Modelling and Analysis of Dendritic Tree Function in Cortical Pyramidal Neurons

Due: April 10, 2022

Professor Gunnar Blohm *Professor Yanglei Song*

Faculty of Engineering and Applied Science
Department of Mathematics and Engineering
Queen's University
Kingston, Ontario

Amean Asad, 20043851
Kai Ferrall, 20065567
Caroline Kim, 20091781
Paul Majewski, 20071739
Eric Venditti, 20070722
Jade Watson, 20052115
Julia Wiercigroch, 20055537

1 Acknowledgements

The team would like to thank Dr. Gunnar Blohm and Dr. Yanglei Song for sharing their knowledge, support, time and guidance throughout the course of the project.

The team would like to thank David Beniaguev, Idan Segev, and Michael London for providing the team access and utilization of data set used in designing and testing of the solution.

Lastly, the team would like to thank Fluer Zeldenrust, Sicco de Knecht, Wytse J. Wadman, Sophie Deneve, and Boris Gutkin for their knowledge on estimating the information extracted by a single spiking neuron from a continuous input time series.

2 Abstract

Biologically-inspired artificial neurons tend to oversimplify dendritic tree contributions in computational neuroscience models of the brain. A temporal convolutional neural network (TCN) with 7 layers was shown to model the nonlinear processing of dendritic trees and predict somatic voltage with 99% accuracy. However, artificial neurons with many parameters may be computationally expensive to implement in a large-scale computational model of the brain. The team explored different neural network architectures to model the nonlinear processing of dendritic trees in L5 cortical pyramidal neurons including fully connected networks (FCNs), convolutional neural networks (CNNs), and TCNs in PyTorch. The team also used information theory to develop a way to quantify information transfer for the dendrite models by computing the entropy transfer from inputs to outputs. TCNs performed the best out of the architectures tested, achieving 43.8% accuracy on the test data. In FCNs, increasing number of hidden layers or parameters decreased model performance whereas CNNs performed similarly to linear models. Researching models for a computationally-efficient artificial neuron that captures biological computations may help scientists understand brain function and may lead to advancements in applications for rehabilitation for neuromuscular disease.

Contents

1 Acknowledgements **ii**

2 Abstract **iii**

3 Introduction **2**
3.1 Liquid State Machines . . . 2
3.2 Computational Neuron Models . . . 2
3.2.1 McCulloch and Pitts . . . 3
3.2.2 Leaky Integrate and Fire Neuron Model . . . 3
3.2.3 Izhikevich's Model . . . 3
3.3 Limitations of Implemented Models . . . 4
3.4 Evaluation of Dendritic Tree Modeling with Deep Artificial Neural Networks . . . 4

4 Problem **4**
4.1 Problem Description . . . 4
4.2 Problem Statement . . . 5
4.3 Stakeholders . . . 5

5 Design Process **6**
5.1 Overview and Assumptions . . . 6
5.2 Data Set Analysis and Loading . . . 7
5.3 Model Development and Training . . . 8
5.4 Design of Information Theoretic Metric . . . 11

6 Results **13**
6.1 Model Accuracy Results . . . 13
6.1.1 McCulloch-Pitts Model . . . 14
6.1.2 Beniaguev's Models . . . 14
6.1.3 Fully-Connected Neural Networks . . . 14
6.1.4 Convolutional Neural Networks . . . 15
6.1.5 Temporal Convolutional Neural Networks . . . 16
6.2 Information Metric Results . . . 17

7 Discussion **18**
7.1 Limitations and Trade-offs . . . 20
7.2 Next Steps . . . 20

8 Engineering Impacts **21**
8.1 Application . . . 21
8.2 Triple Bottom Line . . . 21
8.3 Economic Analysis . . . 21
8.4 Standards, Codes, and Ethical Considerations . . . 22

9 Conclusion **22**

3 Introduction

3.1 Liquid State Machines

The brain is capable of solving extremely complex problems, however scientists have barely scratched the surface of understanding how neuron communication and brain circuits are involved in generating everyday perceptual experiences and actions. Neuroscientists have turned to computational approaches to model neural mechanisms through Liquid State Machines (LSM) comprised of recurrent and random spiking neural networks (SNN) [1]. Unlike other neural networks, the building blocks of LSM are biologically-inspired artificial neurons. Scientists hope that by using a model derived from foundational neuroscience that they will be able to gain insights into the computational principles explaining brain function and dysfunction.

