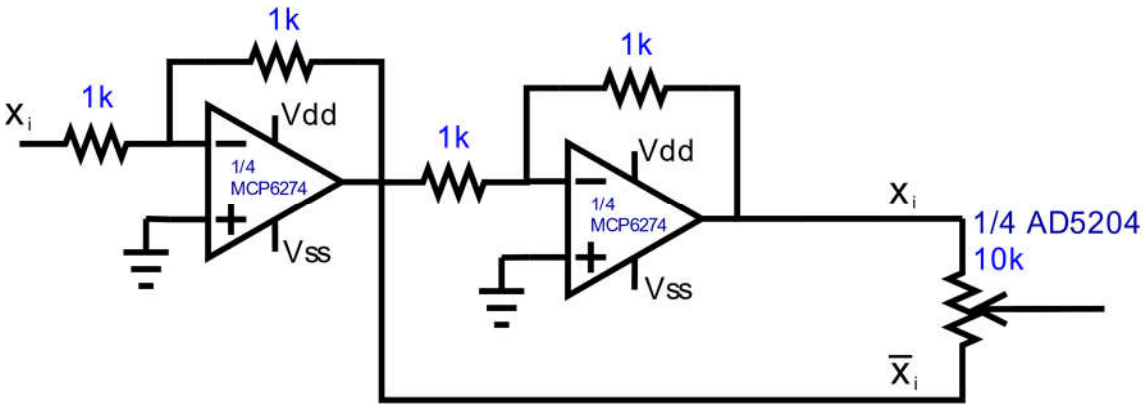


Figure 5.1: Complimentary input buffer pair schematic

The setup provided above is the least hardware-intensive implementation possible without compromising signal fidelity; a total of only six discrete components is necessary to buffer a single input. There are no reasonable alternatives to this design, as other implementations either sacrifice input safeguards or introduce redundant sources of error to the network.

Once these voltages are buffered, they are supplied to each end of a potentiometer, producing a voltage gradient across the device; the wiper of the potentiometer is then moved as desired along the device, allowing the desired voltage level to be chosen. This effectively 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.

5.2 Summing Amplifier Design

The summing amplifier circuit stage occurs immediately after the weighted input synapse circuit stage. The summing amplifier stage's purpose is to collect the outputs of the preceding digital potentiometers and sum their output voltages. ln 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. 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 foreach input to a given neuron, with n being the total number of inputs.

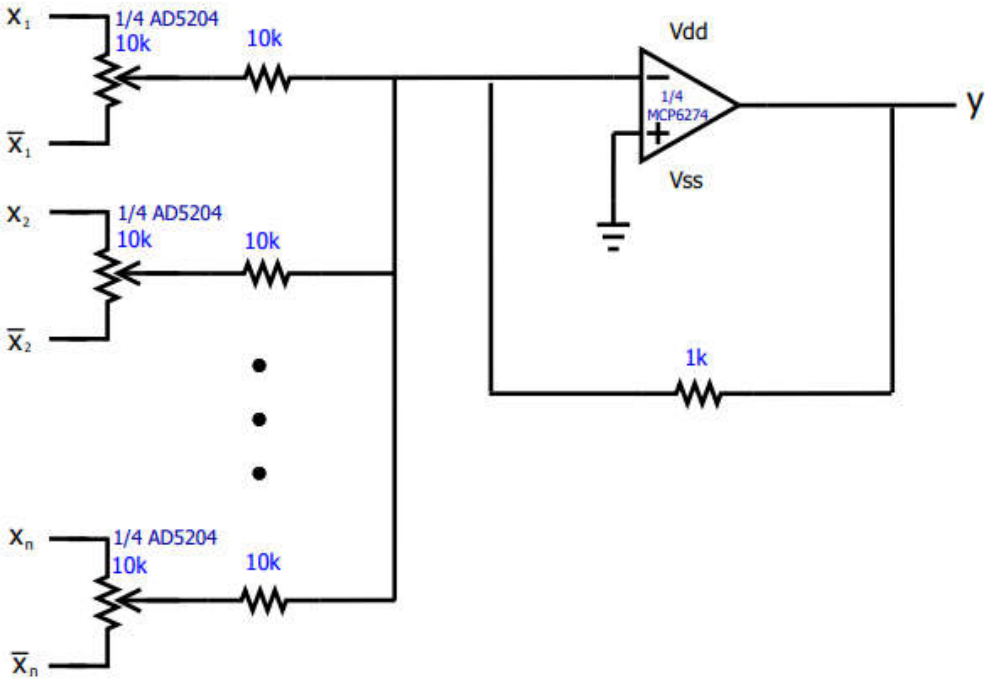


Figure 5.2.1: Intermediate Inverting Summing Amplifier Stage

The Eagle schematic shown in Figure 5.2 depicts the intermediate summing amplifier stage of a 100-input neuron in the daughterboard circuit of our final network. The 40 synapse inputs are tied together to a common voltage source, though each potentiometer is independently adjustable, allowing for different combinations to be provided to the summing amplifier. This configuration allows for thorough testing of a given neuron, as it allows the full range of possible input voltage combinations to be observed simply by adjusting each potentiometer. With all potentiometer wipers set to their maximum values, this results in a unity-gain linear output response. The linear responses of the daughterboards' 100-input adding trees can be seen in the plot of output voltage ADC readings daughterboard input activation provided in Fig. 5.2.2 below.

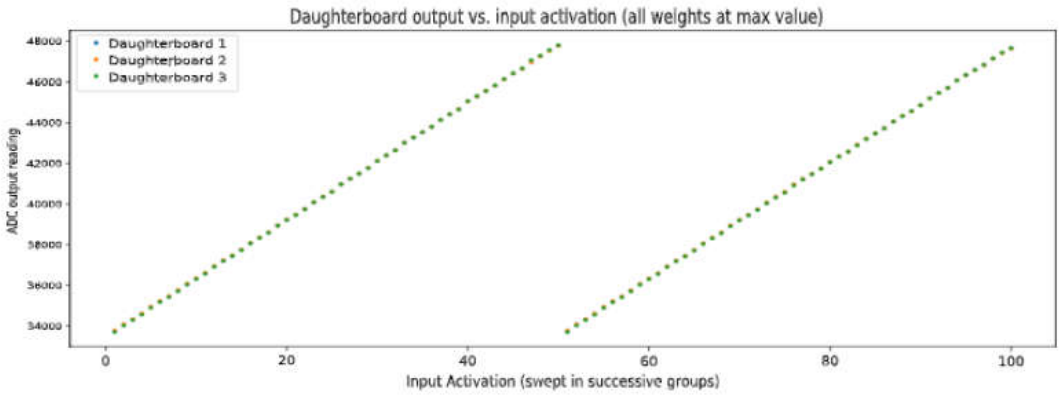


Fig 5.2.2 Daughterboard Summing Amplifier Test Result

5.3 Activation Function Circuit Design

The activation function stage is one of the most crucial portions of the neuron circuit, as it is the stage which produces the actual output function and can vary wildly depending upon the chosen configuration and component values. It is responsible for creating a transfer function between the input and output that resembles a particular function – in this case, a hyperbolic tangent function. The importance of the hyperbolic tangent function is that it is especially useful for network models that need to predict probability, which is a very common neural network application and is effectively the underlying function of a network, regardless of application.

The circuit design for the activation function is a bit more complex than the previous building blocks of the overall neuron circuit. It is, yet again, another inverting amplifier circuit, though with a few important modifications. The activation stage was built as an inverting amplifier so that the transfer function between the input of the summing amplifier stage and output of the activation function stage goes through two inverting stages, resulting in a positive output given a positive input.

In this case, the positive terminal of the operational amplifier is connected to ground, as usual, and the input to the stage into the negative terminal is provided from the output of the synapse circuit stage and is fed across a 10 kΩ input resistor. This resistor value was chosen to minimize the input current to the activation stage to a current value well below 1 mA while avoiding distortion caused by the input bias and offset currents of the amplifier.

As in a standard inverting amplifier, the output of the activation stage is fed back to the negative input terminal; however, this is where the similarities end. This feedback is provided across both a diode-clipped resistor and an independent resistor in series with the voltage clipper. The independent series resistor is a 51 kΩ resistor, while the voltage clipper resistor is a 220 kΩ resistor in parallel with two sets of diodes, one in each direction, as shown in Figure 5.3.

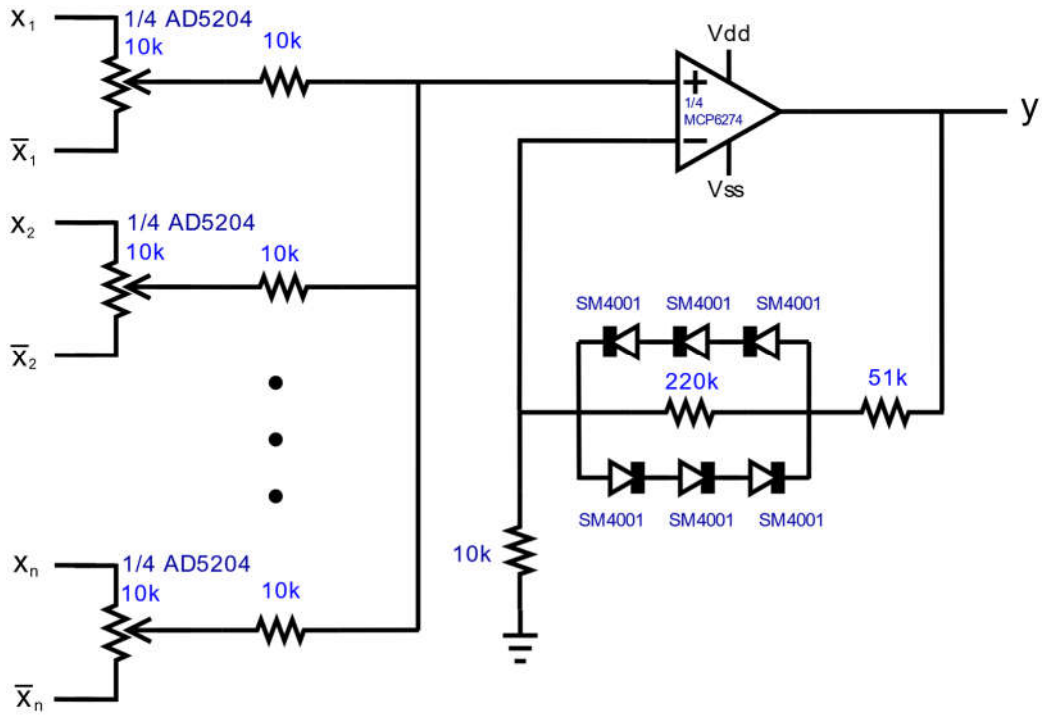


Figure 5.3: Activation Amplifier Schematic

Each series diode configuration consists of three diodes in series that are facing the same way, with the polarity of each branch opposing other. This is so that one pair of diodes is set to clip positive voltage and the other pair of diodes is set to clip negative voltage. Once the output of the activation stage reaches a certain point, the voltage across the 220 kΩ resistor can no longer increase, resulting in a stable current draw through the feedback of the amplifier. Given this constant current, the maximum output amplitude can then be chosen by selecting the corresponding series resistor value.

5.4 Neuron-Output Voltage Reading

Finally, the output signal provided by the activation stage is fed into another unity-gain non-inverting buffer to avoid output distortion. While this is redundant for most hidden layers due to the input buffers on each successive stage, it is necessary to buffer the outputs of the last layer of any network, as additional current draw induced when performing ADD measurements or supplying LED or LCD indicators may result in erroneously low-magnitude output voltages.

Depending on the current draw of the output setup, whether it be effectively zero, as with comparators as described in Section 5.5, or fairly high, as in a fairly complicated LED network, output buffering may not be necessary. Since the activation function supplies its output directly from the output terminal of the neuron, it is possible to draw current from the activation function without affecting its output voltage. However, amplifiers cannot be presumed to maintain a perfectly constant output voltage regardless of current draw, so attention must be paid to the characteristics of the amplifier used in the network.

While elimination of the output buffer would reduce the amplifier count to two, allowing two neurons to be fit into a single 4-amplifier chip like the TL084, other concerns arise. Predominantly, if high current is being drawn from multiple amplifiers on a single chip, the power dissipation may become unreasonably high in the device, especially if the supply voltages are significantly higher than the output voltages of the amplifiers. Additionally, some amplifier chips are prone to crosstalk; while this is not typically a significant factor in most chips, certain amplifiers under particular conditions can generate non-negligible amounts of interference among themselves within a single chip, resulting in mutual output distortion.

5.5 Output ADC Readings

Standard software-based neural networks often deal with extremely high value precision, calculating to multiple decimal places when processing the values at each hidden layer and output node. In many networks, where each output is a different value – for instance, predicted temperature, humidity, wind speed, and so forth – each output value is free to be as precise as it can be, with no meaningful effect on the other values. However, when dealing with network applications like handwriting interpretation, there is a limit to the acceptability of this behavior. For instance, if the network is struggling to differentiate between “F” and “P”, it is not an acceptable response for the network to simply output a value of .5 for F and .5 for P, or something near these values. A user likely expects the network to either make a definitive guess or to indicate it does not have confidence in its response. While this is easy to solve in a software-based network in a few lines of code, such as simply selecting the highest output value and flagging it for low confidence if it is below a given threshold, it is somewhat more complicated to handle when dealing with physical circuitry.

There are a few solutions to this problem which vary in suitability depending upon the context of their use and the network they are applied to. The first approach is to simply use a positive and negative reference voltage via a divider and supply these voltages to a pair of comparators; by attaching LEDs to the outputs of each of these comparators, one will be able to determine if the network is confident that it is or is not a given character. This solution is somewhat arbitrary, which can be viewed as a positive or a negative. Because the voltage reference level can be changed at will, the desired confidence level of the circuit can be tweaked. This can, of course, be a strong positive when high confidence is required or expected; however, in cases where networks produce weak but accurate calculations, one may never see an output despite the network consistently producing correct outputs.

Another solution to this problem is to add another output to the desired network. This neuron can be used as an “unclassifiable” output, which should be indicated whenever the network is unable to match another output to an acceptable degree of confidence. For instance, in the 4-pixel test network, it would be unreasonable to expect the network to guess “diagonal”, “horizontal”, or “vertical” if it were given a blank or completely filled in sample; doing so may result in a higher rate of misclassification, especially if the network is given grayscale inputs instead of binary black-or-white inputs. Using a fourth neuron, an “unclassifiable” output can be used as the default state of the network. In an ideally trained network, this output should always be enabled unless the network is highly confident in its classification of a signal using the other discrete outputs. This allows new, ambiguous, or

invalid inputs to the network to safely be disposed of by default instead of forcing the network to attempt a solution. This is especially important when using noisy grayscale datasets. If a "horizontal" flagged training image is somewhat slanted, it may produce an ambiguous solution between horizontal and diagonal in the network and make training extremely tedious as the network repeatedly fails to classify either the specific image or other images as the neuron weights are constantly pulled back and forth in an attempt to converge. While such a sample can always simply be ejected from the training data, it is much more useful to simply reclassify the image as "unclassifiable" in the dataset, providing the network an example of an ambiguous input and better equipping it to handle novel instances of such inputs in the future when handling live data.

Using ADC readings from our motherboard MCU occurred to be the most efficient and user friendly solution. By feeding the three hidden layer outputs, and final layer outputs back in to the ADC3 channels of the STM32H743, we were able to see readings of significant voltages at hand. Channels being used of the onboard ADC3 peripheral of the STM32H743 include channels 13-18. These voltage output readings are then fed in to three LED voltmeters which will be used to determine the highest output amongst the three. This will serve as the TACOCAT's final prediction with a nice graphical display of precise voltage readings. This will also allow the user to see exactly how large of a voltage difference is between the correct and incorrect prediction, which serves as the networks certainty of it's prediction.

5.6 Modified Input Layer Design

The multi-layer perceptron neural network design for the 2-pixel network contains a 4-neuron input layer with a synapse circuit design consisting of a buffer and inverter operational amplifier circuit pair, that provides the complementary PB0 and PA0 voltages to the digital potentiometers for weight adjustment. These inputs for the 4-Pixel network are treated as the digital bits corresponding to the image data taken as input to the trained network. These digital inputs to the network are done in parallel from the signal processor used in the microcontroller.

It is easy to see that as the inputs to the network are incremented, the number of parallel pixel data inputs to the synapse and neurons of the input layer will rapidly increase to an unpractical number of parallel inputs to the input layer. There needs to be a more practical approach to designing the inputs to the synapses of the input layer of the network to avoid unnecessary current draw from the microcontroller, as well as a large amount of traces needed to the input layer.

A practical and efficient approach to planning the digital image pixel data to the input layer is utilizing a shift register IC. This is done by connecting a shift register chip to the signal processing microcontroller of the design. The serial data bits are be fed to the input layer's synapse circuit using serial communication protocols, either I^2C or SPI, utilizing the frequency of the microcontroller's clock. Each high or low voltage level corresponding to the pixel data is provided to the synapse circuit containing a buffered and inverted version of the signal to provide the range of weighted inputs to the neuron. Having the digital pixel input data fed to the synapse circuits serially avoids the need for multiple parallel traces to the neurons in the input layer, as a single serial trace line is connected to all the PB0 and PA0 pins of the digital potentiometers after being buffered and inverted.

To achieve the buffering and inversion of the output pixel data from the shift register, the use of a hex inverting and non-inverting chips must be used to provide the voltage weight range of the digital potentiometers as opposed to the operational amplifier buffer and inverter circuit designs used for the 4-Pixel prototype network. This is due to the input data to the synapse circuits of the input layer being provided digitally by the serial output of the shift register IC. This was unnecessary for the intermediate design, as the analog voltage levels being provided by the voltage regulators LM337 and LM317 were analog in nature and did not require the need for logic-level inversion and buffering. A hex inverter and non-inverter CMOS chip allows for logic-level conversion of the shift register's output to maintain the voltage level desired, and needed, to allow for proper weight ranges that the neural network multi-layer perceptron design needs when being trained for character recognition, which is the intended application of the final network's design. Thus, the final design of the input layer is made up of the shift register IC, providing input to a modified synapse circuit consisting of hex inverter/buffer pairs that supply voltage ranges for the weights of the digital potentiometers.

Below, in Figure 5.3, an updated block diagram of the 100-input neural network design can be seen, along with a table of descriptions of the blocks included in Figure 5.3. This diagram is generally accurate for all of the networks developed for this project, regardless of dimensions, as full interconnectivity and multi-layer networks are hallmarks of the chosen network architecture for this project. The number of B blocks per C block is always equal to the number of inputs to the network, and the number of A blocks is always equal to the number of C blocks; thus, each neuron on a given layer looks the same as and is connected identically to all of the other neurons on the same layer. The only variation is in the algorithm-determined weighting of each potentiometer feeding each neuron.

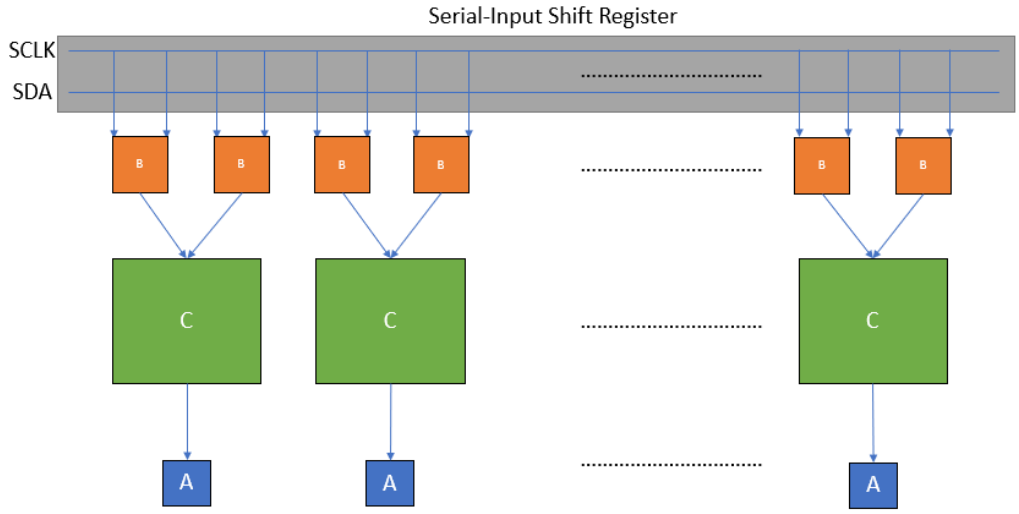


Figure 5.4 Modified Input Layer Block Diagram

Block Name	Function
A	Line Driver
B	Adjustable Synaptic Weight
C	Artificial Neuron

Table 5.1: Modified Input Layer Block Descriptions

Figure 5.3 and Figure 5.3 provide the general arrangement of each network layer. Each line driver block consists of a non-inverting buffer to buffer the output of each neuron. Each adjustable synaptic weight block consists of an inverting and non-inverting buffer pair and a digital potentiometer; each potentiometer is fed the original input signal on one end of the device and the inverted signal on the other. The output of each potentiometer is then supplied to each artificial neuron block, where it is summed via the summing amplifier stage of each neuron. Finally, the activation stage of the neurons produces the final output signals, which are then fed to the corresponding line driver blocks. The signals are then buffered by these line driver blocks and either passed forward to each of the inverting/non-inverting buffer pairs in the following layer or supplied to the final output handling section of the network, which may be hardware-based (LEDs, comparators, etc.) or software-based (a digital readout of ADC values, a message indicating the highest value, etc.).

6 Software Design

The following sections provide a description of the design process for the integrated software/firmware system that was created for the purposes of planning, testing, and implementing the TACOCAT hardware-based neural network.

6.1 Background

There is no single characteristic way to design, train, or model an artificial neural network. Some networks are fully connected, while others are only partially connected. Some are many layers deep, while others have only a single hidden layer. There are numerous activation functions and training algorithms that can be used, and there are a variety of possible methods of modeling such implementations. Since a static approach must be chosen early on to allow for a discrete hardware implementation to be built, it is necessary to weigh the performance and suitability of the many different categories of network architecture and algorithm to find the best candidate for translation into a physical device.

6.1.1 Neural Network Architecture

Before starting work on the design process, it was necessary to determine what kind of neural-network architectures would be able to successfully perform the classification tasks that are described in the project specifications. The two main families of ANN architectures that were considered were the multi-layer perceptron (MLP) and convolutional neural network (CNN). Several different CNN-based designs have shown exceptional performance in image recognition tasks, but these networks tend to be large, and the convolutional layer schemes included in these networks were expected to require a more-complex hardware design.

6.1.2 Training Algorithm

The typical training algorithm used with MLP networks is gradient descent through backpropagation. This algorithm requires the outputs of each neuron layer to be known, and the algorithm is applied one layer at a time, starting with the last layer and moving backwards through the network to the input layer (hence “backpropagation”).