3.2 Computational Neuron Models

LSM are made up of three layers including an input layer, a reservoir or liquid and a memoryless readout circuit [2]. Both the reservoir and the output circuit layers contain artificial neurons that are modeled to replicate the input-output relationship of biological neurons.

Neurons in the brain are constantly bombarded by electrochemical signals from surrounding neurons, resulting in altered membrane potentials in their dendrites (Figure 1a). These altered voltages propagate to the soma where they are non-linearly combined into one somatic voltage. Depending on the function and location of the neuron, if the somatic voltage is above a specific voltage threshold, the neuron triggers an action potential and releases chemicals at its axons (Figure 1a). Neuronal communication is often characterized by the spikes in somatic voltage as shown in Figure 1b. Since each neuron is predicted to have thousands of connections to other neurons, researchers refer to the inputs and outputs as spike trains.

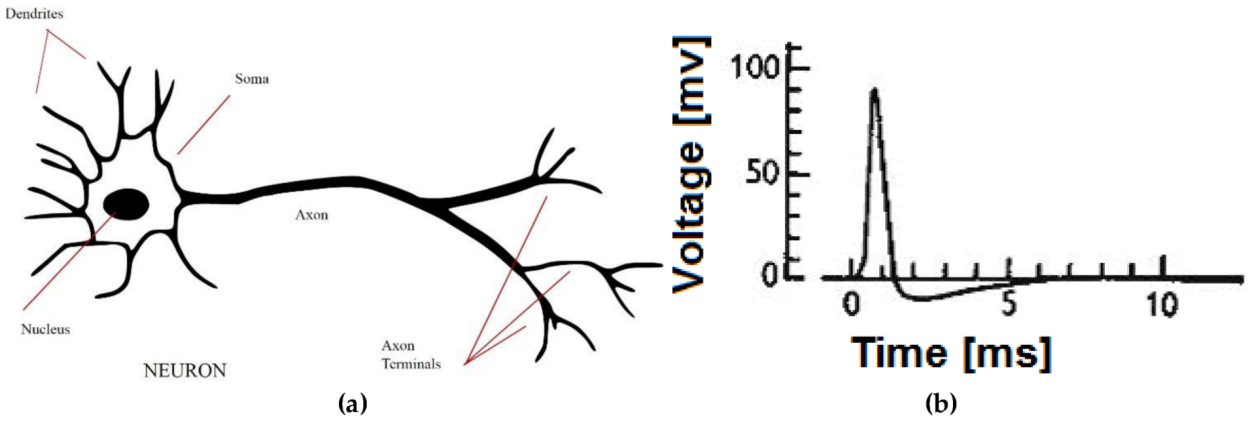


Figure 1: Neuron Anatomy and Membrane Potential

Developing an accurate but computationally-efficient model to be used in LSM is highly sought after in computational neuroscience. Currently, the three most commonly used computational models of neuron include:

- McCulloch and Pitts - Linear Threshold Units [3]
- Leaky Integrate and Fire Model [4]
- Izhikevich Neuron Model [5]

3.2.1 McCulloch and Pitts

The McCulloch and Pitts model is a linear model that takes the weighted sum of the dendritic voltages to predict the somatic voltage (Figure 2). The model assumes that all the dendritic inputs converge linearly to a single compartment and have a predetermined positive synaptic weight [6]. If the sum exceeds the threshold potential, a spike is generated.

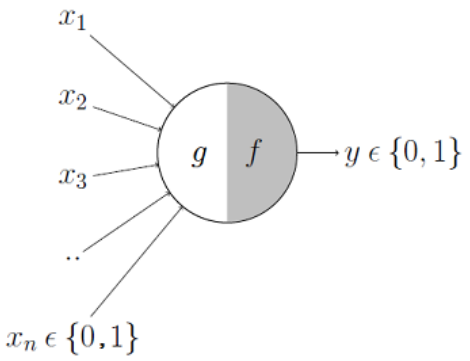


Figure 2: McCulloch and Pitts Computational Model

3.2.2 Leaky Integrate and Fire Neuron Model

This model uses a linear differential equation to derive the state of the neuron by comparing it to the voltage across a capacitor in parallel with an ohmic resistor as shown in Figure 3 [4]. The voltage trajectory executes a random walk that depends on the nature of the synaptic input. Once a fixed voltage threshold is reached, a unit pulse is generated, the voltage is reset, and the charge is removed from the capacitance.

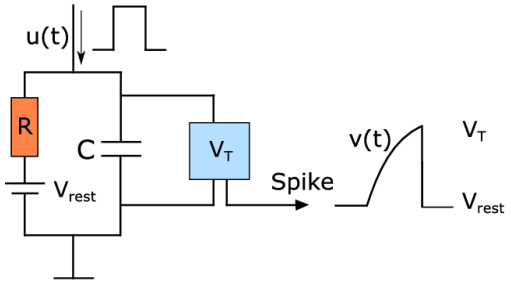


Figure 3: Leaky Integrate and Fire Computational Model

3.2.3 Izhikevich's Model

Izhikevich's model is computationally simple yet capable of producing accurate spiking dynamics of a neuron. It is a two dimensional system of differential equation as shown below [5].

$$\dot{v} = 0.04v^2 + 5v + 140 - u + I \tag{1}$$

$$\dot{u} = a(bv - u) \tag{2}$$

where v represents the membrane potential and u represents a membrane recovery variable and provides feedback to v.

3.3 Limitations of Implemented Models

The models currently used in LSM do not accurately account for the nonlinear behaviour of synaptic signals or their processing along dendritic trees. Some of the limitations of the models include :

- Synaptic inputs are not constant current sources that linearly increase or decrease the RMP of the neuron [6].
- Synaptic inputs briefly change the electrical properties of the postsynaptic membrane [7].
- Dendritic trees can amplify or shunt synaptic input signals before they reach the soma [8].
- The distance of the synapses on the dendrites to the soma can lead to more/less time for dendritic modifications [8].

3.4 Evaluation of Dendritic Tree Modeling with Deep Artificial Neural Networks

In an attempt to model the non-linear modifications performed by dendritic branches, Beniaguev *et. al.* trained temporal convolutional neural networks [9] with several different depths to observe if the output somatic voltage and spike trains could accurately be predicted from input spike trains. The researcher was able to predict the correct output with 99.71% accuracy [9]. However, the best performing model had 7 layers and 128 channels per layer [9]. Although this network is able to accurately predict the output for one neuron, the size of the network would make it extremely challenging to incorporate in a large-scale LSM to model the brain.

To gain insight about the computational principles governing brain function and dysfunction through LSM, scientists need to assemble the model with accurate and computationally efficient artificial neurons. Current models either oversimplify dendritic tree computations or are too computationally expensive to incorporate in LSM. Therefore, a an artificial neuron model that can balance the trade-off between accurately modeling dendritic computations and computational complexity needs to be researched. This new building block for LSM will help create a better replica of the computational model of the brain and may help researchers understand brain functionality.

4 Problem

4.1 Problem Description

There is a gap for a simple (low complexity), yet sufficiently accurate modelling technique for, specifically, an L5 cortical pyramidal neuron. This project aims to address this gap and ultimately produce a model with low complexity and an accuracy higher than the McCulloch-Pitts neuron model.

In addition to requiring low-complexity and a minimum accuracy, other obligations for this project included:

- The potential for seamless integration within Dr. Blohm's larger, more all-encompassing model of the brain.
- For the same reason, the model had to be a neural network created with PyTorch.
- The ability to evaluate models by creating a metric for computational complexity.

Given these requirements, several constraints naturally arose in which this project was forced to overcome. Some of the more notable include:

- The existing neural network in which this project drew inspiration from, created by Beniaguev, was developed using Tensorflow rather than PyTorch [9]. Moreover, the documentation and readability of his solution was difficult to comprehend.
- The dataset used from Beniaguev was over 10 GB in size.
- The neuron output (spikes fired within a specific time window) were incredibly sparse within the dataset.

To overcome the first constraint, prior to model building, the structure of Beniaguev's model was carefully analyzed and a custom Pytorch Dataloader, then subsequently neural network were created. Regarding the documentation, the team reached out directly to Beniaguev who kindly clarified questions and misunderstandings about the dataset. Furthermore, to save time and computational demands, the team only used 3% of the data for training and testing new models.

The third constraint posed a challenge in developing an evaluation method for the computational complexity, especially in creating a probability distribution. This was tackled by employing quantization – as shown in section 5.4.

4.2 Problem Statement

Explore different neural network architectures to model the nonlinear function of L5 pyramidal cortical neuron's dendritic trees in determining somatic voltage and evaluate the trade-off between the model's accuracy and computational complexity.