In order to use this training algorithm, we must define a “loss function” that relates the output values of the neurons in a given layer to the expected output values. Once we have determined the loss for a layer of neurons during a given training iteration, we can take the gradient of the loss function and multiply it by a scalar “learning rate” coefficient to calculate an estimated set of synaptic weight adjustments for the neuron layer. The object of this process is to find a set of weight parameters that will minimize the loss for each layer.

Once the output values of the final layer have been compared to the expected values and the backpropagation album has been applied, an expected set of output values from the second-to-last layer can be inferred, and the process is repeated until the gradient descent calculations have been applied to the network’s input layer.

There are some known shortcomings of this training algorithm. Where a multi-dimensional “error surface” can be described by the network’s loss function, classification problems

will sometimes have error surfaces that have local minima as well as saddle points. Non-linear activation functions are thought to mitigate the difficulties caused by these issues.

6.1.3 Existing Software Models

There are a variety of open-source and commercially available software libraries and frameworks (e.g. Torch, TensorFlow, Keras, Caffe) that allow researchers to design and test software-based neural networks of arbitrary sizes, using a wide range of training algorithms and other parameters.

One option that was considered was the use of a ready-made framework in order to evaluate different network topologies and hyperparameter values in preparation for the hardware design phase. However, the main drawback of this approach was that any complex training algorithms that a framework might offer could be very difficult to implement in a lower-level embedded programming language without already having understood and written the code in a higher-level language.

6.2 Design Overview

There are three separate software projects that are being used to implement the hardware-based neural network. First, a software-based neural-network model is used to simulate the expected behavior of the network. This same model can also be used to train the hardware-based network. Second, the training and I/O interface controls for the network are implemented in an embedded software program that is designed to be run on a microcontroller. Finally, a user interface (running on a minimal PC platform) allows users to apply real-world inputs to the hardware-based neural network.

Python 3 was chosen as the development language for the software model/simulation component of the project. The microcontroller firmware is written in C/C++. Arduino libraries were used for the small-scale intermediate prototype network, while the 100-pixel network uses libraries provided by ST Microelectronics for the STM32H743 microcontroller. The graphical user interface (GUI) component of the project will be implemented in Python and run on a mobile-touchscreen device with a Linux operating system.

6.3 Development Tools

A range of development environments were available for both the software and firmware used in this project. Since both Python and C programming are utilized both on the network controller and on the host computer, it is not reasonable to attempt to handle everything using a single development environment. Consequently, the various development tools and environments used in this project are briefly described in the following subsections.

6.3.1 Python Development Environment

The software model was written in Python 3, and version 3.7.3 of the Python interpreter was used for development. The interpreter was run inside a Conda environment that was managed using Anaconda.

Microsoft Visual Studio Code was used as a code editor and debugging environment for Python modules.

6.3.2 Firmware Development Environment

For the small-scale prototype network, firmware development was carried out in the Arduino IDE software with the Teensyduino extension package. The Arduino IDE offers convenient integration with the Arduino code library and with the toolchains that are required to compile, program, and run software on various microcontroller platforms.

One shortcoming of the Arduino IDE is that it does not include a debugger. Any debugging is typically carried out using "print" statements and monitoring program output through a PC-to-MCU serial connection.

After completion of the small-scale prototype network, firmware was re-written for the STM32H743 using the STM32CubeIDE, which is an Eclipse-based IDE provided by ST Microelectronics. The STM32CubeMX tool was used in conjunction to the IDE in order to generate header files and initialization code for the microcontroller.

6.3.3 Version Control

Git was employed for version control in the TACOCAT codebase. While the Python code and C/C++ firmware code were developed in separate repositories these repositories can be grouped together as submodules in a "wrapper" repository that act as a central store for the overall project codebase. Hosting for remote Git repositories has been generously provided at no charge by GitHub.

While Git is sometimes avoided for firmware development processes due to its historical lack of support for tracking changes in binary data files, due to all of our project work files being encoded as text, we do not expect this to cause any issues in the course of firmware development for TACOCAT.

6.4 Neural Network Software Model Design

Since most neural networks at present are entirely software-based, idealized software simulations of the proposed hardware designs were created first. However, idealized software network simulation is not enough. Simulations of the physical behavior of the hardware must also be performed to predict and verify the behavior of the circuit once its physical implementation is brought to reality. The following subsections describe the reasoning and design process behind the software simulations used for this project.

6.4.1 Software Architecture for Simulation

It is fairly straightforward to implement mathematical models for MLPs, and because the MLP-based models have shown relatively good performance in handwriting recognition tasks, the MLP was chosen as a starting point for software simulations.

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

The simulation software, as shown in Figure 6.1, 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, and a core module that contains the construction, training, and validation logic for the neural network.

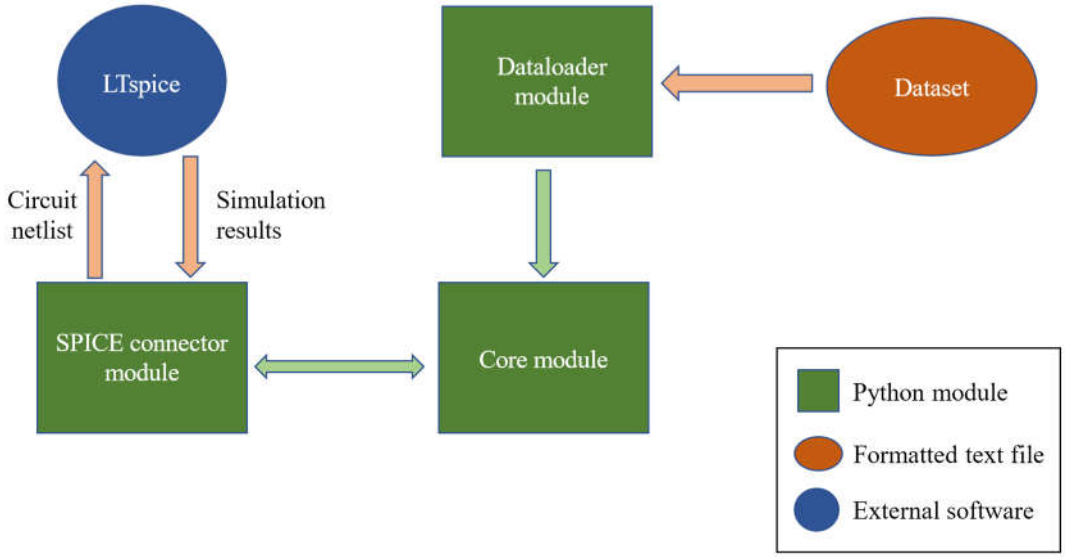


Figure 6.1: Block diagram of software architecture for simulation using SPICE hardware model

6.4.1.1 Dataloader

The dataloader module uses the Mlxtend library (Raschka) 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 28 x 28 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 Figure 6.2.

In a sense, this thresholding operation is implementing an input-layer activation function. This was considered as a possible area 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 hyperbolic tangent 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, a comparator circuit could also be used to implement the comparison operation in hardware.

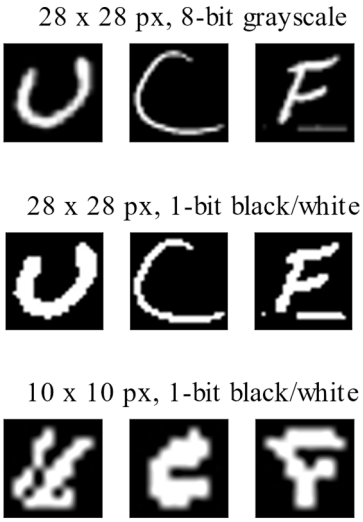


Figure 6.2: Original EMNIST sample images (top), black/white threshold operation applied to images (center), black and white images compressed/downsampled (bottom)

6.4.1.2 Simulation of Network Training and Prediction Operations

Hardware simulation was implemented using LTspice, which is a SPICE simulation software provided free-of-charge by Analog Devices, Inc. In order to integrate SPICE simulations into the software simulation model, a Python module was created to generate SPICE netlists based on the weight and input values described by the state of the software-model network, run LTspice simulations of the netlists, and parse the output from LTspice into a data structure that can be returned to the core module, which will then update the state of the software-model network.

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 has also been 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.

6.4.1.3 Additional Applications for SPICE Connection Module

The SPICE connection module was designed to be somewhat generalized so that it could also be used for testing simulated circuit behavior aside from the roles that it plays in the integrated testing of neural network training and classification/prediction. **Error!**

Reference source not found. shows the output obtained from the SPICE connection model in conjunction with an activation function test-fixture script.

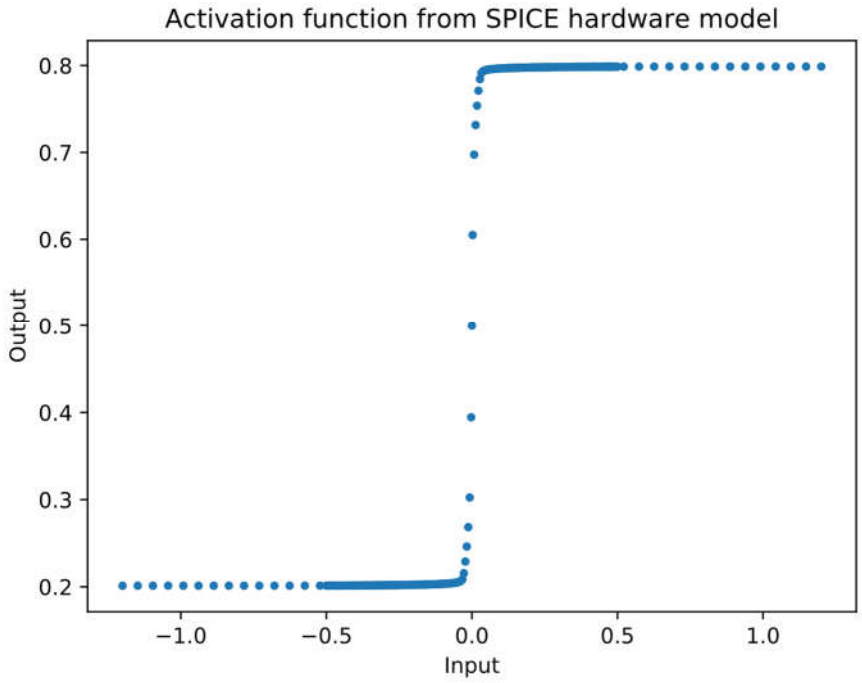


Figure 6.3: Plot of hyperbolic tangent activation function behavior based on integrated Python/SPICE model

6.4.1.4 Core

The core module contains definitions for a NeuronLayer class, with member variables to store metadata regarding the neuron layer's number of synaptic inputs and its neuron count along with a reference to a stored array of synaptic input weights, as well as a NeuralNetwork class, which contains a collection of references to NeuronLayer objects along with methods for training and validating the neural network model.

This module also maintains collections of metrics related to training, testing, and validation and includes methods that can create graphs and charts to visualize this data (examples shown in Figure 6.4).

The core module was specifically 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 can remain 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 the MCU).

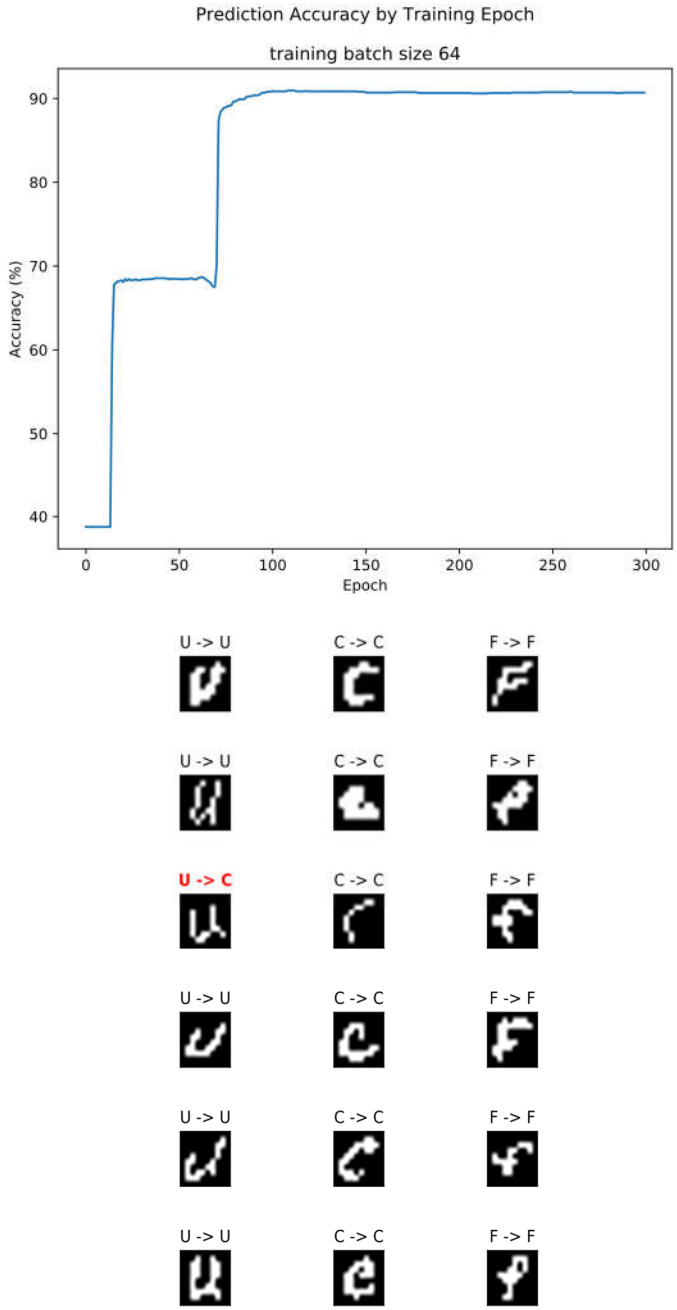


Figure 6.4: Results of software simulation for 10 x 10-pixel EMNIST recognition with 3 input classes

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 SPI, I^2C, and UART.

6.5 Firmware Design

A block diagram describing the integration of the firmware modules with the software modules is shown in Figure 6.5. One of the goals for the overall design of the integrated software/firmware system was to use consistent interfaces in similar modules. By maintaining similar interfaces between the SPICE connection module and the firmware library, we are able to simply "plug in" the firmware library connectors to the same Core-module methods that interact with the SPICE connection module as shown in Figure 6.1. This ability to easily re-use code is one of the major benefits of the object-oriented software design paradigm that is employed in this project.

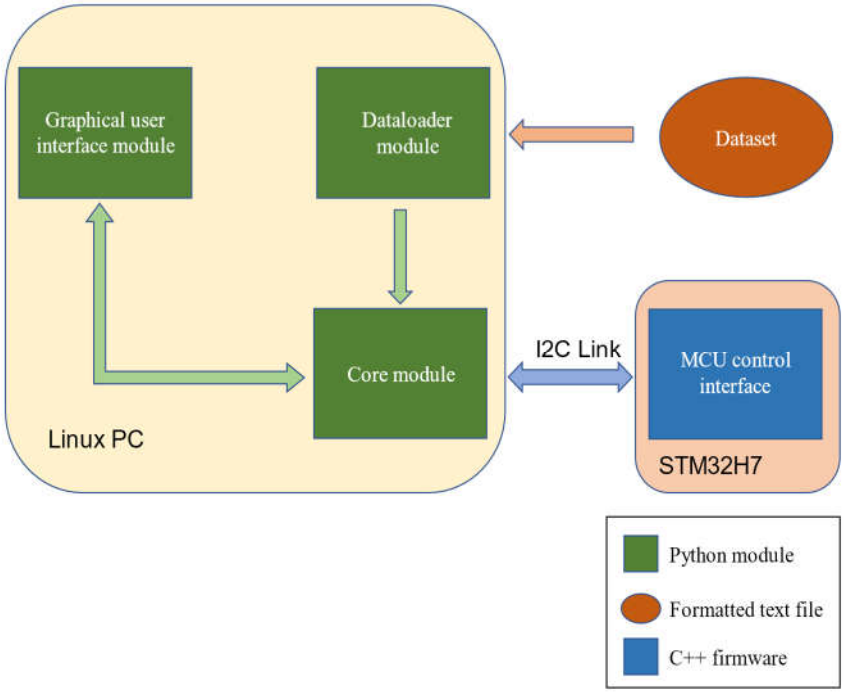


Figure 6.5: Block diagram of software/firmware architecture for hardware neural network implementation

The firmware operations that are required for the project can generally be divided into the following categories:

- Initialize circuit training process
- Adjust synaptic weight values
- Configure and activate data input signals
- Measure neuron outputs

All of the operations listed above depend on serial communication with the Python-based software model, and the function for adjusting synaptic weight values requires an additional communication interface with the digital potentiometer ICs. Functions for

transmitting and receiving data are imported from a publicly available library called SMBus2, which was designed for the SMBus protocol that is a subset of I2C. Any calls to the communication library functions are also wrapped in higher-level functions that act as interfaces that separate the implementation of the serial communication processes from the firmware's central program logic.

6.6 Software/Firmware Communication Protocol

In order to coordinate the operations of the software and firmware programs, a communication link is required. In terms of a multi-layered communication network model, this section will describe a protocol that is implemented at layer 2, directly above the physical layer.

6.6.1 Assumptions about Physical Layer

While one major design objective for this communication protocol is that it should not be overly dependent on the designs of its adjacent layers, some assumptions were made about the physical layer. It is assumed that the physical-layer protocol will be designed for sending and receiving data in groups of bytes, and it is also assumed that the physical-layer protocol does not implement any sort of error-detection or error-correction mechanisms.

It is also assumed that the communication channel may be only half-duplex and that a master-slave communication scheme is implemented, where the master device is primarily in control of network communications. In TACOCAT's case, the software modules control the master communication device, and the firmware modules control the slave device.

A typical microcontroller serial communication device should have a buffer size of 32 bytes, so it is assumed that the data frame size should be 32 bytes or less.

6.6.2 Command Codes

In very simple communication networks, all data messages may occur in the same context. For example, a network consisting of a temperature sensor and a microcontroller may only carry one type of message: temperature data that is sent from the sensor to the MCU. However, in TACOCAT's inter-device network, messages can be sent in several different contexts.

A system of command codes is implemented so that the master device can describe each message's context to the slave device. The slave device can then configure its response appropriately for the message context. Command codes are sized at one byte so that they can be sent quickly. The number of commands for this protocol are expected to be far less than the 256-command capacity for the byte data-type.

Table 6.1 shows the commands that are implemented in TACOCAT's software/firmware communication protocol. While the numeric values of the codes can be stored in a shared file that is accessible by both the software and firmware packages, the meanings of the codes and the required response behaviors must be implemented manually in each relevant code package.

Command	Code (hex value)
Initialize network	0x01
Set synapse weights	0x02
Set data input values	0x03
Read neuron outputs	0x05

Table 6.1: Communication Protocol Command Codes

6.6.3 Error Detection

Initial plans for the software/firmware communication protocol included error detection using CRC checksums. In practice, preliminary tests showed that the I2C datalink performed well without any error detection/correction, and the decision was made to omit these procedures. This saved time and design effort, and it also avoided lower communication throughput due to extra calculations and data transmissions for CRC checksums.

6.6.4 Communication Sequences and Error Correction

Communication operations between the software and firmware packages can be characterized by one of the two following communication modes: the master device requests data from the slave device, or the master device writes data to the slave device. Command codes 0x01 and 0x04 fall under the category of "data requests", and command codes 0x02 and 0x03 fall under the category of "data writes".

The Request and Write modes both begin with a command data frame (depicted in **Error! Reference source not found.**) that contains the command code and an expected number of bytes to be transmitted. When the slave device receives this frame, it configures its program state so that it will be ready to begin the transmission type specified by the command code and then sends a single-byte acknowledgement message.

6.6.5 Timeouts

In addition to bit-errors in the data transmissions, the possibility of data timeouts were also considered. Timeout procedures vary between sending and receiving devices, but for either case, a timeout period and a maximum number of re-try attempts should be specified as program parameters in the firmware and software code implementations of the communication protocol.

If a receiving device does not receive an expected data frame, it can continue to send a single-byte acknowledgement representing the sequence number of the next byte that it expects to receive until the maximum number of attempts is reached, at which point some action might be performed to notify the user of an error condition.

Similarly, if a sending device does not receive an expected acknowledgement byte, it can continue to re-send the previous data frame until the maximum number of attempts is reached. At this point, it can also try re-sending its previous command frame until the appropriate acknowledgement is received or a maximum number of attempts is reached. This requires a logical block in the receiving device that aborts a receiving operation if a

command frame is received when a data frame is expected. If these attempts all fail, the sending device might notify the user of an error condition.

The I2C datalink used in TACOCAT has pre-set timeout periods on the microcontroller and on the PC. When the microcontroller detects a timeout, it returns to the state of waiting for a command code. If the PC detects a timeout during transmission, it will restart the transmission process from its beginning.

6.7 User Interface Design

The user interface components that are implemented in software are mainly concerned with gathering handwritten-character input data from a user for testing/demonstration purposes. A camera was considered for this purpose, but a digital touchscreen interface was chosen instead as it seemed to be a likely source of real-world input for similar hardware-based neural networks in mobile devices. Image data from the digital touchscreen interface can also be fit into the appropriate number of pixel inputs for the given network with minimal processing required in the software. A screenshot of the user's view for the touchscreen interface is shown in Figure 6.6

Because of the wide availability of phone and tablet devices that integrate touchscreen inputs with web-browsing capabilities, an Android device running a JavaScript-based web application based on a framework such as React or Angular was strongly considered. However, the steep learning curve for mobile app development frameworks and the complexities of creating a direct interface between a microcontroller device and a JavaScript-based web application appeared to be significant enough to steer the design process away from this approach.

Instead of a web-application implementation for the user interface, we chose to focus on a graphical Python-based interface that uses a peripheral touchscreen device for input and display purposes. The freely-available Tkinter package for Python offers an interface

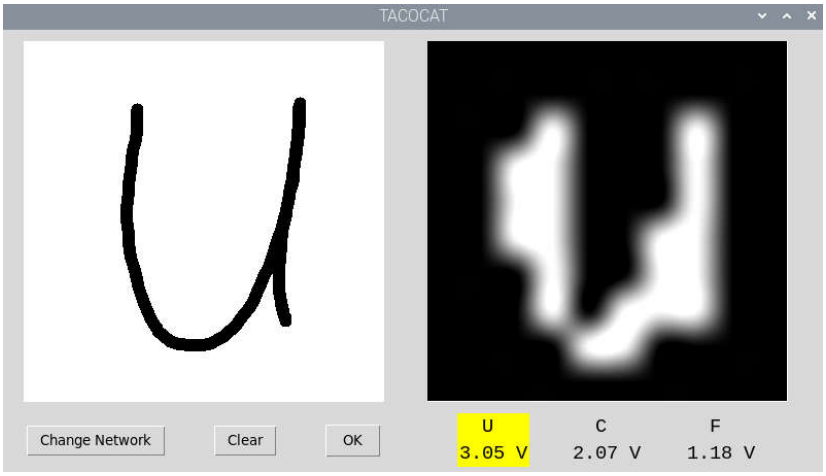


Figure 6.6: Screenshot of the graphical user interface

between Python and the Tk graphical user interface library, which was originally developed for the Tcl scripting language.