4.3 Stakeholders

Various stakeholders are involved with the application of a single neuron model. These stakeholders and their relation to the project is shown below in Table 1:

Table 1: Stakeholder Analysis

Stakeholder	Stakeholder needs	Effect on project
Dr. Blohm, Direct recipient of the final model and all associated code	- Working model by April, 2022 - Code with the model that is compatible with a current network modelling project - Exploration of basic neural network architecture performance	- Model that is generalizable to single neurons with a dendritic tree - Model developed in Python before April 30th, 2022 - Code delivered code as a package, uploaded on Github, with proper documentation
Mathematics Researchers, General research community	- Use proper mathematical tools in deep learning (amount of data used, pre-processing data, network creation) to understand results [10] - Use proper mathematical tools to model synaptic inputs from dendritic trees (stochastic processes and nonlinear operations) [7]	- Assess how much data is required to train/test a reliable DNN to model a single neuron - Understand mathematical operations used by dendritic trees to compile synaptic inputs

Continued on next page

Table 1: Stakeholder Analysis (Continued)

Scientific Researchers General research community	- Single neuron model with non-linear assumptions to account for complex computations at dendrite [11] - Decrease dependence on non-human primate models for studying brain function and disease [12] • It is unethical to use captured non-human primates due to concerns with captive management, invasive studies, and non-consensual subjection to treatments [13]	- Add dendritic tree involvement for brain network modelling, which can lead to more accurate building blocks for LSM models of the brain and better interpretability for scientific community.
Medical Doctors and Caregivers	- Specialized treatments for patients suffering from Neuromuscular disease are needed to reduce wait times of several months to one year in Ontario [14] - Reduce Medical costs and labour hours for Doctors and Caretakers [15] who have patients suffering from Neuromuscular disease.	- Generalizable to specific neurons and neurotransmitters to emulate real-life neurons [11] - Constrained usage of discrete vs continuous input signals in the model, as they model different behaviours (focused task vs. everyday living) [11]
Patients with Neuro-muscular illnesses	- More than 50,000 patients in Canada suffer from Neuromuscular disease and require sophisticated treatment and adaptive devices for rehabilitation. - Medical costs are expensive and significant labour hours are required to rehabilitate patients.	- Easy to revise model to continue model improvements after the team is finished with the first stage - Creating an open-source model to increase collaboration between medical professionals

5 Design Process

5.1 Overview and Assumptions

The design process was broken down into three components: data pre-processing, neural network design and training, and evaluation. During the data pre-processing phase, statistics of the dataset were explored to identify redundancies and assess the variability of the data. Three different types of neural networks were explored: temporal convolutional networks (TCNs), fully connected networks (FCNs), and convolutional networks (CNNs). The complexity of these models was evaluated using information theoretic tools.

The goal of this project was to model the I/O processes of dendritic trees in a single cortical pyramidal neuron by using an analogous neural network. More specifically, The key objectives were to:

(1) Convert the Beniaguev models from Tensorflow to PyTorch
(2) Develop, train, and test simple neural networks to examine the model accuracy with varying architecture
(3) Evaluate the model using mathematical and neurobiological tools to gauge its complexity

A general representation of the goal is shown in Figure 4 below.

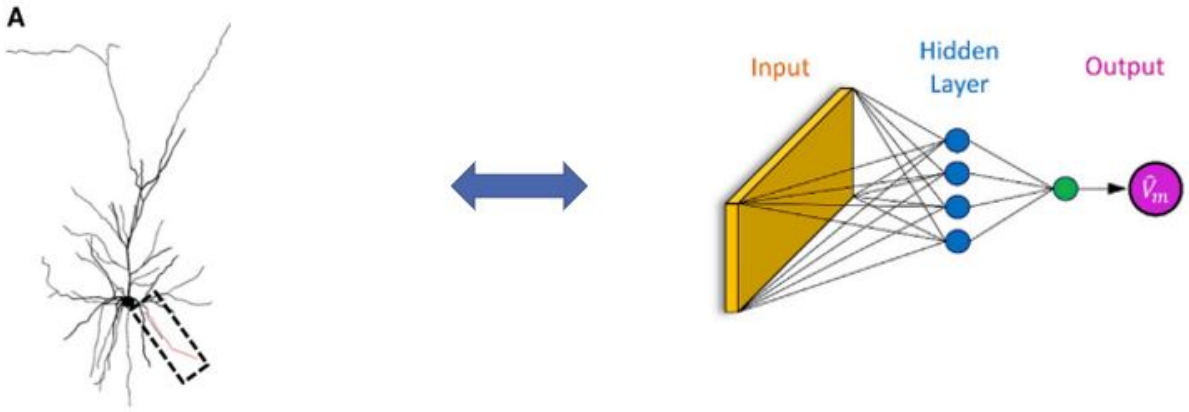


Figure 4: I/O mapping of a single dendrite of cortical pyramidal neuron by using an analagous neural network