Along with a Python library to handle the generation of graphical output for the user interface, we use the native support for touchscreen control input that is included with Linux distributions such as Ubuntu. Integrating these two components allows the application to display information to and gather information from the user.

7 System Fabrication

A significant amount of component testing, network prototyping, and equipment optimization was performed during the breadboard testing phase. While breadboarding is extremely useful for such activities, it becomes impractical to produce any significant circuit beyond a few neurons due to the inherent variability of the breadboard connections and the dozens of jumpers and through-hole components used. Because any network implemented as in this project is extremely sensitive to more than a few millivolts of signal distortion, breadboarding is not an acceptable long-term approach to network implementation.

The obvious solution to this shortfall is the use of carefully designed printed circuit boards (PCBs). Through intelligent PCB design, jumpers and leads that were once several inches long can be efficiently reduced to a few millimeters, and components can be placed and oriented at will to reduce the footprint and complexity of the design. While there are a few limitations and considerations that must be taken into account during PCB design, the overall result is a significantly more reliable, compact, and portable device.

7.1 PCB Design Software

PCB design is largely accomplished through the use of semi-automated design programs which allow the user to lay down a device's schematic and realize the equivalent PCB layout. A great multitude of design programs are available for schematic capture and board design, ranging in functionality from barebones PCB design tools to software suites with schematic capture, automated trace routing, and 3D board visualization. Predictably, these programs range wildly in price from being completely free with or without an educational license, such as Autodesk's EAGLE, to costing nearly $10,000, as in the case of Altium's Altium Designer software.

Because of the availability of educational licenses, ease of use, and relative familiarity due to use in previous coursework, EAGLE was selected for use in this project. EAGLE provides linked schematic capture and board design, allowing for on-demand updates to device schematics to be translated onto a board layout, streamlining the design process. The availability of custom package, symbol, and footprint generation in EAGLE enables the creation of new hardware symbols and layouts when a custom design is necessary, or when a pre-designed component symbol is either inaccurate or unavailable. The "ratsnest" command and autoroute tool both greatly reduce the time required to lay out the board traces by displaying the most efficient connection routes as airwires, as shown in Figure 7.1, and by automatically routing traces between these points, respectively. As components are shuffled around on the board layout, the ratsnest command will automatically recalculate the most efficient routing connections in order to minimize trace length and connection overlap. Use of the autoroute function will automatically lay traces in accordance with these airwires; several different solutions will be presented simultaneously, with each solution prioritizing different trace directions and different layers. As a result, it is possible to produce clean, useful board layouts and save significant amounts of time and effort when dealing with less complicated designs. Because autoroute

is not able to automatically create voltage planes and can often produce unnecessarily long or complex routes for certain traces, autoroute should be judiciously and supplement manual routing and board design.

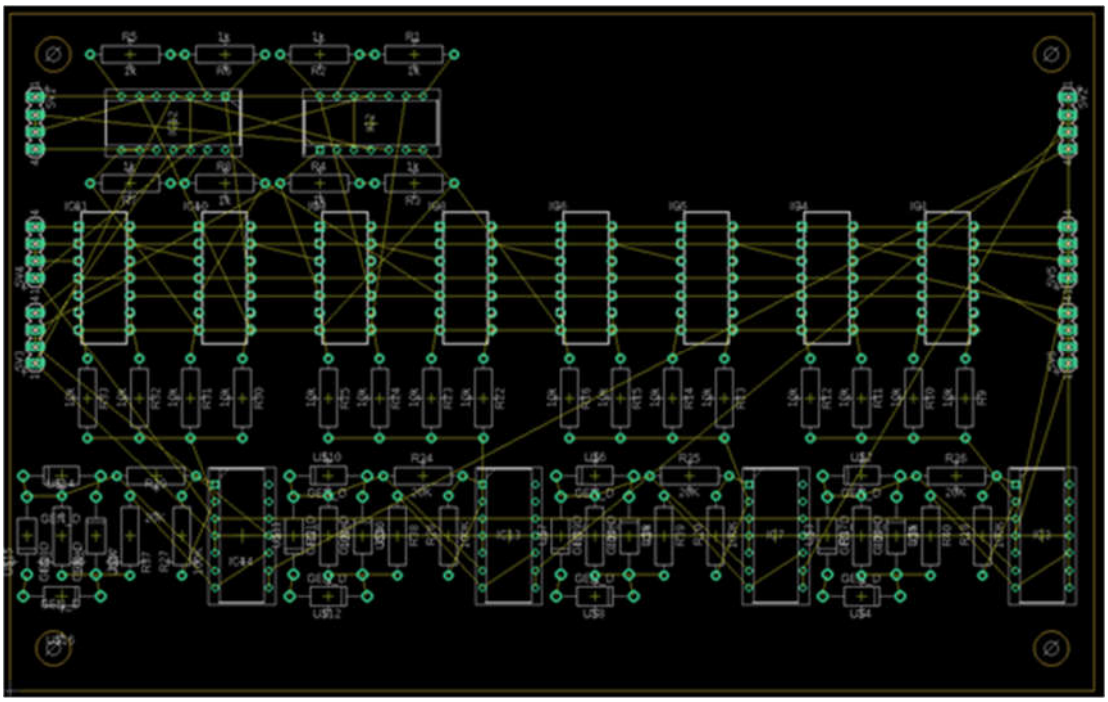


Figure 7.1: A 4-input, 4-output layer board with airwires displayed

7.2 PCB Design Philosophy

Early in the design process, it quickly became apparent that while single-board implementation of the network was possible, it would be difficult, if not impossible, to produce a two-layer PCB design for a single-board neural network. The primary factor in this difficulty is the high level of interconnectivity between each layer of the network; each neuron must provide two connections, an inverted and noninverted signal, to the input of every neuron in the next layer. For instance, with 10 neurons in two adjacent layers, a total of 200 logic connections between each layer are required, along with 100 potentiometers to receive these signals. Because there are already numerous other connections present on the board, such as the analog and digital rails, SPI connections to the potentiometers, and amplifier circuit resistor traces, it quickly becomes impractical to produce a single two-layer PCB for the entire network, even when dealing with smaller networks as in the case of the 4-pixel test network. While a DAC implementation as discussed previously would eliminate this problem, it would produce problems of its own by dramatically increasing the amount of digital logic lines entangled in the analog-heavy board and would require constant handling by the microprocessor to update the output of each DAC on the circuit every time inputs were changed.

Another consequence of using a single PCB implementation is a significant reduction in versatility. Because all the connections are static within the board, it is unreasonable to attempt to modify the dimensions of the network, especially when using surface mounted parts. While it would be possible to reduce the size of a layer, it would be impossible to increase the size of the layer beyond whatever limit was present on the board. For instance, the least-effort method to reduce the size of a layer would be to remove the potentiometers on the following layer that relied upon the output of the neuron being removed and jumper the data in and data out pins together for each potentiometer removed. However, this would not only require a potentiometer to be removed from every neuron on the following layer but would leave excess hardware on the board. Similarly, it would be very difficult to remove layers from the network and would be impossible to add additional layers to the board.

A third problem with a single-board implementation is the difficulty of obtaining output measurements from each neuron, especially when needed for ADC data for training purposes. Because the microcontroller's training algorithm requires ADC voltage conversions from the output of every neuron in the network, the output of every neuron must be routed to a location accessible to the ADC. Because it is not reasonable to attempt to measure output voltages from multiple locations across the board, it is necessary to route these connections to a single location for multiplexing. However, due to the previously mentioned high density of connections on – and large size of – full-network boards, traces must be run from each neuron's output to a central location, requiring unacceptably long traces to be routed across the board.

These factors point to the need for a more versatile, modular design. While the most obvious answer to this problem may be to simply use one neuron per board with plenty of input and output pins, allowing for completely arbitrary network dimensions, this requires that every single connection to and from the neuron, both analog and digital, be repeatedly jumped from board to board. Because the SPI daisy-chain configuration can potentially involve dozens of devices in series, and because the calibration of the network relies on very precise voltage readouts, there is far too high a risk of signal attenuation and corruption to employ this approach. An intermediate – and more feasible – method of modular implementation is to create individual boards for each layer of the network. This approach allows for the easy addition, removal, or adjustment of layers simply by exchanging the boards, especially if a standard "format" of board is used. For instance, if a 4-neuron layer board is used as the standard format of the network, a 4-4, 4-4-4, and 4-4-4-4 neuron network could all be fabricated simply by connecting successive boards in series with the previous layers. Not only can layers be added and removed as desired, but the layers can easily be reduced in dimension as needed by omitting components. Because the only jumpers required in this case would be across the serial input and serial output SPI pins of the missing potentiometers, both the issue of excessive jumpers with the individual-neuron implementation and the issue of superfluous hardware and lack of modifiability of the single-board implementation are mitigated.

Additionally, boards can be created with non-equal numbers of inputs and neurons. For example, if a 4-8 network with 8 inputs is desired, this network can be achieved using a single 8-input, 4-output board design. One of these boards can form the first layer; to produce the output layer, one can simply populate two of the boards with only half of their potentiometers, producing 4-input, 4-output boards. By placing these boards in parallel, one can produce a 4-input, 8-output layer. If an additional 4-input, 4-output layer hidden layer is desired between these two layers, it is very easy to simply half-populate another board and connect it between these two layers. Similarly, if an 8-input, 8-output layer is desired, two of the 8-input, 4-input boards can be placed in parallel. Thus, the maximum size of a layer is constrained by the number of inputs available on a single board.

Due to this high degree of flexibility, it is possible to realize a very wide range of network dimensions and implementations using the same small pool of hardware. Networks with large numbers of inputs that would be otherwise impossible to accomplish on a single two-layer board can be achieved via parallel operation as described previously. However, width is not the only dimension that benefits from this approach. Because neural networks benefit more from increased depth than increased width, a network's width has a strong impact on the accuracy of its outputs and on its convergence time during training. Depending on the difficulty of the task, such as differentiating between capital P and capital F, a shallow network may completely fail to converge regardless of training duration. Thus, network hardware that is otherwise unsalvageable for use in a certain application can be successfully used and reused in more complex applications simply through addition of hidden layers to the network.

7.3 PCB Design Limitations

A number of PCB design limitations were observed for this project, especially due to the use of multiple copies of the same PCB design. The first and most stringent limitation observed was the requirement that any and all PCBs used be designed with a maximum of two layers. While single- and two-layer PCBs are fairly straightforward to manufacture, boards with higher layer counts become exponentially more expensive to produce, as multiple layers of substrate are required. Because this project is self-funded and is already utilizing multiple boards, it is entirely reasonable to simply reduce the complexity of the boards as needed to maintain reasonable two-layer designs.

Another limitation for this project is the physical size of the boards. Because numerous boards are required for this project and board cost is closely related to the total area of the board, it is well worth the effort to minimize the dimensions of the board to as high a degree as reasonably possible. However, care must be taken not to take this design compression to an excessive degree, especially since board population in this project is performed by hand. While board size is not a particularly stringent limitation for this project when dealing exclusively with surface mount components, the through-hole versions of the components used in this project are, on average, approximately three times larger on average than their surface mount counterparts. While the use of through-hole components dramatically improves the ease of assembly, there must be enough space between component pads to avoid undesired contact between components after soldering.

7.4 PCB Design Preferences and Practices

While the design limitations for the PCB needed to be observed, several other practices were observed to improve the functionality and ease of use of the boards and decrease the likelihood of interference or signal aliasing.

7.4.1 Voltage Planes

The first design practice was the creation of a ground plane on the second layer of each board. While single traces can be used to route ground signals as with any other signal, this is generally a bad idea due to the non-zero resistance of traces. Because high currents through these traces can potentially produce significant voltages on device pins that should nominally be grounded, multiple devices that should all be grounded can operate at different ground levels, dramatically increasing the risk of noise or interference on the network, especially when this occurs with voltage rails. This can be counteracted through use of a ground plane, which fills the unused board space on one layer of the PCB with copper and ties it to every ground signal on the board with vias. This is extremely helpful in simplifying routing and is unquestionably useful when using through-hole components, as the vias of the components themselves can be used to bring the desired pins to ground.

7.4.2 Via Minimization

While vias are useful in combination with voltage planes, their use should generally be minimized. Vias often have higher resistance than comparable lengths of trace, so repeatedly jumping between layers of the board with vias can introduce significant resistance to the trace and provide more opportunities for noise to interfere with the carried signal, especially when handling high frequency signals such as the SPI clock and data lines. While not as applicable to this project due to its small scope, devices should have the number of vias minimized to speed up production and reduce costs, especially when dealing with mass-produced products.

7.4.3 Efficient Component Spacing

Another design consideration is component spacing. While every attempt should be made to minimize unused board space, especially on commercial products, care must be taken when considering the device must actually be populated and does not magically complete itself. Normally, the only major thought that must be given to this factor is when dealing with high-density surface mount components, as the limitations of the pick-and-place machine being used for assembly must be respected. However, in the case of the boards used in this project, population is performed by hand. High-density component layouts are not practical to deal with, especially when using small components (below 0603), as beyond this point the silkscreen quickly becomes unreadable. Similar care must be given when designing the layout of through-hole components, as each of these components must be manually soldered instead of cleanly mounted using a solder screen. Since there is inherent variation when individually soldering so many connections, it is well worth the effort to ensure there is a reasonable margin between pins and devices to avoid undesired contact between solder joints.

7.4.4 Input and Output Pin Alignment

A final consideration specific to this project is the alignment of the input and output pins of each board. In Figure 8.1 above, shows the PCB schematic (ground layer omitted for clarity) of the 4-input, 4-output board used in the 4-pixel test network; three sets of four pins are visible on the upper left and upper right side of the board. Starting at the top, the first row of pins, offset from the next two rows, holds the four network inputs to the layer. The next row of pins holds the four power rails for the circuit: +5 V, +1.65 V, -1.65 V, and -5 V. The final row of pins holds the ground rail and the three SPI lines: clock, data, and chip select. Since all communication used in this project is serial, it is very straightforward to simply connect the SPI communication lines in series between each board. Because these pins are located next to the edges of the board and correspond to the pins on the opposite end, each layer can be daisy-chained together using very short jumpers, minimizing any voltage drop or signal interference that would otherwise occur. This also provides a much more intuitive visual display of the circuit, as each layer has its own discrete board and can clearly be seen in the order the network operates.

7.5 Prototype PCB Schematics

Because this project relied on multiple boards and had both a prototype and final network design, multiple board layouts and schematics were used. However, an early iteration of the 4-input, 4-output PCB used for the layers of the 4-pixel test network can be used as an example, as it is a fairly simple layout which, while fairly reflective of the given design preferences and practices, can be improved in a number of ways. The design shown in Figures 8.2 and 8.3 was produced through manual placement of components and the use of ratsnest to minimize trace length and connection overlap; the routes were then completely generated through use of the autoroute function.

The autoroute tool was run using the “high effort” option, which generated many more unique solutions than normal, allowing the most ideal routing layout to be selected. Because voltage planes are not generated when using the autoroute function and are often necessary – especially when using ground planes – to avoid noise generation and signal distortion, a routing solution which prioritized the top layer was chosen to allow for the creation of an as-intact-as-possible ground plane on the bottom layer of the board.

While the autoroute tool can save enormous amounts of time, attention must be paid to the details of each solution. Heavy use with complicated designs can result in extremely long and inefficient traces, increasing losses and taking up unnecessary amounts of space on the board. However, the autoroute tool can be used as inspiration for a final layout; problem traces can simply be deleted and run manually, or existing traces can be adjusted in such a way to allow for better trace routing.

As shown in Figure 7.2, circuit components were laid out manually with a flow similar to that of the schematic to improve both the routing and the appearance of the circuit. As previously mentioned, the autoroute tool was used to produce a top-biased routing strategy to allow for the creation of a large, intact ground plane. By maximizing the number oftraces on the first layer (red) and minimizing the number of traces on the second layer (blue), the risk of ground loops or other noise-generating phenomena can be significantly reduced. As

shown in Figure 7.3, the ground plane is nearly completely intact, with only a few small sections of the bottom layer isolated from the ground plane.

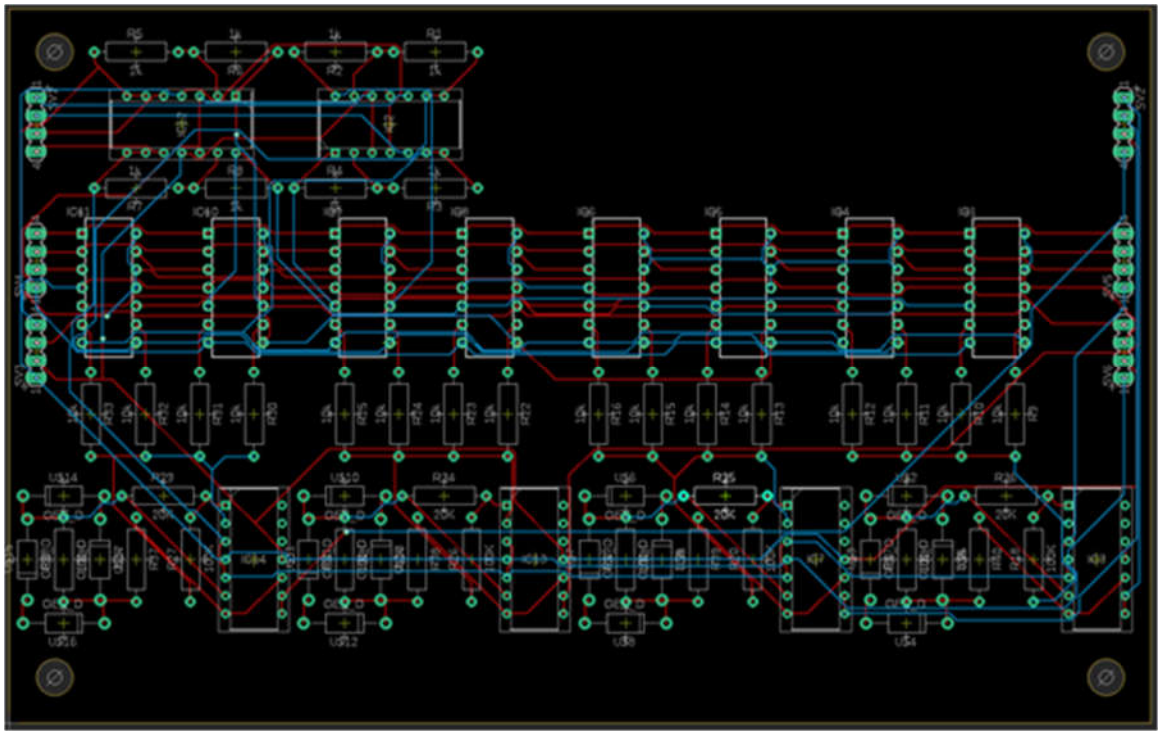


Figure 7.2: 4-input, 4-output test network layer PCB design

The ground plane can be easily generated on the second layer via the polygon tool. By creating a polygon surrounding the entire second layer of the board and naming it “GND”, the name of the ground signal in EAGLE, the ratsnest command can be used to automatically remove any ground traces on the board, fill in the second layer with a copper pour, and connect any ground pins directly to the ground plane. Because all of the components used in this design are through-hole, voltage planes are immensely helpful in reducing the number of traces and vias on the board, especially when a commonly used voltage level, such as ground, can be made accessible across the entire board. One can note that the number of vias (excluding the through-hole component connections) is very low, with only around half a dozen appearing on the board; a natural consequence of eliminating any ground traces from the board is that any traces that would have otherwise required vias to cross can simply be routed straight to their destination on the first or second layer instead of requiring multiple jumps back and forth between the two.

One shortcoming of this design is the fairly substantial amount of unused space on the upper right and lower portions of the board. While this could largely be resolved by widening the board and moving the input buffers from the top left to the left side of the design, this would result in a board over twice as long as it is tall, resulting in more fragile boards and unnecessarily long arrangements when daisy-chaining multiple layers. As a consequence of increased length, the length of the signal traces along the network would become non-negligible and would introduce significant voltage losses across the network.

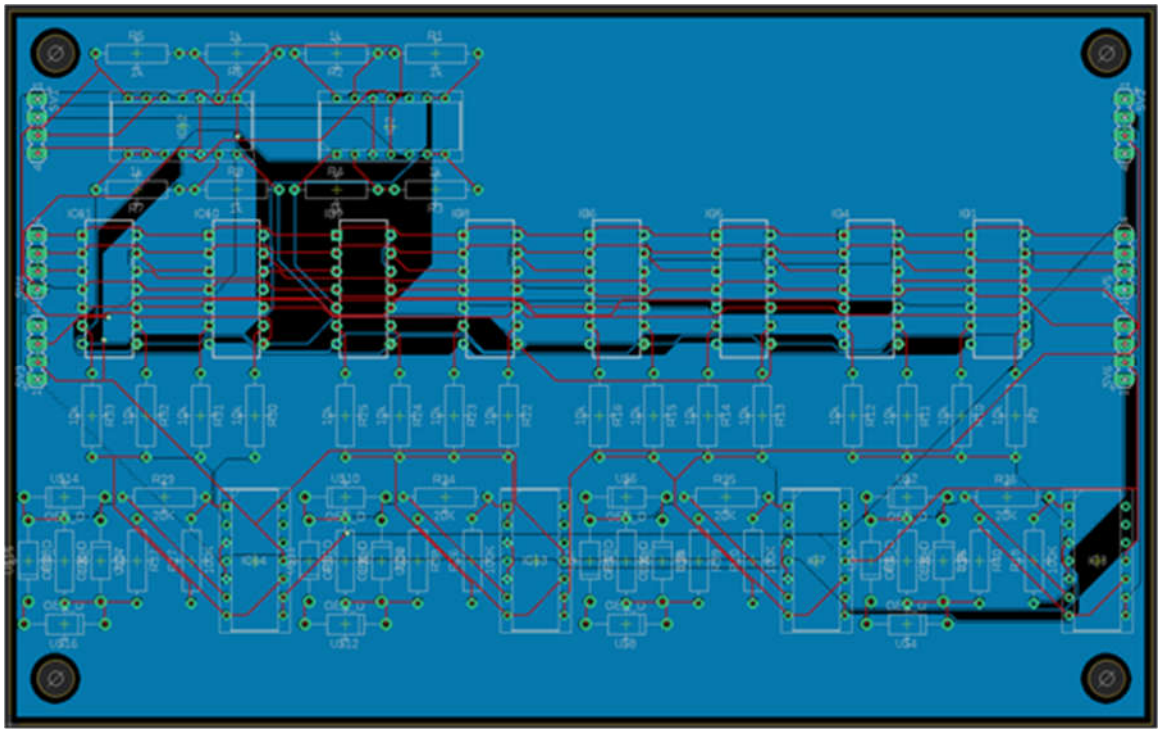


Figure 7.3: 4-input, 4-output PCB design with ground layer

Arranging the potentiometers and neuron infrastructure in series, as shown in Figure 7.2 and Figure 7.3, is effectively non-negotiable, as such a linear configuration minimizes trace length and avoids the need for more than a handful of additional vias in the entire board. This board also demonstrates the importance of selecting ratios of inputs to outputs that match well not only from a logic standpoint but from a physical one; since the footprint of the synapse hardware for each neuron is roughly equivalent in size, the layout of the board is dramatically simplified and does not require complicated routing or enormous swaths of unused space.

This board serves as an excellent example of both the benefits and shortcomings of the autoroute function. There is a significant number of components on the board despite its relatively simple nature, and many require connection to a large number of other components. While many of the traces are quite short owing to the thoughtful component arrangement on the board, it is still tedious to manually route so many connections. However, there are drawbacks to autoroute usage. One may note that the second row of pins on the right side of the board, carrying the positive and negative analog and digital voltage rails, feature extremely long traces all the way down the right side of the board, across the bottom, and back up the left side. While this does not have a significant impact on the digital signals, which are routed similarly, this is a significant issue when dealing with voltage supplies and analog signals. Because the autorouting solution for these voltage rails was to route these traces through the relevant devices before terminating the connections on the other side of the board, there will inevitably be non-negligible voltage drop due to the devices' current draw across these traces. A better solution would be the inclusion of an additional, much wider trace directly across the board to aid in power

transmission, or, much more preferably, the use of small voltage planes on the otherwise unused upper layer of the PCB to eliminate many of the traces entirely.

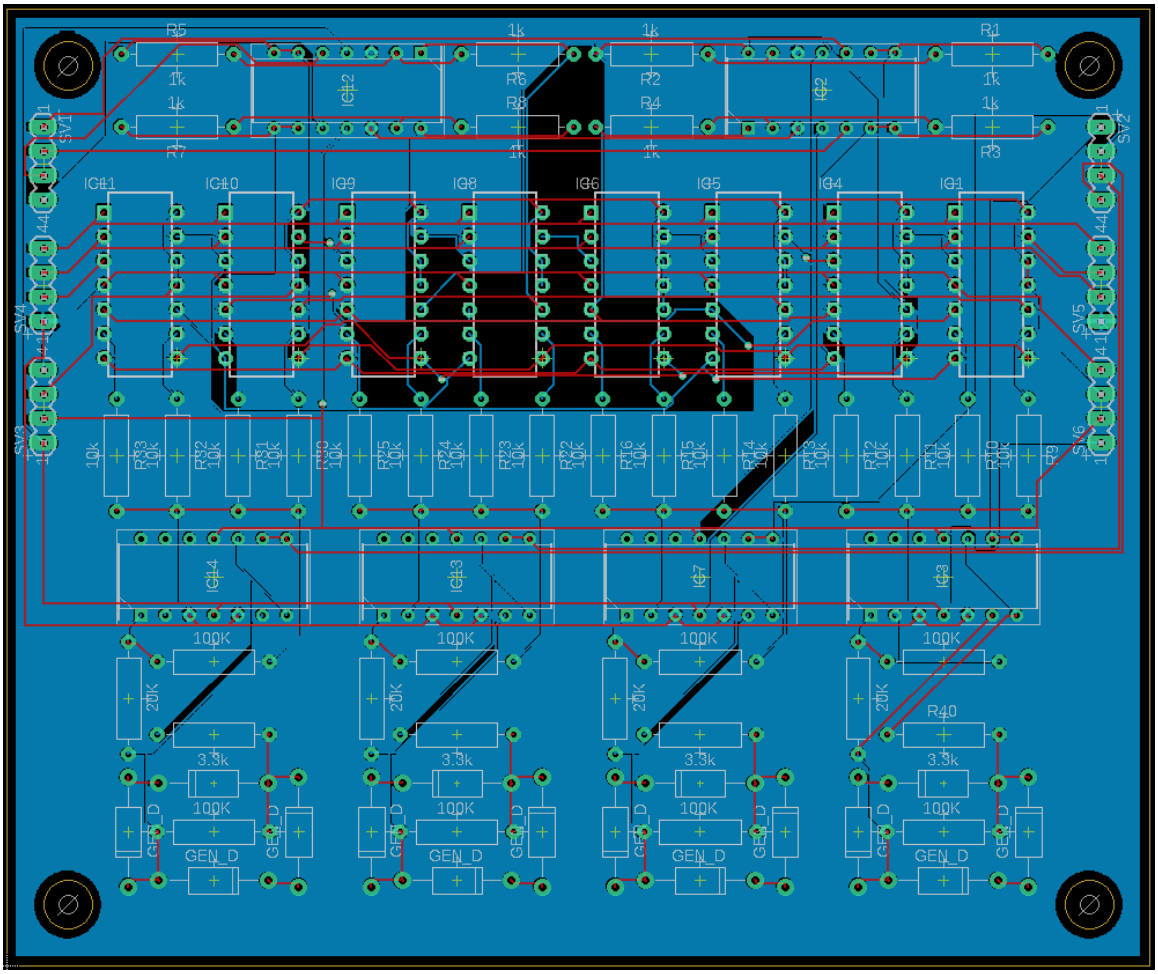


Figure 7.4: Second iteration of initial 4-input 4-output board

One can note in Figure 7.4 that a number of these problems have been addressed. By adjusting the component spacing and orientation – specifically, compressing the buffer hardware and rotating the neuron hardware 90 degrees – a significant reduction in board dimensions, excess trace length, and unused space results. While a few long traces still occur, predominantly when handling the output signals from the network, the overall result is a decrease in trace length for much of the analog signal pathways. While the number of vias has increased slightly, this is an acceptable tradeoff considering the substantial improvements in several other regards. Further inclusion of partial voltage planes on the unused areas of the top layer to handle the relevant signals, enhancing conductivity across the board and further reducing the number of traces present.

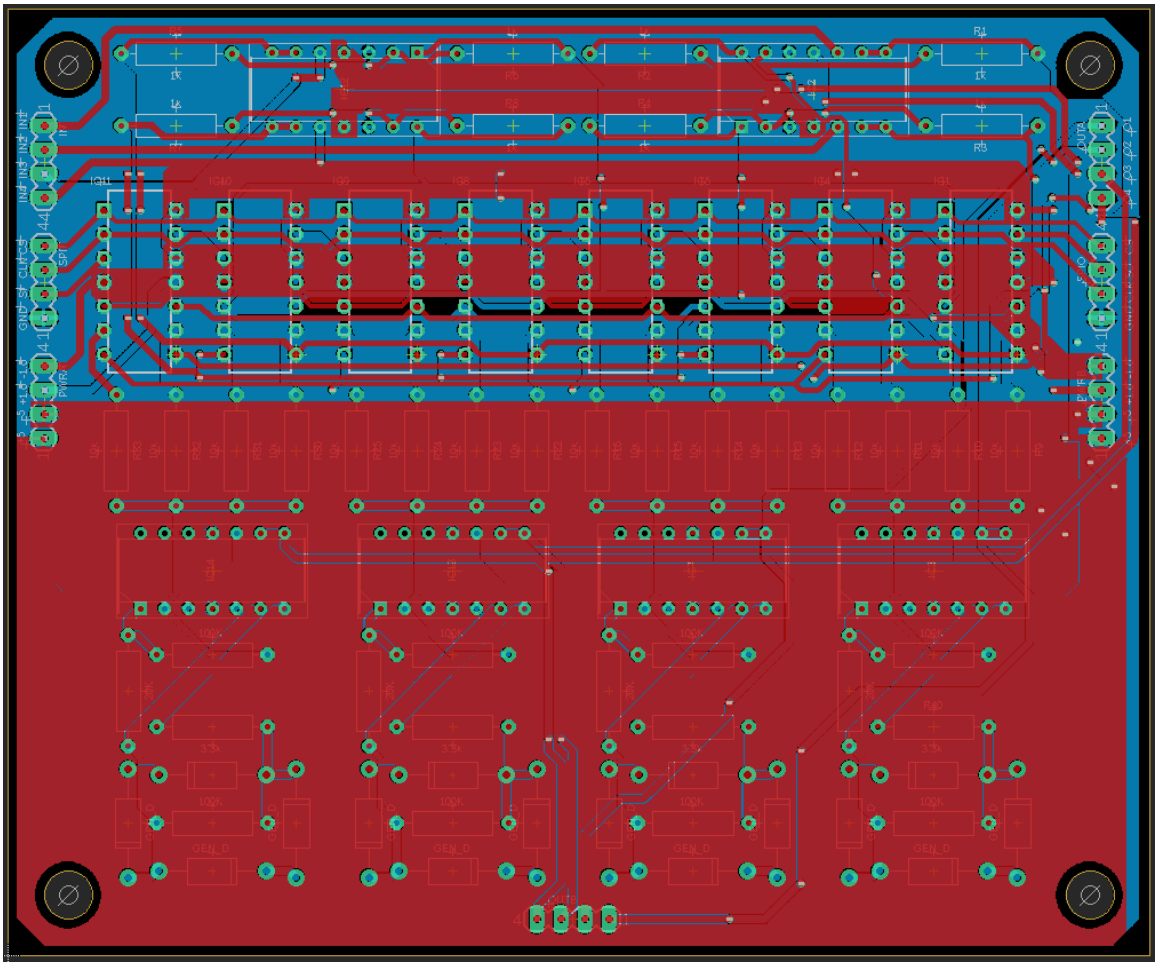


Figure 7.5: Final iteration of 4-input 4-output board

Finally, Figure 8.5 shows the final iteration of the PCB design. Heavy use is made of voltage planes on both the top and bottom layers of the board, and traces are shortened through the strategic use of vias and with 45 degree angles. While the number of vias is significantly higher than in earlier iterations, their use in this case allows for both shorter connections and significantly thicker traces, minimizing voltage loss across the board. Additionally, the use of vias in certain chokepoints encourages the integrity of the voltage planes, minimizing the amount of empty space; comparing Figures 8.4 and 8.5 clearly showcases this reduction.

After some discussion, the decision was made to include a second set of output pins at the bottom of the board to allow for easier access for ADC readings when training the network. Since the other set of pins would otherwise be taken by the following board when daisy chaining the network's layers, inclusion of another output pin set eliminates the need for additional hardware, such as breadboards, to allow for the ADC connections to be made. This simplifies the circuit board in a way that makes a modular approach to constructing a larger network much easier.

7.6 Final PCB Schematics

With the results of and lessons learned from the design of the prototype network, the design of the final network could be completed. As with the prototype network, a modular approach was taken to creating the network. With the network's dimensions chosen to be 100-3-3, it was readily apparently that it would not be practical to achieve a single-board implementation of the final network. The decision was made to separate the synapses for each of the 3 neurons of the first hidden layer into separate dedicated daughterboards, which took all 100 inputs, applied synapse weighting, and summed the resulting signals before returning a single result to the motherboard. On the motherboard, these signals were then provided to the neurons, which functioned as usual. The motherboard was designed with an embedded STM32 controller to handle the input voltage selection, daughterboard synapse weighting, and ADC readings of both the returned signals and the output voltages of the two neuron stages. The motherboard schematics can be seen in Figure 7.6.

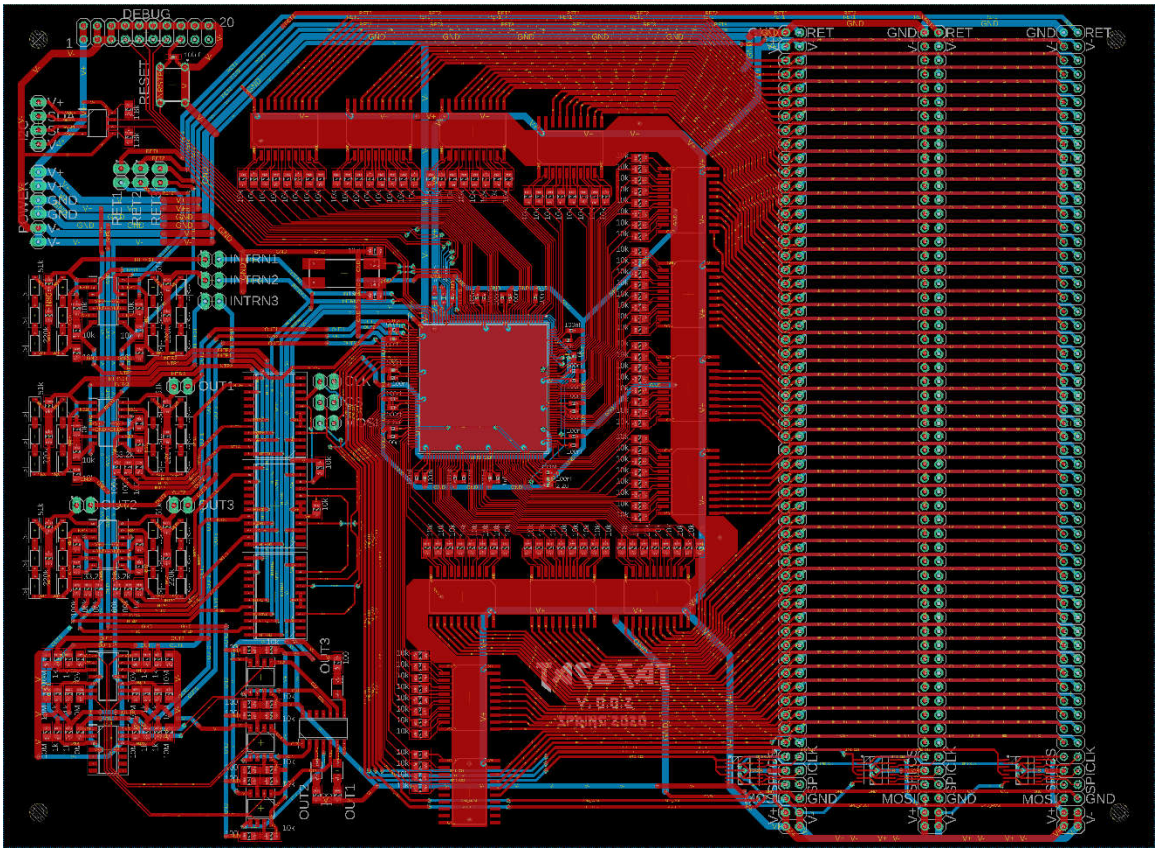


Figure 7.6: Final motherboard design

As can be seen, the final network's motherboard design is dramatically more complex than the prototype network's boards. The two neuron layers are visible on the left side of the board, with the STM and buffers in the center of the board. The three daughterboard connection slots are on the right side of the board, with small hex buffers used to buffer SPI communications to each daughterboard. A number of jumpers are present on the left side of the board to allow other neuron daughterboards to be connected in the event that the neurons on the motherboard do not function properly or other dimensions are desired.

It should be noted that some of the ground plane of the board is disabled to improve legibility; the second layer is completely filled with said plane, in reality.

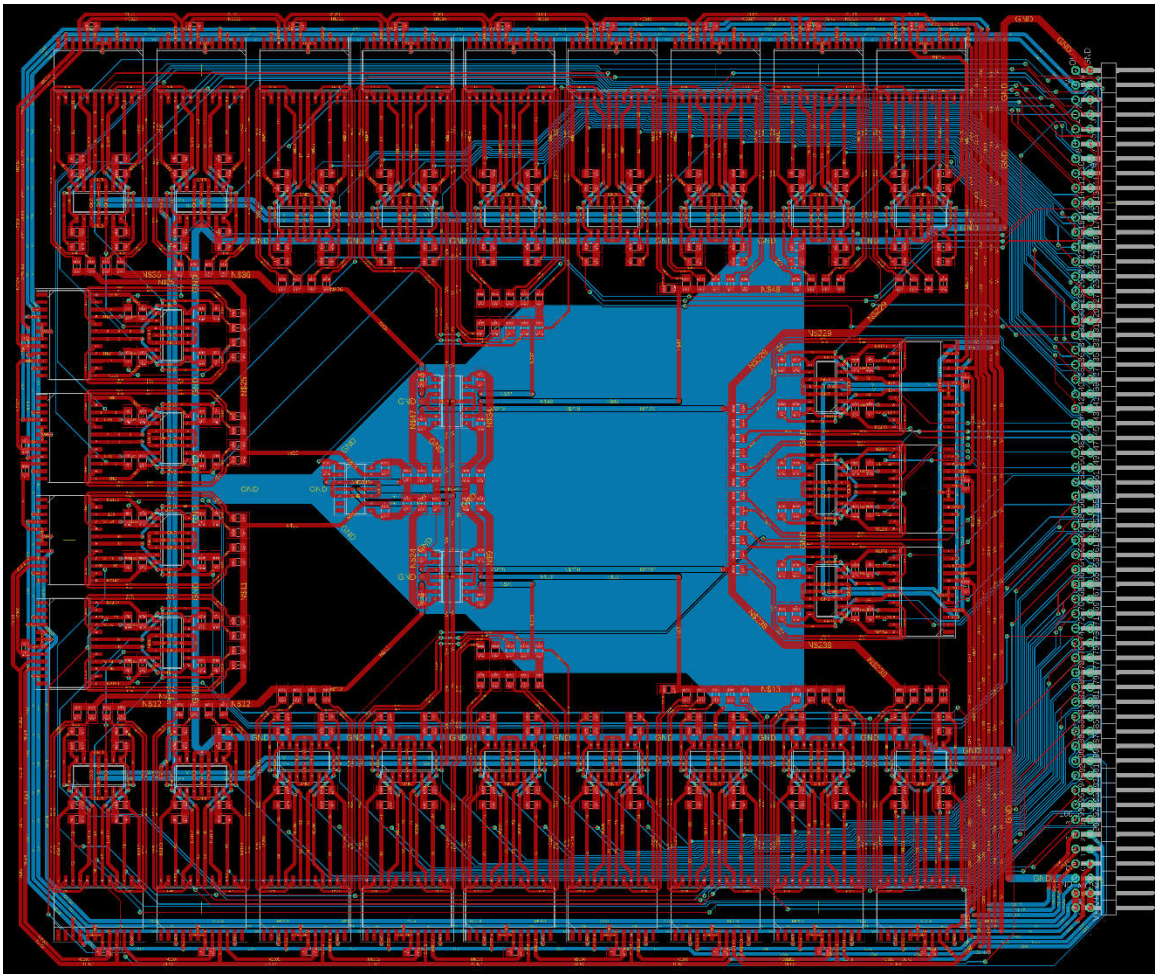


Figure 7.7: Final daughterboard design

In Figure 7.7, the PCB layout of the daughterboard can be seen. The daughterboards connect vertically to the motherboard through use of two-row 90 degree pin headers. Because of the number of traces that must cross the board on both layers, there is not much space for voltage planes. To counteract this, several large voltage busses are present on the second layer of the board, supplying ground and power to each row of amplifiers and potentiometers, with a relatively small ground plane on the second layer of the center of the board helping to minimize any ground loops that may otherwise occur. The potentiometers are arranged around the very outside of the board, while the amplifiers used to generate the complimentary signals for each potentiometer are further towards the center of the board, with the summing amplifiers themselves in the very center. The SPI clock and chip select lines are split into 3 separate signals by the hex buffers on the motherboard, allowing each signal to be divided roughly equally among the 25 potentiometers on the board to avoid heavy loading and corruption of either signal.

8 Prototype System Testing

Prototype testing is an important part of getting every major project to operate properly. This section will discuss how different tests were conducted throughout the construction of the artificial neural network. This led us in the right direction to implement solutions for any issues that occur when creating the network. There is no final functional network without necessary prototype system testing, and it is important to make sure hardware implementations of theoretical testing match the theoretical results.

8.1 Prototype Hardware Testing

Conducting tests of our hardware prototype in a logical fashion to ensure accurate functionality is a huge part of getting our project to operate. Every voltage and current level needs to be within certain boundaries. To make certain they are, we measured almost every part of our circuit to see that our physical circuit was lining up with our expectations based on schematic simulations. The following subsections outline the different tests we are conducting to confirm that several key circuits in the neural network would function as expected.

8.1.1 Individual Neuron Prototype Testing

Prototype hardware testing is a tedious but extremely necessary process in constructing a functional analog neural network. Starting from ground zero, every output of every circuit requires testing along with several different significant currents drawn throughout an individual neuron circuit. Our most initial prototype of a full individual neuron actually used analog potentiometers to weight the four inputs that feed in to the inverting summing amplifier stage. We created the circuit based on an existing project that actually used memristors as synapse weighting devices. After doing some tests with the circuit based on a previous implementation of an analog neural network, we made some modifications that better fit a circuit using potentiometers as synapse weighting devices. We got some exciting results initially, but knew we needed to spend some time redesigning the circuit for our network. The first test we conducted was to look at the transfer function obtained between the input of the summing stage and the output of the activation stage. Results of the hyperbolic tangent response on an oscilloscope using an x-y plot from our initial individual neuron circuit are shown below in Figure 8.1. The hyperbolic tangent response is apparent, but the output waveform is significantly different from the expected waveform. Further design work would be required to produce the desired activation function behavior.

Because the original neuron hardware was designed to handle memristor junctions, a significant amount of excess hardware was present in the neuron design to allow for intermittent high current draw for programming purposes. After the first attempt at an individual neuron circuit, the circuit was modified by removing several components and designing simple input buffer circuits. Components removed included a basic current mirroring transistor configuration that was used to regulate currents for the memristors that was no longer necessary and several neuron amplifier stages that were consolidated into the two stages used for this project's design. Additionally, due to the potential variability and poor scalability of an analog potentiometer implementation, a digital potentiometer implementation was chosen, as they are controllable with a microcontroller, allowing for

greater precision and responsiveness when making weighting adjustments. These changes provided a much more reliable hyperbolic tangent response to work with.

After constructing the new neuron circuit, it was possible to gather a new response curve, given in Figure 8.2, from the oscilloscope. This was the much cleaner hyperbolic tangent response that was sought after; one can see how much cleaner the response is, as the implementation now uses much higher resistance digital potentiometers instead of the lower impedance analog potentiometers originally used, avoiding thermal drift as the devices heat up due to high current draw. Not only was the hyperbolic tangent response function at this stage thoroughly tested during prototyping, but the output of the summing amplifier stage was tested to make sure the input voltages were being summed properly, since the activation function is wholly dependent upon the output of the summing stage. Additionally, the voltage levels on each of the digital potentiometers' wiper pins were probed to verify that they were scaling with respect to the 8-bit digital value assigned to the wiper's position.

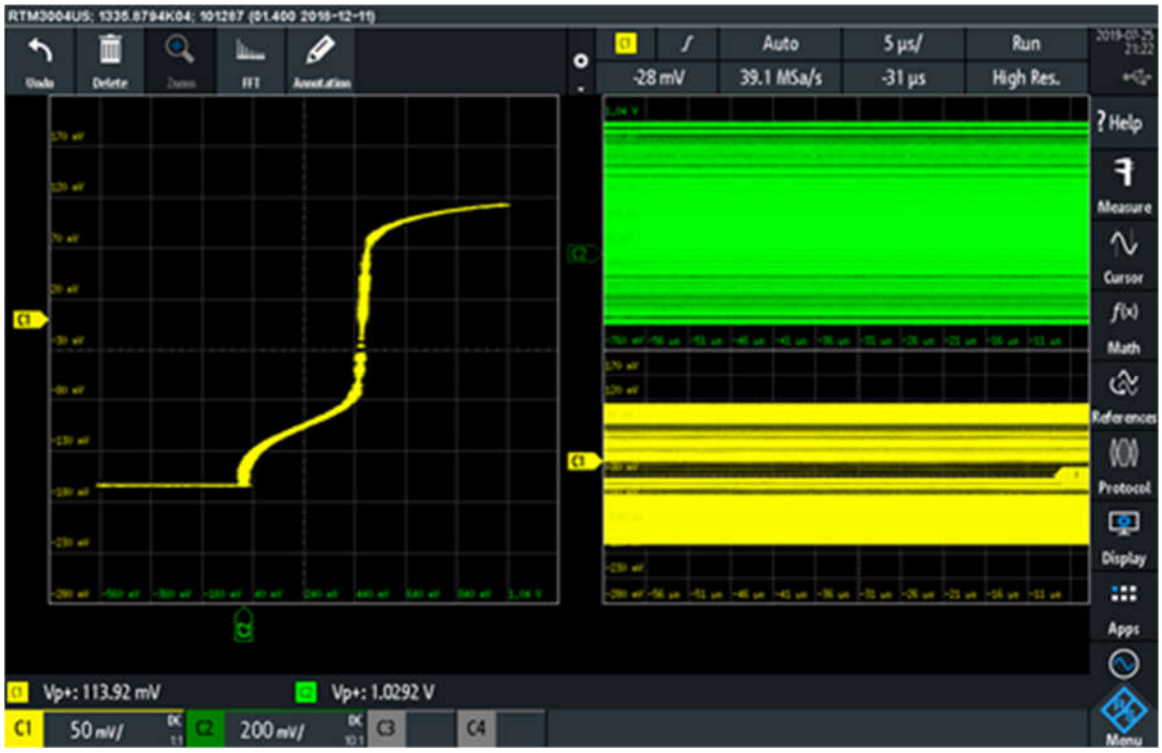


Figure 8.1: Initial Neuron Circuit Hyperbolic Tangent Response

Error! Reference source not found., again, shows the hyperbolic tangent response using the initial current-driving design. While the waveform looks approximately like the expected shape for the upper half of the trace, an unanticipated additional hump occurs below this point before the voltage abruptly runs into a hard stop. There is also a substantial offset visible, as the vertical portion of the trace should be centered, but is instead off by an entire division to the right.

Figure 8.2: Modified Neuron Circuit Hyperbolic Tangent Response

By comparison, **Error! Reference source not found.** shows the resulting hyperbolic tangent response with a digital potentiometer once the current-driving hardware had been abandoned. The difference is night and day; the vertical portion of the hyperbolic tangent is perfectly centered, with an almost perfectly symmetrical waveform on both the x and y axis of the plot. While the slope is quite steep in comparison, this can be adjusted by changing the values of the resistors used in the activation function. One thing to consider is that the above response results from tying all 4 inputs to a common input, effectively multiplying the slope of the hyperbolic tangent by 4; the effective response to a single input is somewhat less extreme, as shown in **Error! Reference source not found.**.

While the hyperbolic tangent response generated by the modified circuit is far cleaner and much more symmetrical than the original response, the slope of the response is much less ideal than the first circuit, as the overwhelming majority of the range is not used for intermediate values. With a 256-tap potentiometer, for instance, a value of 128 should be approximately 0 V of output. However, with such a steep slope, a maximum output magnitude of 1.65 V (or the equivalent value chosen for a given network) will quickly be reached within about 20 taps in either direction of 128. While this results in a very responsive network, it ultimately wastes more than 80% of the taps of the potentiometer, as the difference between 156 and 256 is effectively zero, as is the case between 0 and 100. While this may not be a significant problem in smaller networks with simple logic conditions, a large network intended for handwriting recognition or outcome prediction may fail to reach a final state any better than random guessing, as there will simply not be

enough signal fidelity between neurons to produce the necessary outputs, especially when dealing with large layer counts.

Although the response produced by a neuron should ideally be unchanged when using different operational amplifiers, there is, in reality, a surprisingly high degree of variability in neuron response behavior when changing amplifiers, even when the same components are otherwise used.

Since the TL084 was the amplifier used for the majority of prototype evaluation, it was chosen as the initial platform for hyperbolic tangent response tuning. The response of the neuron is heavily dependent upon three neurons, each of which controls a different aspect of the device's response. These resistors are the summing amplifier feedback resistor, the diode clipper resistor, and the activation function output resistor. The summing amplifier feedback resistor controls the overall slope of the hyperbolic tangent response, while the diode clipper resistor controls the slope of the response of the hyperbolic tangent when the output magnitude is less than about 50% of its maximum value. The activation function output resistor controls the maximum output voltage magnitude of the neuron.

Because the relatively steep slope of the hyperbolic tangent response was the primary cause for concern, reducing the values of the feedback and diode clipper resistors was the most obvious approach. As the activation function is reliant upon the output of the summing amplifier, it makes little sense to only adjust the diode clipper resistance, as the steep slope of the summing amplifier's response would simply overwhelm whatever reduction was effected in the activation function. Consequently, the two resistors must be adjusted in tandem to avoid excess gain on either stage. However, a single input, given all other inputs are zero, should be able to drive the neuron to a maximum or minimum output by itself; otherwise, if a single signal path is necessary to produce the proper output for the network, it may attenuate and be lost by the time it reaches the output stage.

An additional adjustment was made in the form of changing the diodes used in the neuron. Because the 1N4001 diodes used in the initial prototypes are a relatively old component, 1N4148 diodes were used in their place to observe their suitability for this application. Since the two diodes do not have identical forward bias voltages, additional tuning was necessary to maintain the proper output voltages.

Component	Initial value	Tuned value
Neuron amplifier	TL084	TL084
Clipping diodes	1N4148	1N4148
Summing feedback resistor	100 kΩ	39 kΩ
Diode clipper resistor	100 kΩ	10 kΩ
Activation output resistor	3.3 kΩ	4.3 kΩ

Table 8.1: TL084-based neuron resistance values

The initial and tuned components and resistances are given in Table 8.1. Given feed-in resistors with values of 10 kΩ, the initial summing amplifier feedback resistance of 100 kΩ corresponds to a gain of 10. Since this results in a maximum output from the summing stage with a single input at 10% of its maximum value, this is far too responsive for larger networks. Reducing the gain to 3 or 4 was found to be the best-functioning range of values, as lowering the gain beyond this point resulted in an entirely linear response from the

neuron, which results in signal attenuation from layer to layer as the output is no longer able to reach a maximum value. Similarly, the activation function amplifier functions as an inverting amplifier at low current levels, so the gain is controlled by the diode clipper resistor and activation output resistor in series. Since the feed-in resistor of the activation amplifier is 20 kΩ, the total resistance of these two resistors in series results in a gain of around two thirds; overall, this results in a gain of around 2 to 2.7, given the gain range of the summing amplifier described previously. This results in a neuron response with a much more gradual response while preserving healthy margins at the extreme ends of the potentiometer. Using 1N4148 diodes, which have lower forward bias voltages than the 1N4001, the activation output resistance must increase to approximately 4.3 kΩ to maintain the proper output magnitude. Initially, three 1N4148 diodes in series were used to produce a clipping voltage close to that of the two 1N4001 diodes in series, but this was reduced to two during tuning, resulting in a more favorable output.

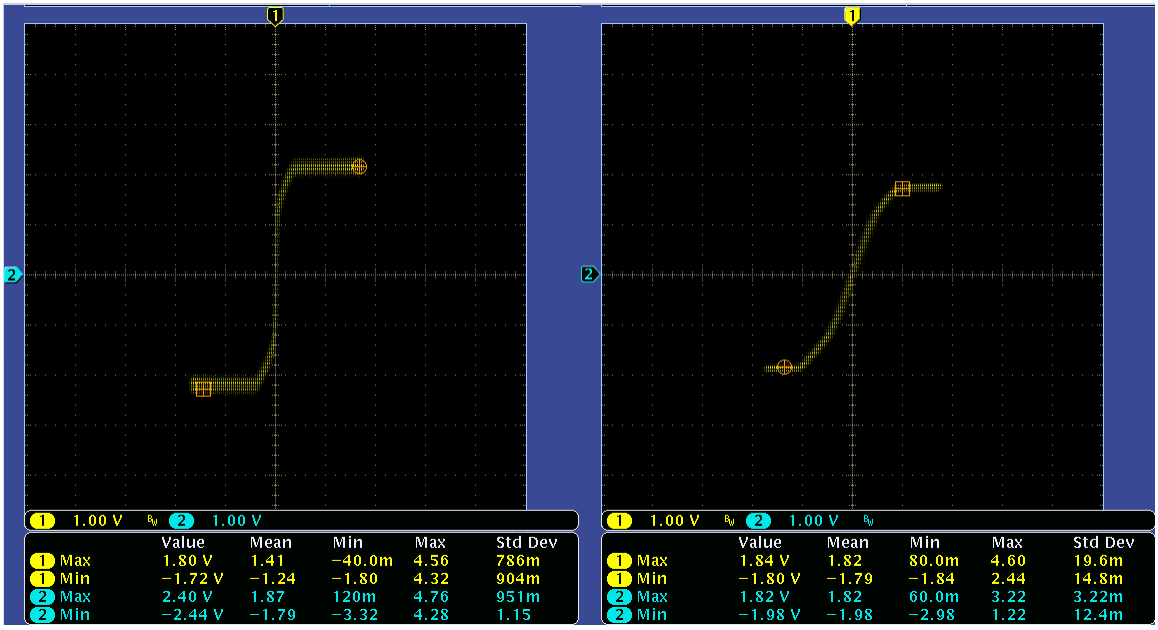


Figure 8.3: Original and tuned TL084-based hyperbolic tangent responses

Error! Reference source not found. shows the original and tuned hyperbolic tangent responses of the TL084-based neuron. The difference between the two responses is obvious; while the original function uses perhaps 25% of its range, the tuned response uses roughly 70% of its range. This effectively triples the precision of the synapse, as almost three times as many taps can be used between the two extremes of the output. Additionally, the reduction in output magnitude from a maximum overall gain of 1.33 to unity gain prevents cumulative signal distortion.

One consequence of this tuning is the introduction of an offset in the positive and negative extremes of the hyperbolic tangent; while the overall shape of the hyperbolic tangent is dramatically better, the voltage levels of the positive and negative halves of the hyperbolic tangent are no longer consistent. This is predominantly due to the inherent behavior of the TL084, as it is not a rail-to-rail amplifier and thus is not guaranteed to have equal maximum and minimum output voltage magnitudes. The TL084s tend to exhibit some play in their

maximum output magnitudes; while pushing them with much higher voltages often results in the high and low values becoming roughly identical, the lower slope of the summing amplifier and lower gain of the neuron overall result in some inconsistency between the two.

Component	Initial value	Tuned value
Neuron amplifier	MCP6274	MCP6274
Clipping diodes	1n4148	1n4148
Summing feedback resistor	100 kΩ	15 kΩ
Diode clipper resistor	100 kΩ	10 kΩ
Activation output resistor	3.3 kΩ	9.1 kΩ

Table 8.2: MCP6274-based neuron resistance values

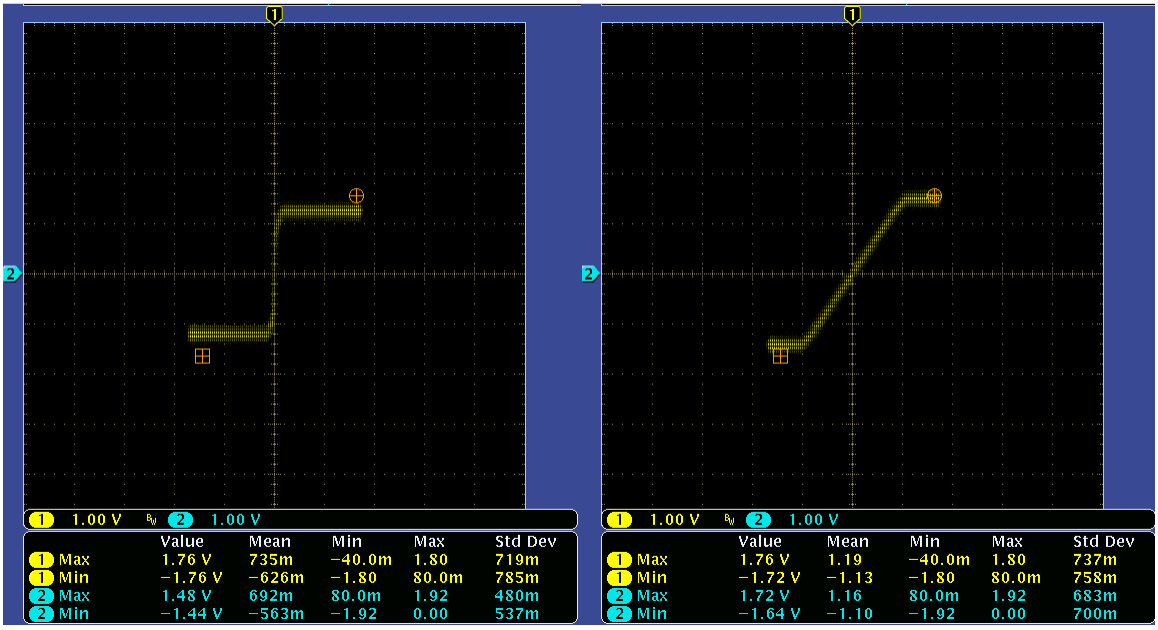


Figure 8.4: Original and tuned MCP6274-based hyperbolic tangent responses

Since rail-to-rail amplifiers offer a significant number of benefits, a neuron design based around the MCP6274 rail-to-rail amplifier was also produced. As with the TL084-based neuron, the MCP6274 neuron required a significant amount of tuning. Because the MCP6274 can operate at 3.3V and is a rail-to-rail amplifier, tuning the output voltage is not as involved a process as with the TL084. While the rail-to-rail nature of the device eliminates the voltage offset observed in the tuned TL084, it also makes it much more difficult to produce the rounded corners found in an ideal hyperbolic tangent function. The summing amplifier gain must be significantly lower, at around 1.5, and the activation function gain must be higher, at around .95, to produce a proper response, as there is no longer any overhead in the output of each stage as there was when using 5 V supply rails for the TL084. These tuning decisions are reflected in Table 8.2, while the results of these changes are provided in **Error! Reference source not found.**.

Another factor under consideration is the frequency behavior of each neuron. While there are no time constants in an ideal system, as the hardware is purely resistive, the amplifiers themselves are not capable of instantaneous response, especially when multiple stages are cascaded, as in the neurons. As such, there is an upper limit to the effective frequency that each neuron can be operated at; this frequency limit is largely dependent upon the amplifier used for the buffer and neuron hardware.

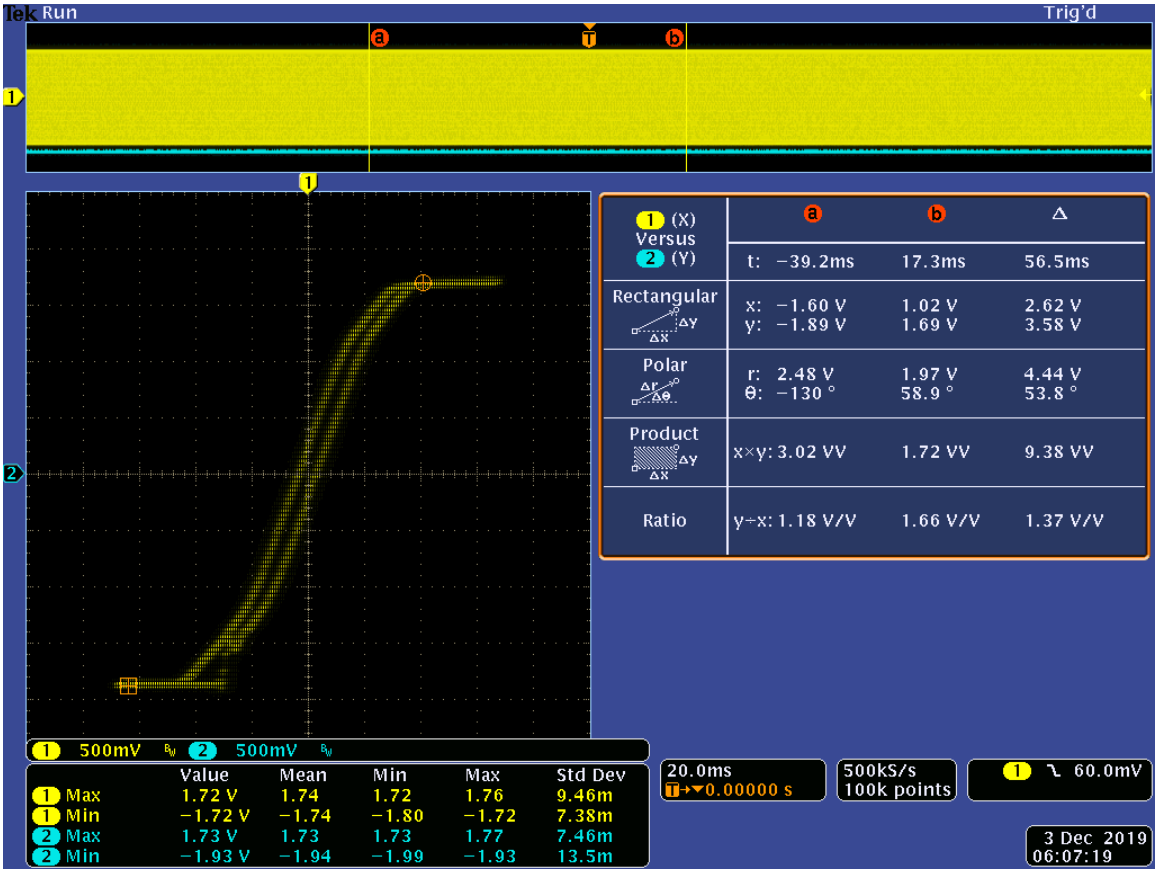


Figure 8.5: TL084-based neuron response at 20 kHz

In Figure 8.5 above, the response of a tuned neuron to a single 3.3 V_{pp} 20 kHz sinusoidal input is provided. Comparing this result with the tuned output from **Error! Reference source not found.**, it is clear that a hysteresis appears as the operating frequency of the neuron increases. Since deviation of more than 10 millivolts can result in signal corruption, it is not wise to operate the device near its upper limit. In the case of the tuned TL084-based neuron, output noise began to appear at around 8 kHz, with obvious hysteresis effects beginning around 15 kHz; consequently, limiting operation to 5 kHz allows plenty of room for error without sacrificing too much time during training operations.

Since the inputs of the network can be changed rapidly between high and low voltages, sinusoidal inputs are not necessarily the most realistic input to test frequency response behavior with. While a square wave input approximates these inputs more closely, it should not be taken as a definitive limitation on the operating frequency of the neurons. The output of each neuron must be read via ADC; as the number of neurons increases, the amount of time required to measure each neuron output also increases. While it is tempting to assume

that a smaller network can simply be run at higher frequencies than smaller networks, the rise and settling time of the neurons must be considered. Performing ADC readings too early in the cycle will result in inconsistent and variable measurements dependent upon the input combination used, while using too much margin for output settling unnecessarily slows down training. Since there are multiple stages cascaded and signal propagation takes progressively longer as more layers are added to the network, there is no simple formula for the behavior of a given network. Each network must be evaluated based upon its dimensions and amplifier technology, and basic experimentation must be performed before the maximum operating frequency can be determined.

8.1.2 Four-Neuron Prototype Testing

The four-pixel input neural network prototype consists of four inputs instead of the full twenty-five, as it is far easier to debug and modify a smaller network. There are seven fully constructed individual neuron circuits in this intermediate prototype; initially, a total of eleven neurons was constructed on breadboards, and several tests were performed on each circuit to verify that multiple neurons could be produced within the desired current and voltage bounds.

The first test measurements that were taken were the precision voltage regulator outputs. Since the digital and analog components do not operate on the same voltage levels, the ±5 V used to power the operational amplifiers must be reduced to ±1.65 V to power the digital portions of the network. After testing the regulator voltages, the voltage levels on each potentiometer and amplifier was recorded; this was necessary to ensure each component is being powered correctly, since both the amplifiers and the potentiometers are sensitive to incorrect supply voltages and a single broken connection can ruin the network.

After verifying proper voltage supplies to the network components, the behavior of the input signal buffer pairs was checked to ensure that the proper signals were being supplied to each device. Finally, using an oscilloscope, the output voltages from the wiper of each digital potentiometer were measured when supplying the maximum input to each neuron's inputs to determine whether or not the output voltages were properly scaling with the n be control signals. This process was performed by using the Teensy to sweep the potentiometers' wiper positions between 0 to 255 bits while observing the output of each potentiometer using an oscilloscope or DMM.

After potentiometer behavior was verified, the final step was to observe the functionality of the summing amplifier and hyperbolic tangent activation stages. Both the input and feedback currents of both stages were measured; these currents are significant because it is necessary to maintain currents that are both low enough to minimize power consumption and high enough to avoid aliasing due to the amplifiers' bias and offset currents. The resulting current readings from each neuron can be seen in the two following tables, Table 8.3 and Table 8.4.

A noticeable degree of variability is present among the neurons tested. While the neurons are, as a whole, fairly similar, the differences present within the set demonstrates the difficulty in creating identical neurons, even when the same values and lot of components is used in each neuron. However, since the training algorithm is quite flexible, it is able to work around reasonably low levels of variability.

Neuron #	Input Current to Summing Amplifier	Feedback Current on Summing Amplifier	Summing Amp Input Bias Current
1	-47uA to +56uA	-56uA to +50uA	0.0065uA
2	-50uA to +56.5uA	-54uA to +48uA	0.006uA
3	-50uA to +57.5uA	-55uA to +48uA	0.0055uA
4	-49uA to +58uA	-56uA to +46uA	0.006uA
5	-49uA to +51.2uA	-53uA to +49uA	0.0056uA
6	-46uA to +47uA	-50uA to +50uA	0.0045uA
7	-59uA to +52uA	-51uA to +57uA	Negligible
8	-59uA to +52uA	-50uA to +57uA	Negligible
9	-50uA to +59uA	-58uA to +49uA	Negligible
10	-54uA to +61uA	-40uA to +69uA	Negligible
11	-58uA to +52uA	-59uA to +57uA	0.008uA

Table 8.3: Summing Stage Current Measurements

Neuron #	Input Current to Activation Stage	Feedback Current on Activation Stage
1	-0.173mA to +0.204mA	-0.210mA to +0.171mA
2	-0.201mA to +-0.176mA	-0.202mA to +0.179mA
3	-0.216mA to +0.168mA	-0.217mA to +0.171mA
4	-0.216mA to +0.159mA	-0.217mA to +0.159mA
5	-0.180mA to +0.180mA	-0.180mA to + 0.179mA
6	-0.196mA to +0.182mA	-0.204mA to +0.182mA
7	-0.179mA to +0.211mA	-0.180mA to +0.203mA
8	-0.179mA to 0.211mA	-0.180mA to 0.206mA
9	-0.215mA to +0.182mA	-0.216mA to +0.183mA
10	-0.127mA to +0.273mA	-0.162mA to +0.272mA
11	-0.178mA to +0.205mA	-0.179mA to +0.204mA

Table 8.4: Activation Stage Current Measurements

After acquiring these readings along with the hyperbolic tangent response of each individual neuron via X-Y plots similar to those in Figures 8.3 and 8.4, the project was ready to progress to the construction of a slightly larger network. At this point, the next step was to construct a very basic four-to-one neuron network and make sure that the desired hyperbolic tangent responses and current readings were still being generated by each summing and activation stage's inputs and feedbacks, which was the case, as anticipated. Results of this testing can be seen in Figure 8.6 in the following section.

Following these intermediate steps, the completed four pixel input network was assembled. The four pixel input network is simply a scaled down version of the final design, and will provide insight into which design implementations work well, and which do not. Since the

number of components is relatively limited by comparison, it is much easier to adjust the components and devices used in the network, as described later in this chapter.

8.2 Software/Hardware Integration Testing

After the hardware components comprising the neural network's neurons and synapses had passed unit testing and the TACOCAT software and firmware packages had been unit tested and validated, a subsection of the four-pixel prototype network was fabricated using solderless breadboards and connected to the MCU and Raspberry Pi mini-PC hardware for integrated system testing.

The circuit, shown in Figure 8.6:, consisted of a 5-neuron network, with a first layer consisting of 4 neurons and 4 synaptic inputs per neuron, and a second layer consisting of 1 neuron with 4 synaptic inputs. The Teensy 3.5 MCU was connected to the SPI bus of the neural network's digital potentiometer array, and each neuron's non-inverted output was connected to a separate ADC input channel on the MCU.

A Raspberry Pi 3B+ mini-PC hosted the TACOCAT software package, and it was connected to the Teensy MCU via an I2C interface. An Analog Devices AD1250 digital isolator IC was used to connect the mini-PC and the MCU, which did not share a common ground level. The Raspberry Pi mini-PC was operated remotely over ethernet via an RDP connection from a laptop PC running Windows 10.

Since the breadboards suffered from noise problems and tenuous connections, especially between digital components, the size of the tested network was limited. However, this testing still produced useful feedback on the behavior of the potentiometers and the SPI signals and proved that the SPI daisy chain approach would work.

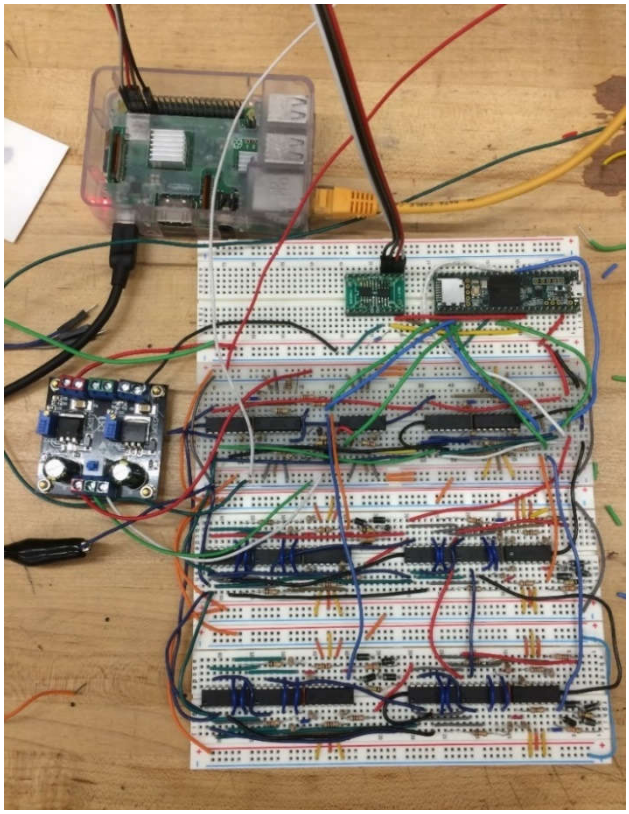


Figure 8.6: Network prototype circuit for software/hardware integration testing

The +/- 5V supply rails were powered by a laboratory bench power supply, and the primary side of a commercially available LM317/LM337-based split-rail voltage regulator unit was connected to these rails in order to power the separate +/- 1.65 V supply rails.

The objective of the integration testing procedure was to ensure that the following software/hardware interfaces and control systems worked properly:

- Software commands transmitted to the MCU over the I2C interface can be used to read and modify the state of the neural network.
- The MCU can send updated synaptic weight values to the digital potentiometers during an appropriate timeframe so that weights will be set for the next recognition task.
- The MCU's onboard ADC can read the network's neuron-output voltages and transmit them back to the software package.

A Python script was written to implement the test procedures using the existing software and firmware modules that were developed for the TACOCAT system. For this test, all of the inputs to the neural network's first layer were set to their maximum values. Neuron activation was gradually increased by sequentially incrementing each neuron's synaptic weights from 0 (full negative weight) to 255 (full positive weight), for a total of 1024 discrete activation input levels. This input sweep was applied to the first layer before being applied to the second layer.

At each one of the 1024 synaptic weights for each neuron's input sweep, an ADC sample was taken using the MCU to measure the neuron's output voltage, and the 13-bit ADC measurement value was transmitted back to the mini-PC. These data points were logged for later graphical analysis.

Initially, results were inconsistent and appeared to be incorrect. The major cause of this issue seemed to be a lack of sufficient delay after adjusting the digital potentiometer's wiper values before taking a sample reading of the neurons' output voltages. After adding a 20 µs delay between the weight-update and output-sampling operations, results became much more consistent. Plots of total synaptic weight value vs. neuron output level for each neuron in the test network are shown in Figure 8.7:.

The test results indicate that all of the hardware/software integration interfaces were operating as expected. Further testing and validation will be needed to obtain quantitative data regarding reliability and accuracy of these systems, but these preliminary results indicate that the TACOCAT system components did not have any major operational errors.

A secondary objective of integration testing was to develop a test platform that could be re-used for validation of circuit board assemblies during the next phase of the prototyping process. This validation logic may also be included in the final TACOCAT software/firmware package's initialization and self-checking diagnostic routines.

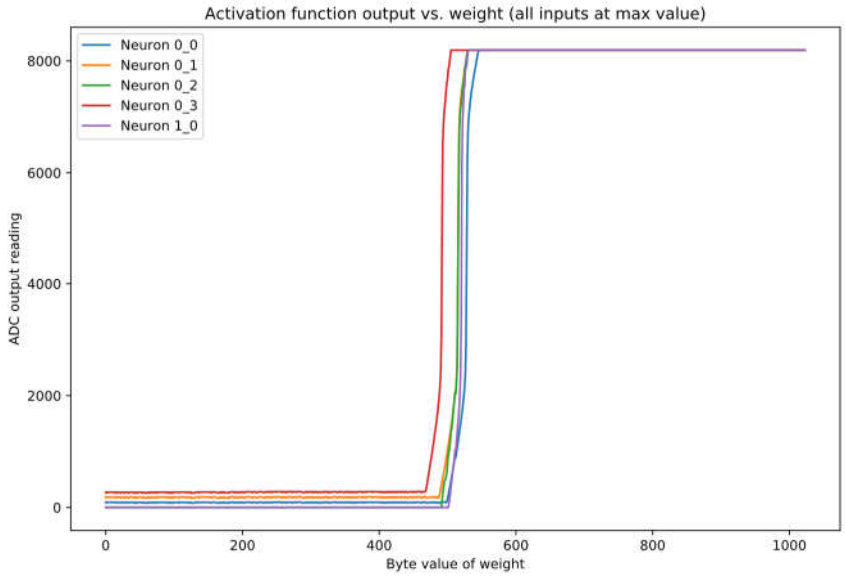


Figure 8.7: Software/hardware integration test results

8.3 Four Pixel Network Op. Amp Testing

Part of validating the design of the four-pixel network's design, is to ensure that the operational amplifier components used in the neuron and synapse circuits perform well with respect to the desired design specifications. One of which, is power consumption. To

maintain a low power consumption throughout the entire network, low rail voltages and efficient use of supply drain current must be taken into consideration. This is the main concern when performing tests of these operational amplifier components and the following tests that were performed.

Firstly, the rail voltages of the operational amplifiers that were chosen for the prototype design, the TL084 Quad op. amp. chips require a +5V to -5V minimum V_{CC} rail voltages. Other components in the neural network circuit only require a +1.65V to -1.65V rail voltages, like the digital potentiometers and the Teensy 3.5, which only needs to have 3.3V V_{CC}. To simplify the overall design of the circuit, and consume less power, an operational amplifier chip needs to be chosen with the same rail voltages, preferably one with bipolar rail to rail voltages run at +/- 1.65V. Furthermore, simplifying the supply voltages of the final neural network, allows for a reduction in complexity of the PCB layout of the 4-neuron modular PCB design. There would no longer be a need for divided voltage planes and vias for the TL084 op. amps, as the newly chosen op amps could use the same +1.65V to -1.65V rails or planes of the PCB.

Two primary operation amplifiers in the market were chosen to be used for tests in a prototype 2-neuron circuit with 4 synaptic weights each, the TL974 and MCP6274 Output Rail-To-Rail Very-Low-Noise Operational Amplifiers. The circuit components that remained from the previous prototype design include the MCP42010 digital potentiometers for the synaptic weights, and the 1N4001 rectifier diodes for the hyperbolic tangent activation function circuit of the neuron. This is to ensure that only the amplifiers are being modified in the circuit, and that a change in results will only be tied to the change in amplifiers.

The primary tests considered for the performance of each amplifier, was the hyperbolic tangent response of each neuron, as this was most significant response to test. As changing the ratio of input voltages to the summing amplifier circuit, to the output voltage of the hyperbolic tangent activation function circuit as a result from lowering the rail voltages, could alter the training time and performance of the final network's design. Aside from the performance of the hyperbolic tangent response of the neuron, the viability of using these amplifiers as buffers and inverters for the input synapse circuits needs to be tested for true voltage polarity inversion and buffering.

The first operational amplifier in contention is the TL974 operational amplifier. It offers potential operation at low rail-to-rail voltages, as low as +/-1.35V and features very low noise and low distortion. However, the functional block diagram of the data sheet reveals that it is a BJT differential input amplifiers. This would suggest a larger current draw at the input terminal of the amplifier, and a much higher temperature variance than a JFET input amplifier would have. To validate the hyperbolic tangent response of the circuit, the response was measured using the same +1.65V to -1.65V rail voltages supplied to all components, while removing the LM317 and LM337 regulator pairs that were used for the previous prototype circuit configuration. The resulting hyperbolic tangent response, at the extremes of the inputs to the summing amplifiers can be seen in Figures 8.7a and 8.7b.

As shown from the oscilloscope image of the hyperbolic tangent response, the performance of the voltage extremes in the hyperbolic tangent response of the circuit is largely degrades in resolution, and noisier. This can be explained by the fact that a much higher input bias

current in necessary to drive the inputs of this operational amplifier. Only voltage ranges in the input of the summing amplifier and the output of the hyperbolic tangent circuit can be seen in the range of -700mV to + 700mV.

To validate the performance of the input synapse circuit's drivers, the inverting and buffer op. amp. circuits, a simple circuit consisting of just the TL974 op. amp. chip, supply voltages of +/-1.65V from a DC power supply, and wires and high-power tolerance resistors were used to build the buffer and inverter circuits. A largely varying output voltage at the outputs of these circuits were observed. Finally, the input voltages at the inverting and non-inverting terminals of the operation amplifier were measured to ensure an adequate open-loop performance was obtained from these op. amps. Voltages in the range of 600mV were seen at the inverting terminal while voltages in the range of 700mV were measured in the non-inverting terminal. When taking the very large open-loop gain of an op. amp into account, this can explain the poor circuit driver performance of the synapse circuit, and ultimately prove a poor choice for the final neural network design.

Next operational amplifier considered for the final network design, the MCP6274 low noise, 0.9V/us slew rate, low rail-to-rail voltage supply, Quad JFET operational amplifier

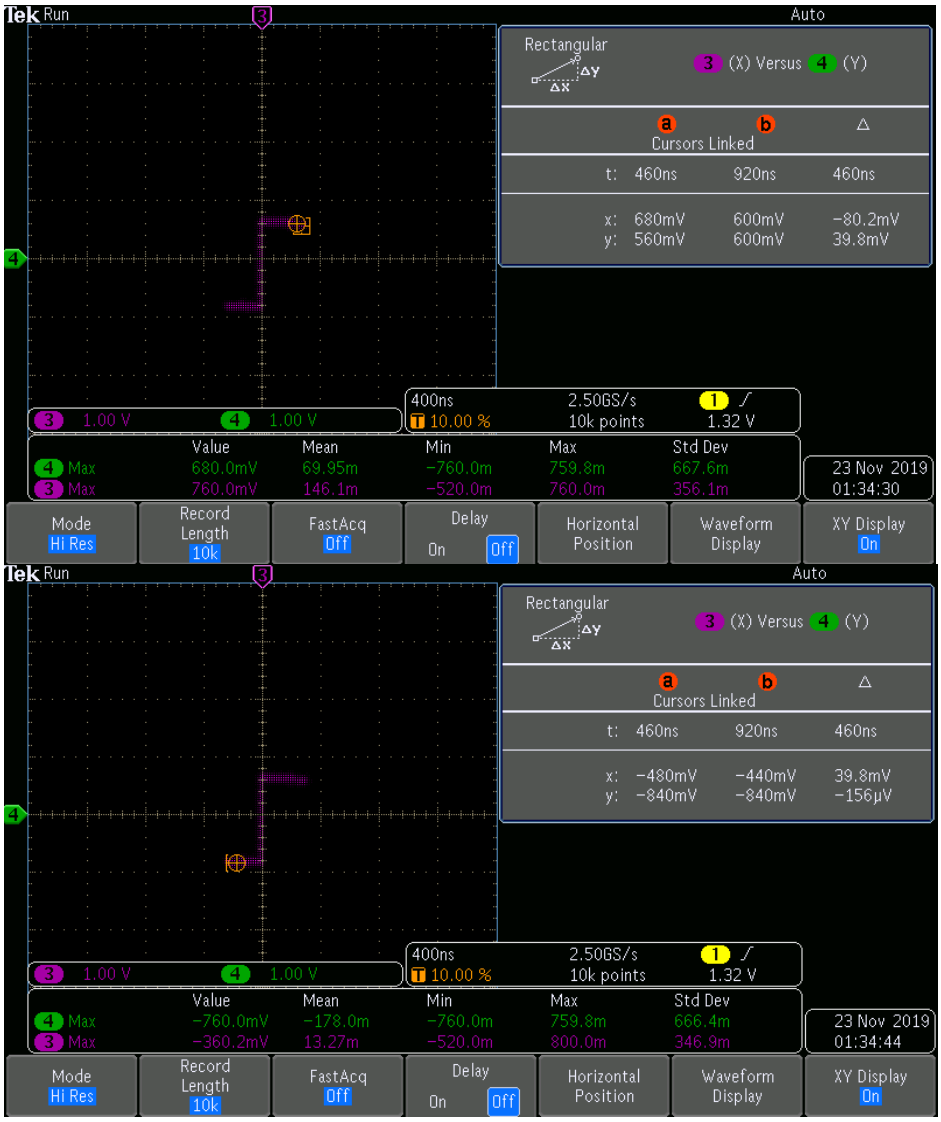


Figure 8.5: Hyperbolic tangent response using TL974 op amps shown at their maximum (a) and minimum (b) amplitudes with respect to the weighted and summed inputs

was used for similar tests. First and foremost, the most significant difference from this amplifier when compared to the TL974, is the fact that it offers a JFET differential input to the op amp. This means that a much lower power dissipation in the op amp, as the input bias current of it's internal circuit would demand a current draw from input terminals of about +/-1 pA range. When compared to the TL974, which offers an input bias current of 1000nA (when operated in +/-2.5V range according to the data sheet), the difference is obviously orders of magnitude. Even at these small current magnitudes, the large number of these op amps that would be needed for the final network design would quickly add up to a much higher power consumption.

The performance of the hyperbolic tangent response using these operational amplifiers was tested using the same testing procedure described above, for the TL974 op amp. The transfer of the input voltages to the transfer function versus the output voltages of the

hyperbolic tangent activation function circuit can be seen in the oscilloscope images taken and shown in Figure 8.8a and 8.8b, below.

An adequate +1.65V to -1.65V range of output voltages in the hyperbolic tangent response can be seen while using these MCP6274 op amps. One concerning result from this test, is the lack of vertical portion of the hyperbolic tangent response. To yield a proper hyperbolic tangent curve, further tests need to be performed when changing the ratio of resistors to provide a smaller/larger feedback in the summing amplifier circuits and the activation function circuits. However, the MCP6274 shows promising results for the final network design and will be used in following experimentation for adjusting the resistors of the summing activation function circuits of the 4-pixel network and final network design.

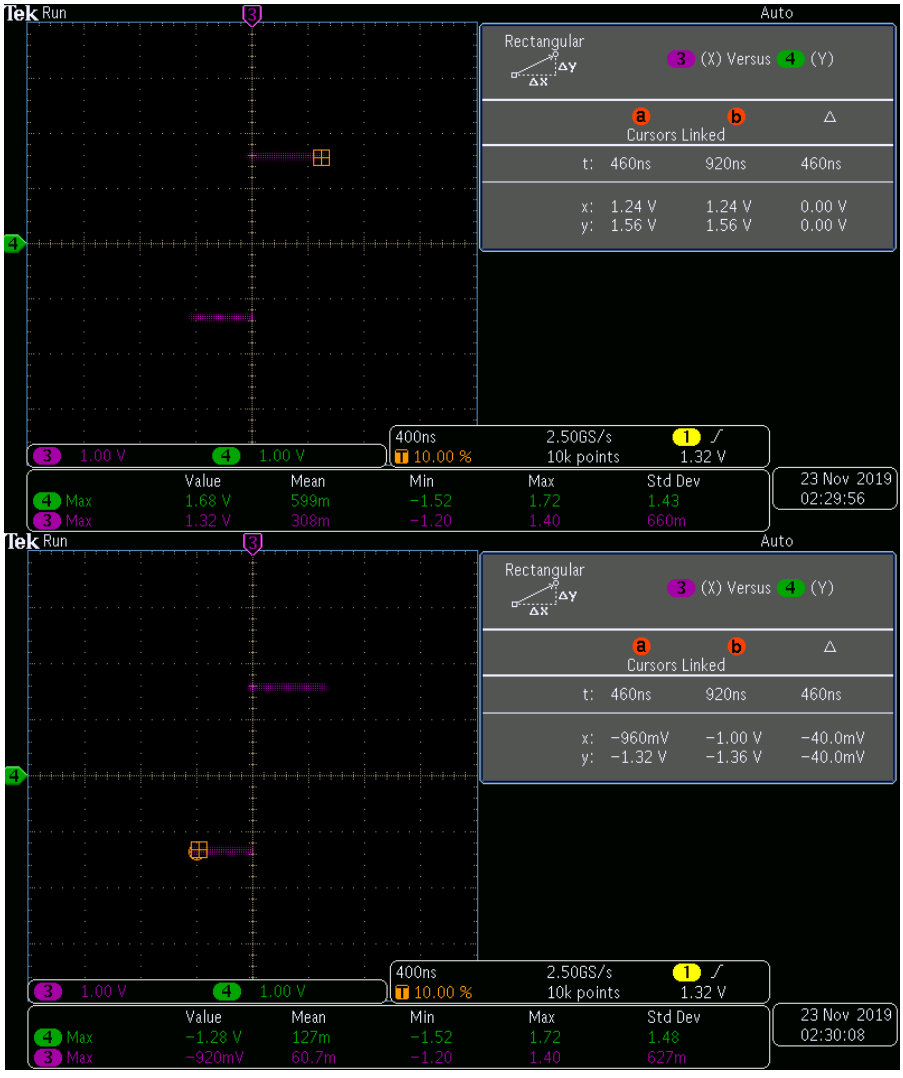


Figure 8.6 Hyperbolic Tangent Response using the MCP6274 op amps shown at their maximum (a) and minimum (b) amplitudes with respect to the weighted and summed inputs

8.4 End-to-End Testing of Four-Pixel Network Prototype

Following the unit testing of a breadboarded version of the four-pixel network hardware, the decision was made to design and order printed circuit boards for the fabrication of a working small-scale network prototype. Using a modular design approach, each circuit board represents an individual layer in the network, allowing layers to be added or removed without any additional PCB design work, provided that each layer contains no more than four neurons, each with a maximum of four synaptic inputs.

The goal for complete end-to-end testing of the four-pixel prototype network was to train the network to an accuracy rate of 100% for recognition of the binarized/black-and-white four-pixel line orientation data set. Validation was conducted by stimulating the network's input synapses and measuring the voltage at each output-layer neuron.

8.4.1 Hardware Configuration

The four-pixel network prototype, designed as described in previous chapters, was mounted semi-permanently to a wooden panel (see photograph in Figure 3.12). Two custom PCBs represented the hidden layer and output layer of the neural network. A Teensy 3.5 MCU was used for circuit-level control of the network training operations according to instructions provided by a Raspberry Pi 3 Model B+ mini-PC. The MCU and mini-PC modules shared an I2C-based data link with galvanic isolation provided by an Analog Devices ADuM1250 digital isolator IC.

The network was powered by a regulated DC bench power supply, which provided a split-rail supply at +/- 5V. An onboard combination LM317/LM337 regulator circuit was used in conjunction with additional 3.3V regulation on the Teensy MCU to create additional split-rail supply lines at +/- 1.65V.

The Raspberry Pi mini-pc was controlled using VNC-based remote access from a Windows 10 laptop via ethernet connection. The mini-pc unit's ethernet jack is galvanically isolated to eliminate any problems caused by mismatched ground levels between the two ethernet controllers.

8.4.2 Software/Firmware Configuration

The mini-pc was loaded with the Python-based neural network model and training control modules that are described in section 6.4. The MCU was programmed with the firmware package that is described in section 6.5.

8.4.3 Testing Procedure

The Python-based training algorithm was run using the binarized four-pixel line orientation data set, which consists of six samples (representing all possible 4-pixel images that contain exactly two white pixels and two black pixels). Each image can be classified according to the spatial orientation of a line connecting two pixels with the same activation state. The three image classes are labeled "Vertical", "Horizontal", and "Diagonal".

The training algorithm was configured to run for 200 training epochs. At the end of each training session, the network's trained state was validated using the full data set. Due to

the small size of the binarized four-pixel data set, cross-validation techniques were not found to be applicable.

8.4.4 Test Results

In initial runs of the training algorithm, the network failed to converge. Even after long training runs of up to 20,000 epochs, the accuracy of the network appeared to oscillate randomly. Upon analyzing the training model and comparing it to the hardware design, a discrepancy was found in the input-layer voltages that correspond to the training data's input values. The initial configuration of the prototype device was using high and low logical voltage values to represent the binary input values 1 and 0, but because the network is designed to represent positive and negative weights, the high and low logic voltages of +1.65V and -1.65V actually represent input values of +1 and -1.

In order to represent the intended binary input values of 1 and 0 as they appear in the software training model, two simple modifications were made to the prototype's hardware and firmware. First, each input-driver pin on the Teensy MCU was connected to the 0V ground supply line via individual 3.2 kΩ resistors. Second, the firmware's input-driving function was modified to place each input-driver pin in a high-impedance/floating state whenever it is set to the binary 0 value by the training software. When the pin is set to a high-impedance state, the attached resistor pulls the pin's voltage to 0V.

After making these modifications, performing some minor firmware debugging, and testing a small number of different pseudorandom initial synaptic weight sets, the prototype was successfully trained to 100% accuracy and validated using the full four-pixel binary training data set. A graph depicting changes in network accuracy during the training process is shown in Figure 8.7, and validation results showing the Teensy ADC-based voltage output readings for each input sample are provided in Table 8.5.

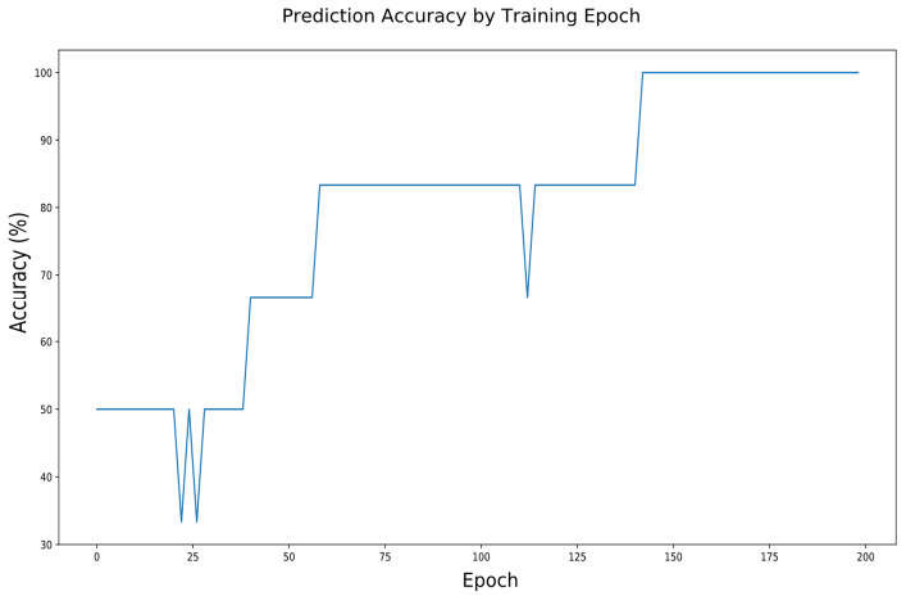


Figure 8.7: Accuracy percentage by training epoch for four-pixel network prototype.

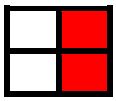
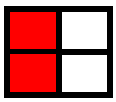
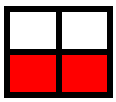
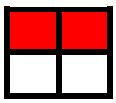
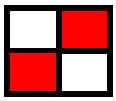
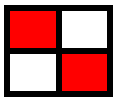
Input Sample	ADC Readings of Neuron Output Voltages (scaled from -1.0 to 1.0)		
	Vertical	Horizontal	Diagonal
	0.73068001	-0.72677329	-0.740691
	-0.26114028	-0.49578806	-0.88621658
	-1.0	1.0	-0.98559394
	-0.90282017	0.52264681	-0.93505067
	-0.85642779	-0.99511659	1.0
	-1.0	-0.89720425	0.83347577

Table 8.5: Prediction results from four-pixel prototype network test. The network's prediction is indicated by the output neuron with the highest voltage.

ADC readings from the Teensy MCU were confirmed manually by measuring output voltages with a digital multimeter. The MCU's 13-bit unsigned ADC output was scaled to a range of -1.0 to 1.0 for readability. These numbers translate to a voltage range of -1.65V to 1.65V.

8.4.5 Conclusions

Once the debugging was completed, the prototype exhibited training and prediction results that were comparable to those found in the software model. This was a positive indication that the foundational design for the final TACOCAT network was working properly. While the training software was set to run for 200 epochs, the network first reached 100% prediction accuracy in less than 150 epochs.

Most input samples showed a substantial voltage difference between the maximum output level and the next lowest voltage level, which is favorable for use with an analog comparator/LED-based visual output interface module. The smallest of these differences in the test results occurred in the second test sample shown in Table 8.5. The difference in

the ADC values of -0.26114 and -0.49579 translates to a voltage difference of approximately 37.9 mV. This voltage difference is an order of magnitude larger than the hysteresis voltage levels used in typical comparator ICs, so it seems unlikely to cause problems.

After noticing that different starting weights created distinctly different training patterns, plans were made to include an "outer loop" in the training algorithm to re-train the network using multiple sets of starting weights. By using the best results from training sessions with different starting weights, the network may be able to achieve higher levels of accuracy and prediction certainty than it would by only using the results from a single set of starting weights. While training with a single starting-weight set appeared to be sufficient for the four-pixel network, it was not known if this would be a significant limitation on the training process for the large-scale TACOCAT prototype, which performs a much more complicated recognition task.

8.5 100-Pixel Prototype Testing

After construction of the prototype was completed, testing was carried out to evaluate its performance and operating characteristics.

8.5.1 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.

Figure 8.8 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 8.6.

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

Table 8.6: Accuracy rates for different datasets trained using identical hyperparameters.

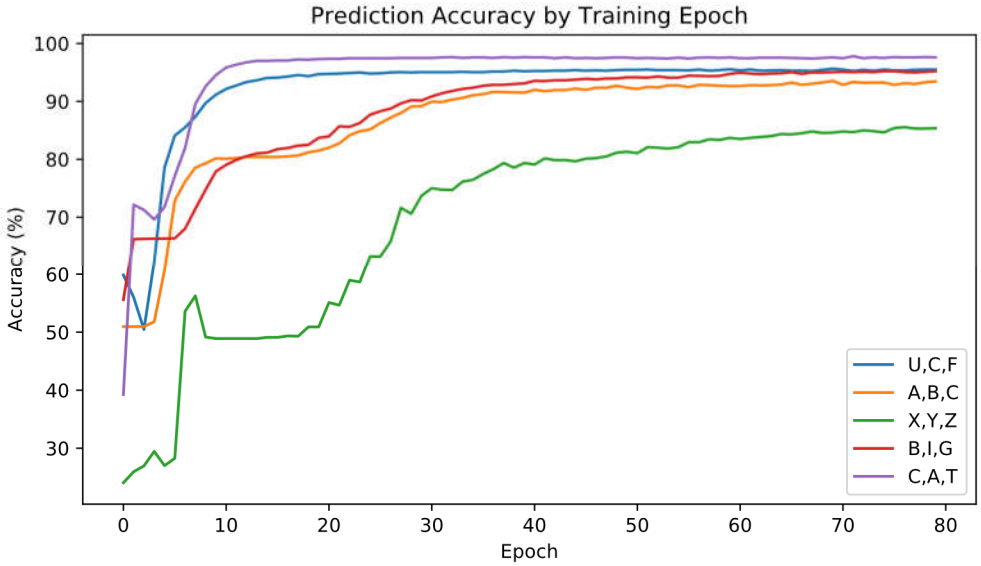


Figure 8.8: Estimated prediction accuracy by training epoch

8.5.2 User Interface

The touchscreen-based user interface was tested for its ability to receive user input via the touchscreen and get prediction results from the TACOCAT network. The design team members served as test users in order to evaluate the interface's performance.

Users were able to enter handwritten letters via the touchscreen interface, shown in Figure 8.9. The demonstration program allowed 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 selected a pre-trained network, they were able to draw their character input on the left side of the touchscreen and press the "OK" button below the input area, and the GUI showed 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 was highlighted to indicate the network's prediction. Voltmeters attached to the network's output neurons concurred with the voltage readings shown on the touchscreen.

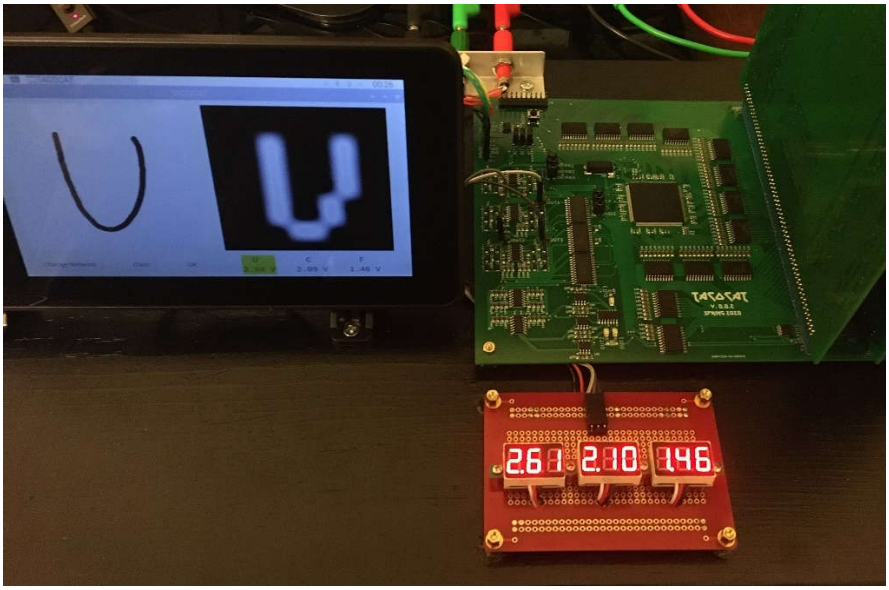


Figure 8.9: Demonstration program showing prediction result for handwritten character from touchscreen input (left of motherboard). ADC voltage readings are shown for each character on the graphical display.

8.5.3 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 8.7.

Parameter	**Requirement**	**Measured**
Output latency	< 10 ms	1.12 ms
Throughput	> 100 operations/second	857 operations/second
Accuracy	75%	94.35%

Table 8.7: Specifications compared to actual performance measurements

9 Project Operation

In this section, we will briefly describe the procedures for a user to operate the TACOCAT network.

9.1 Power

The Raspberry Pi PC should be connected to a USB power supply that is rated for at least 1.5A current draw. The TACOCAT power rails should be connected to a dual-rail power supply, rated for 400 mA current draw, using the following connections for each power jack: Red to +1.65V, Green to 0V, and Black to -1.65V.

9.2 Startup

Turn on the power supplies for the hardware network and the Raspberry Pi PC, and wait for the touchscreen display to show a Linux desktop. Then, use the touchscreen display or a USB-connected mouse to double click the "TACOCAT UI" icon on the desktop.

9.3 Getting Network Predictions

Draw an input character in the left-hand square of the user interface app, then press/click the "OK" button. The network's prediction voltages should appear on the right side of the screen below a downsampled version of the input image. The highest voltage, indicating the network's prediction, is highlighted in yellow.

9.4 Selecting Different Datasets

In the TACOCAT UI program, click the "Change Network" button to select a different pre-trained neural network. Saved network files are labeled with their creation date and the letters from the training dataset.

9.5 Shutdown

Click the "Raspberry" icon in the upper-left corner of the Linux desktop screen to show a system menu. At the bottom of the menu, click "Shut Down" and follow any prompts to shut down the Raspberry Pi. Once the Raspberry Pi has shut down, turn off the Raspberry Pi power supply. Also turn off the TACOCAT network's dual-rail power supply.

10 Administrative Content

To maintain progress and ensure the creation of a functional artificial neural network that is affordable, some administrative content needs to be considered. In this chapter, the overall project budget and timeline for completion will be discussed, as a basic project budget and schedule are essential for any major design project. The project milestones exist as a general guideline rather than a strict requirement. This project was developed over a period of approximately eight months, and the actual project performance versus the anticipated performance is provided in the following sections.

10.1 Project Budget

This project was entirely self-funded by its participants. The goal of this project was to build a 4-pixel (2x2) test network designed for basic pattern recognition, then use the results and behavior of this network to improve and up-scale the design to a 100-pixel (10x10) network for basic handwriting recognition. Because the costs of the network increase exponentially with the size of the network, a reasonable limit must be placed on the dimensions of the input image and on the width of each layer in the network. Since, as previously discussed, certain networks have more to gain from increased depth than width, it is possible to reduce the total costs of the network while simultaneously improving performance by using multiple smaller layers instead of a few wide ones. This becomes increasingly important as the number of inputs increases, as each neuron in the first hidden layer requires synapse hardware between it and the input layer. To put this into perspective, the difference between a 10-neuron layer and a 4-neuron layer for the first layer of a 25-input network is 150 potentiometers, which, at $1 per potentiometer based upon estimates in previous chapters, is a reduction in price of $150. Even if two additional 4-neuron layers are connected in series with this first layer, a net reduction of 118 potentiometers or $118 results, more than enough to offset the cost of the additional boards and neuron components despite the overall neuron count increasing relative to the single 10-neuron layer implementation.

Because the overall dimensions of the final network were dependent upon the performance of the 4-pixel test network, it was difficult to produce an error-proof budget for the project. However, by taking advantage of the cost-saving measures discussed previously, it was possible to reconfigure the network to maximize the possible performance from limited hardware if costs became prohibitive. With these considerations in mind, the project budget provided in Table 10.1 results.

Budget item	Allocated funds	Actual spending
Prototyping development boards	$50	$50
PCBs	$150	$175
Network microcontrollers	$50	$22
Potentiometers	$250	$320
Operational amplifiers	$75	$105
Buffer and neuron feedback resistors	$25	$10
Activation clipping diodes	$25	$5
Sample collection camera/pad	$100	$93
Total	$725	$780

Table 10.1: Estimated project budget item breakdown

The first set of items provided for was the group of prototyping development boards needed for the network. While the end result was total integration of the training microcontroller into a custom PCB, the complexity of the later networks was so high that this was not a practical goal to reach in one step. As such, some development boards were necessary for both the prototype and final networks.

The next budget item listed is the allocation for the project's PCBs. Since the manufacturer used for this project, JLCPCB, is able to produce modestly sized 2-layer PCBs at less than $1 per board before shipping, $150 was provided as an overestimate to cover higher-than-anticipated expenses. While this level of funding would have been sufficient for the project had all boards worked as intended, the need to revise the final motherboard and re-order the accompanying stencil resulted in some out-of-budget expenses.

The third item listed is the network microcontroller provision. While some of the final network is handled by external devices, an on-board processor was necessary to handle the network input signals. Although the processors are generally not particularly expensive, there was a reasonable degree of uncertainty as to what approach would be taken to integrating processors into the network, so the number and overall cost of the processors used was difficult to estimate. As such, a greater-than-necessary allocation was made, but only around half of this allocation was used.

The next item, the potentiometers, is by far the largest on the budget. While the potentiometers themselves are not terribly expensive, the comparisons provided in Chapter 3 produce a price point of around $.80 to $1 per potentiometer. A 100-3-3 network would require one potentiometer per connection, or 309 potentiometers, which would quickly run over budget if any problems were encountered; as a result, the maximum network size that can be achieved by this project is limited by budget. Indeed, total spending was around $320, about $70 above the upper estimate.

The operational amplifiers for this project, by contrast, have a much smaller allocation. Since the minimum number of amplifiers per synapse is one, a total of just over 300

amplifiers would be required for a 100-3-3 network, or around 80 chips. At $.50 to $.60 per chip for the TL084, for instance, this amounts to a cost of only $40 to $50 for a 100-input network, or around 15% of the price of the potentiometers. However, since substantially more expensive 3.3 V rail-to-rail MCP6274 amplifiers were used in the final network, total costs were about $30 above budget.

The next two items are the external resistors and diodes for the buffers and neuron architecture. The number of resistors used in the final network is significant, and a total of six diodes are used for each neuron. The overall price for these components is quite low, at around $.02 per resistor and $.07 to $.10 per diode, so the overall spending in these categories was quite low.

Finally, the sample collection device was one of the the least certain sections of the budget. Because sample collection could only be integrated once the complexity and design of the final network had been evaluated, it was difficult to ascertain how complicated the setup will be, which kind of device would be preferable, or what the total cost of implementation would be. As such, $100 was allocated to the sample collection device; indeed, total spending of just over $90 was required to purchase and mount a touchscreen for the network.

Overall, spending was close to the original budget. The second revision of the motherboard incurred around $90 in additional costs; without this issue, the budget would have been easily met. Outside of the budget, around $100 to $150 in additional spending for miscellaneous coursework-related items such as documentation printing and binding was made, though these costs are not directly related to the project itself.

10.2 Project Milestones

The milestones listed here were general estimates based upon progress-to-date at the end of Senior Design 1. The goals were somewhat pessimistic, as adequate room needed to be left for unanticipated complications, setbacks, or delays. The milestones were chosen with the assumption that they should, on average, be met slightly ahead of time, but that they may not necessarily be met in order. As a significant amount of research had already been performed prior to the beginning of Senior Design 1, a substantial amount of prototyping of the constituent portions of the network and testing of the individual neuron and synapse devices had been completed early on. Since the network's neurons and synapses had already been successfully tested both individually and in combination at the beginning of Senior Design 1, the next steps were to construct a basic 4-input network and attempt to train it to perform basic pattern identification using the prototyping and component testing data. With the test network completed and trained, a significant hurdle in the project was passed, as increasing the size of the network once a functioning network has been produced is predominantly a logistical problem rather than a technical one. With the test network as a starting point, the ensuing goals were to streamline the training algorithm and correct any chronic identification errors, potentially build a larger intermediate network to scale up and test the efficacy of troubleshooting, and design the final network's PCB and peripheral hardware. Ultimately, a final network of 100 inputs and 3 outputs was chosen, and a

potential intermediate network was skipped in favor of dedicating additional attention to the final network's design. A schedule of milestones throughout both semesters of this project is provided in Table 10.2 and Table 10.3.

Date	Goal	Comments	Status
September 30	4-pixel test network built on breadboards		Met September 7
October 15	Complete initial training algorithm for 4-pixel test network; integrate microcontroller into network to prepare for training	2 weeks to modify training algorithm to be compatible with physical network	Milestone skipped in favor of PCB implementation
October 31	4-pixel network trained and evaluated; 60-page draft completed	2 weeks for troubleshooting and adjusting algorithm	Milestone skipped in favor of PCB implementation
November 15	Finished refining algorithm; potentially construct intermediate network if necessary; 100-page draft completed	Additional 2 weeks for algorithm streamlining or ordering parts if necessary	Met November 15; 4-pixel PCB-based network substituted for intermediate network
November 21	Intermediate network trained and evaluated	May not be performed depending on 4-pixel network performance	Met November 19; merged with 4-pixel network milestones
December 2	Finalize part list that will be used for first iteration of final design; final documentation completed and submitted	Parts list dependent upon performance of prototype networks	Met November 26

Table 10.2: Senior Design I milestones

It should be noted that the milestones for the 4-pixel and intermediate network were adjusted or skipped based upon the results of breadboard testing. The inconsistent and error-prone nature of breadboarding resulted in intolerable difficulties in network evaluation, so the 4-pixel breadboard network and intermediate PCB network goals were merged to allow for meaningful data to be recorded. Consequently, the intermediate network in these milestones is effectively the PCB implementation of the 4-pixel network, while the original 4-pixel network is the original breadboard implementation.

Date	Goal	Comments	Status
January 15	Network hardware component of PCB design complete (neurons, synapses)	6 weeks including winter break to collect footprints and design PCB network layout	Met January 28
January 21	Power, microcontroller, communication components of PCB finished	Additional week to finish layout of peripherals and power	Met February 11
February 7	First iteration PCB ordered and populated	1-2 weeks to obtain and build	Met February 29
February 14	First iteration training completed	1 week due to streamlining from earlier networks	Not met due to design flaws
February 21	PCB updated and re-ordered if necessary	Additional week to troubleshoot and update design	Met March 13
March 7	Corrected PCB obtained, built, and retrained	2 weeks to obtain PCB/parts and retrain/troubleshoot	Met March 26
March 21	Finalized training and hardware		Met March 29
March 31	Complete integration of sample reader	Dependent upon overall progress and chosen method of reading new samples	Met March 16
April 15	Network organized and completed	2 weeks to neaten up setup and smooth any remaining issues	Met April 12

Table 10.3: Senior Design II milestones

Overall, the major milestones for this project were met roughly on schedule. The only milestones that were consistently met later than expected were related to the PCB design, assembly, and initial training of the final network, as several rounds of adjustment and partial redesign were involved in the first round of boards, resulting in about a month of delays. The inclusion of an additional month of breathing room for a potential redesign of the PCBs proved wise, as several design flaws in the motherboard necessitated an additional revision. Despite these delays, however, the final milestones were met roughly on time.

11 Project Summary and Conclusions

In this project, we sought to build a hardware-based artificial neural network using commonly-available components. After reviewing literature, running system-level simulations, and testing individual circuits, we moved on to constructing a small-scale prototype network designed to classify four-pixel images.

After successfully testing the small-scale prototype, we began working to scale the design up to a full-sized prototype for the classification of 100-pixel images. We constructed a design based on one motherboard with three attached daughterboards, which was controlled using a 32-bit microcontroller that communicates with a Linux PC. 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.

12 Appendices

12.1 Appendix A: Copyright Permissions

Image Reproduction Permission Request

Hi,
Thank you for contacting us. Please feel free to use the image.

...

Thank you! | Thank you so much! | Great, thank you so much!

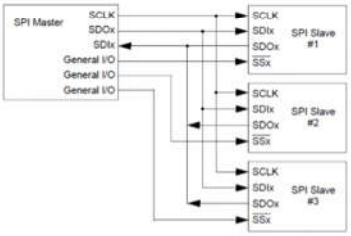
Are the suggestions above helpful? Yes No

German Romero Castro
Fri 11/15/2019 9:03 AM
mail@electrosome.com

Good Morning,

I am a student at the University of Central Florida, and I am a part of a team of students who are building a hardware neural network as a Multi-Layer Perceptron for our Senior Design project. We would like to use some of your images for documentation. Our report will be published on the UCF website upon its completion (http://eecs.ucf.edu/seniordesign/). I was hoping you might be able to assist me in obtaining permission for reproduction.

We are hoping to obtain permission for the following images:
https://electrosome.com/wp-content/uploads/2017/04/SPI-Master-and-Multi-Slave-Connections-600x404.png



Please let me know if this is possible. Feel free to direct me to someone more suitable if you are not the proper point of contact.

Thanks in advance for your help,

Image Reproduction Permission Request

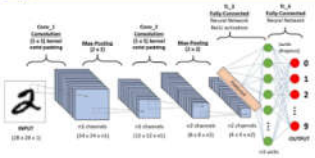
German Romero Castro
Thu 10/31/2019 9:42 AM
legal@medium.com

Good Morning,

I am a student at the University of Central Florida, and I am a part of a team of students who are building a hardware neural network as a Multi-layer Perceptron for our Senior Design project. We would like to use some of your images for documentation. Our report will be published on the UCF website upon its completion (http://eecs.ucf.edu/seniordesign/). I was hoping you might be able to assist me in obtaining permission for reproduction.

We are hoping to obtain permission for the following image:

https://miro.medium.com/max/1644/1*uAeANQIOQPqWZnnuH-VEyw.jpeg



Please let me know if this is possible. Feel free to direct me to someone more suitable if you are not the proper point of contact.

Thanks in advance for your help,

German Romero Castro
Candidate for B.S.of Electrical Engineering
University of Central Florida (Spring '20)

mobile: 321.333.0618
email: german.romero@ieee.org

View my LinkedIn Profile

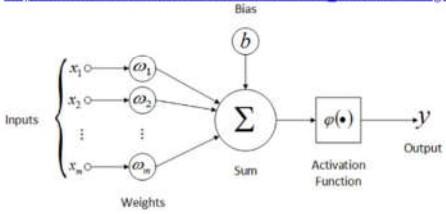
Reproduction Permission Request

German Romero Castro
Thu 10/31/2019 10:23 AM
legal@medium.com

Good Morning,

I am a student at the University of Central Florida, and I am a part of a team of students who are building a hardware neural network as a Multi-Layer Perceptron for our Senior Design project. We would like to use some of your images for documentation. Our report will be published on the UCF website upon its completion (http://eecs.ucf.edu/seniordesign/). I was hoping you might be able to assist me in obtaining permission for reproduction.

We are hoping to obtain permission for the following image:
https://miro.medium.com/max/1500/1*WRG_Re8vGVuHDYigtq2IBA.jpeg



Please let me know if this is possible. Feel free to direct me to someone more suitable if you are not the proper point of contact.

Thanks in advance for your help,

German Romero Castro
Candidate for B.S of Electrical Engineering
University of Central Florida (Spring '20)

mobile: 321.333.0618
email: german.romero@ieee.org

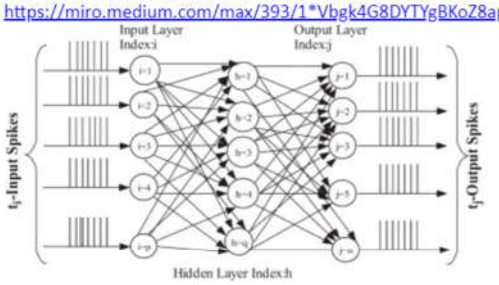
Reproduction Permission Request

German Romero Castro
Thu 10/31/2019 10:18 AM
legal@medium.com

Good Morning,

I am a student at the University of Central Florida, and I am a part of a team of students who are building a hardware neural network as a Multi-Layer Perceptron for our Senior Design project. We would like to use some of your images for documentation. Our report will be published on the UCF website upon its completion (http://eecs.ucf.edu/seniordesign/). I was hoping you might be able to assist me in obtaining permission for reproduction.

We are hoping to obtain permission for the following image:
https://miro.medium.com/max/393/1*Vbgk4G8DYTYgBKoZ8ap2NQ.jpeg

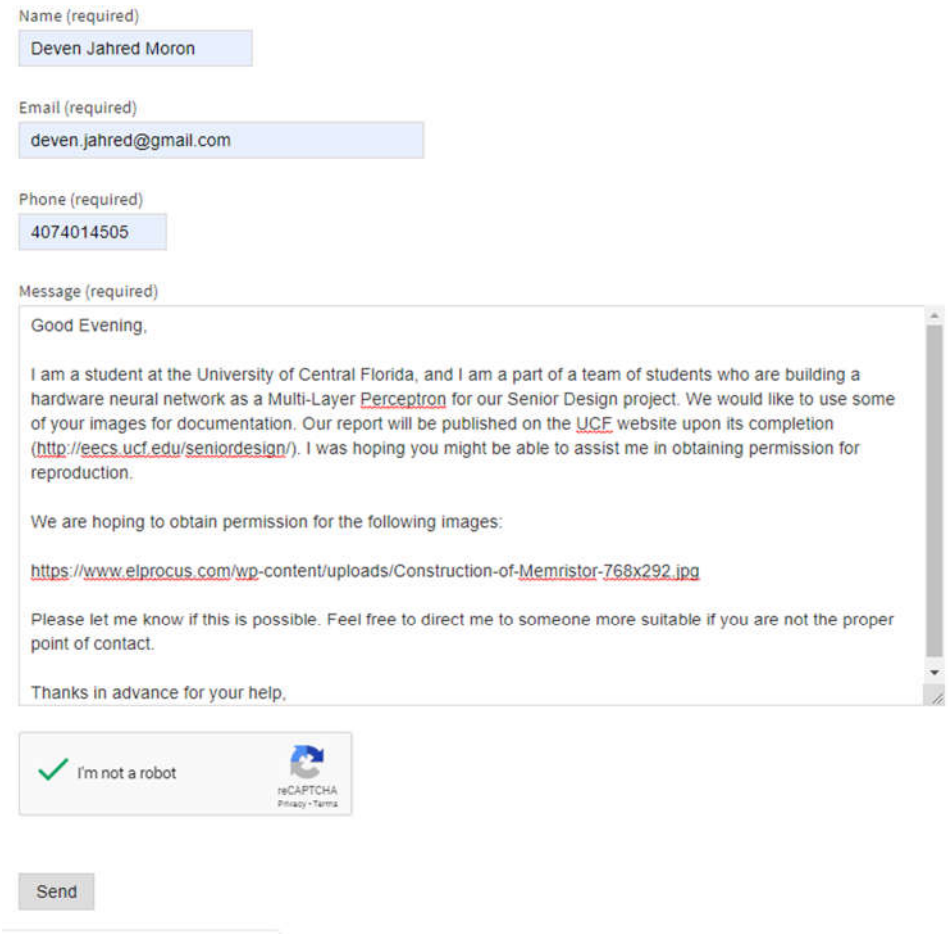


Please let me know if this is possible. Feel free to direct me to someone more suitable if you are not the proper point of contact.

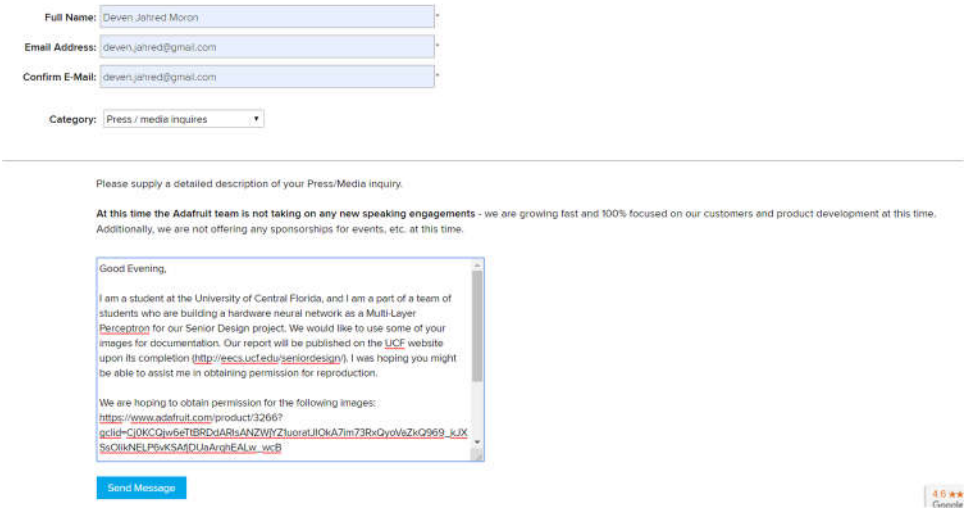
Thanks in advance for your help,

German Romero Castro
Candidate for B.S.of Electrical Engineering
University of Central Florida (Spring '20)

mobile: 321.333.0618
email: german.romero@ieee.org



Permission Request for titanium dioxide memristor diagram from https://www.elprocus.com/



Permission Request for Teensy 3.6 Development Board image from https://www.adafruit.com/

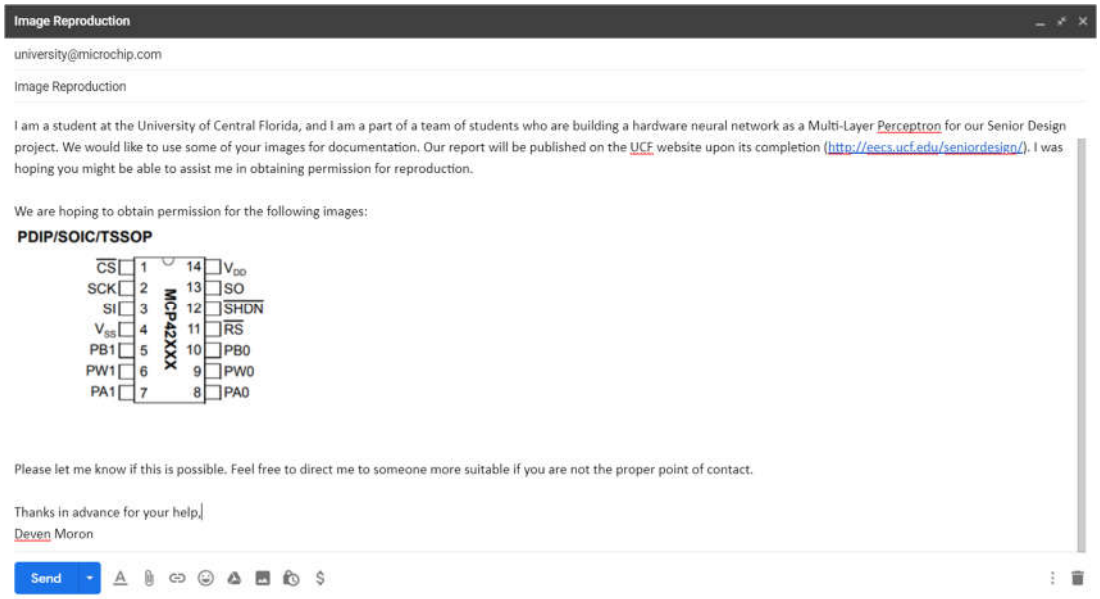


Image Reproduction

university@microchip.com

Image Reproduction

I am a student at the University of Central Florida, and I am a part of a team of students who are building a hardware neural network as a Multi-Layer Perceptron for our Senior Design project. We would like to use some of your images for documentation. Our report will be published on the UCF website upon its completion (http://eecs.ucf.edu/seniordesign/). I was hoping you might be able to assist me in obtaining permission for reproduction.

We are hoping to obtain permission for the following images:

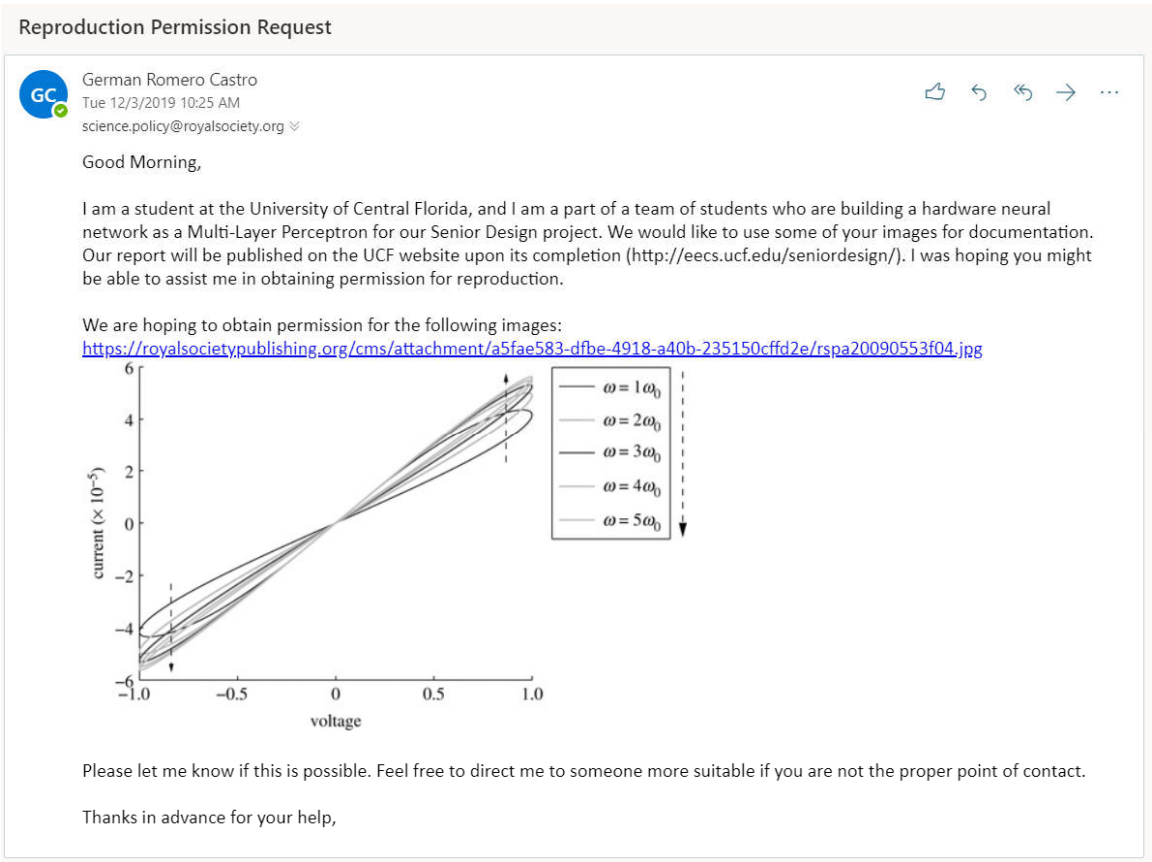
PDIP/SOIC/TSSOP

Please let me know if this is possible. Feel free to direct me to someone more suitable if you are not the proper point of contact.

Thanks in advance for your help,
Deven Moron

Send

Permission Request for MCP42XXX digital potentiometer pin out image from https://www.microchip.com/



Reproduction Permission Request

GC

German Romero Castro
Tue 12/3/2019 10:25 AM
science.policy@royalsociety.org

Good Morning,

I am a student at the University of Central Florida, and I am a part of a team of students who are building a hardware neural network as a Multi-Layer Perceptron for our Senior Design project. We would like to use some of your images for documentation. Our report will be published on the UCF website upon its completion (http://eecs.ucf.edu/seniordesign/). I was hoping you might be able to assist me in obtaining permission for reproduction.

We are hoping to obtain permission for the following images:
https://royalsocietypublishing.org/cms/attachment/a5fae583-dfbe-4918-a40b-235150cffd2e/rspa20090553f04.jpg

Please let me know if this is possible. Feel free to direct me to someone more suitable if you are not the proper point of contact.

Thanks in advance for your help,

12.2 Appendix B: Component Information

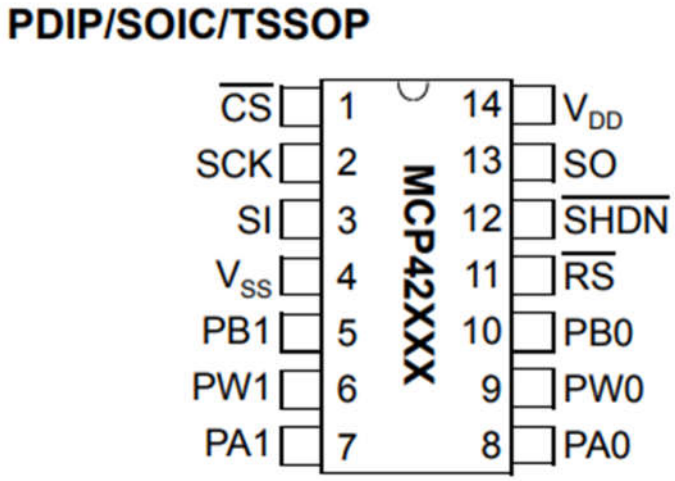


Figure A1.1: MCP42010 Digital Potentiometer Pin Out

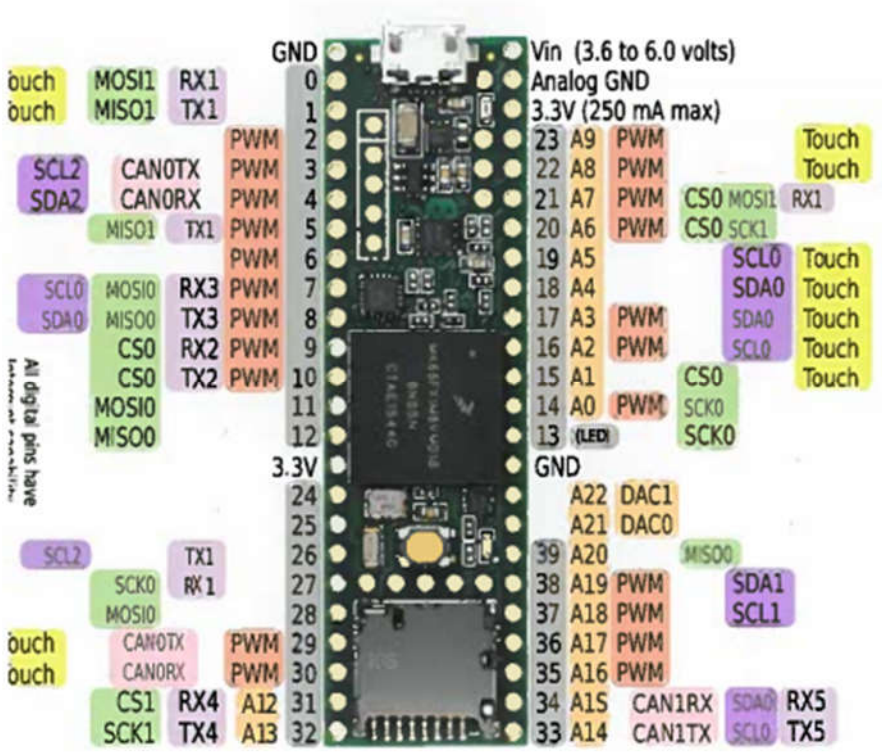


Figure A1.2: Teensy 3.5 Board Pin Out

12.3 Appendix C: References

Portions of this paper were also included in the authors' unpublished conference paper, "TACOCAT: Trainable Acceleration of Classification Operations via Commonly Available Technology".

Cohen, G., Afshar, S., Tapson, J., & Van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. Proceedings of the International Joint Conference on Neural Networks, 2017-May, 2921–2926. https://doi.org/10.1109/IJCNN.2017.7966217

LeCun, Y., Bottou, L., Bengio, Y., & P. Haffner. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Mikhaylov, Alexey & Morozov, O. & Ovchinnikov, P. & Antonov, I. & Belov, A. & Korolev, Dmitry & Koryazhkina, M. & Sharapov, Alexander & Gryaznov, Evgeny & Gorshkov, O.N. & Kazantsev, Victor. (2017). Towards Hardware Implementation of Double-Layer Perceptron Based on Metal-Oxide Memristive Nanostructures.

Nicholson, Chris. "Spiking Neural Networks." *Skymind*, 2019, skymind.ai/wiki/spiking-neural-network-snn.

Pfeiffer, Michael, and Thomas Pfeil. "Going Deeper in Spiking Neural Networks: VGG and Residual Architectures." *Frontiers*, Frontiers, 25 Jan. 2019, www.frontiersin.org/articles/10.3389/fnins.2019.00095/full.

Rajput, Abhishek. "Introduction to ANN: Set 4 (Network Architectures)." *GeeksforGeeks*, 17 July 2018, www.geeksforgeeks.org/introduction-to-ann-set-4-network-architectures/.

Raschka, S. (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. Journal of Open Source Software, 3(24), 638. https://doi.org/10.21105/joss.00638.

Soni, Devin. "Spiking Neural Networks, the Next Generation of Machine Learning." *Medium*, Towards Data Science, 16 July 2019, towardsdatascience.com/spiking-neural-networks-the-next-generation-of-machine-learning-84e167f4eb2b.

Stewart, Matthew. "Intermediate Topics in Neural Networks." *Medium*, Towards Data Science, 27 June 2019, towardsdatascience.com/comprehensive-introduction-to-neural-network-architecture-c08c6d8e5d98.

Tisan, Alin & Oniga, Stefan & Gavrincea, Ciprian, "Hardware implementation of a MLP network with on-chip learning", (2006), 5th WSEAS Int. Conf. on Data Networks, Communications & Computers.

Zyarah, A. & Ramesh, A. & Merkel, C. & Kudithipudi, D., "Optimized hardware framework of MLP with random hidden layers for classification applications," Proc. SPIE 9850, Machine Intelligence and Bio-inspired Computation: Theory and Applications X, 985007 (12 May 2016); https://doi.org/10.1117/12.2225498