5.2 Data Set Analysis and Loading

The dataset used for modelling is a 10-hour simulation of a cortical layer 5 pyramidal neuron. This dataset has been used in prior research to successfully model cortical neurons as deep neural networks. The total size of the dataset is 100 GB and the resolution is 1ms time intervals. The dataset represents the inputs and outputs of neuronal activity. The inputs are the activations of each synapse of the dendritic tree per 1ms interval, represented as binary values. The outputs are the somatic voltages of the neuron per 1ms interval, represented as real values.

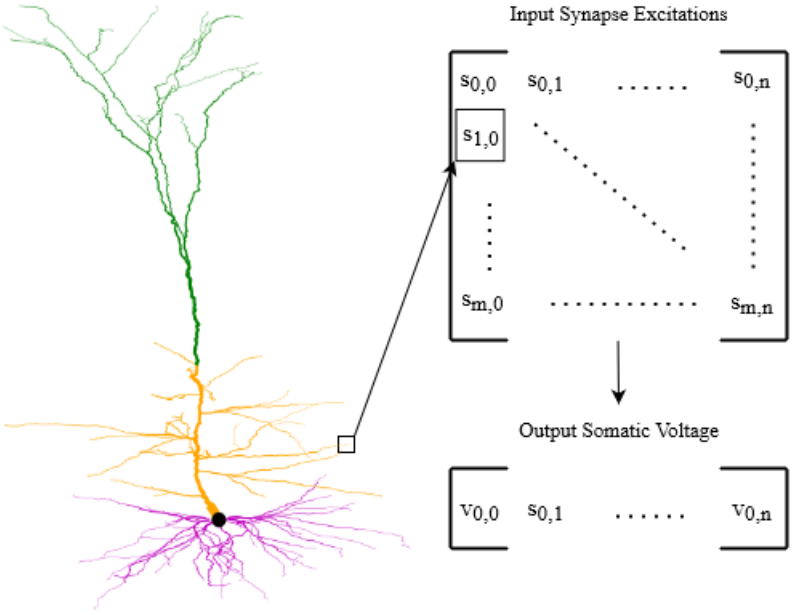


Figure 5: Dataset Visual Description

To formalize the input-output map construction, we define the inputs as a spike train, which

is a sequence of binary vectors $S(N,t)$, such that $S(N,t) \in [0,1]^N$ where N is the number of synapses in the dendritic tree and t is a time index. For each time interval t, $S(N,t) = (s_0, s_1, \cdots, s_N)$, $s_i \in [0,1]$. The input spike trains have a corresponding sequence of output somatic voltage values $v(t)$. The goal is to predict $v(t)$ based on input window of size K where $(S(N,t_i), S(N,t_{i-1}), \cdots, S(N,t_{i-K})) \mapsto v(t_i)$.

Using this dataset does come with some constraints and assumptions. The distribution of inhibitory and excitatory inputs are uniform. The input-output resolution is also constrained to 1ms time intervals.

The data was pre-processed using Python scripts to be fed into the models. First, the inputs and outputs were stored in separate data structures. For each simulation file, the synaptic inputs were stored in a matrix S of size $N_{syn}XM$ where N_{syn} is the synapse number and M is length of the time interval in milliseconds. Due to memory constraints, we pre-processed 1.4 million datapoints (in milliseconds) amounting to around 3.6 hours of simulation time. The number of synaptic inputs, N_{syn} is 1278. The corresponding voltage output was stored in a vector of length M, each entry corresponding to a column of the input matrix X.

The input-output map is fed to the model through a custom PyTorch dataset class. The dataset class divides the into batches, where a batch size of 28 was used. The class also had an input parameter that identified the window size K used to sample the time interval used for sampling. The batching method relied on using Python generator functions. This allowed us to store each batch in memory sequentially without storing the whole input matrix in memory. Given a model ANN, we make a prediction denoted by $\hat{v}(t) = ANN(\hat{S}(t,k), \theta)$ where θ denotes the model parameters.

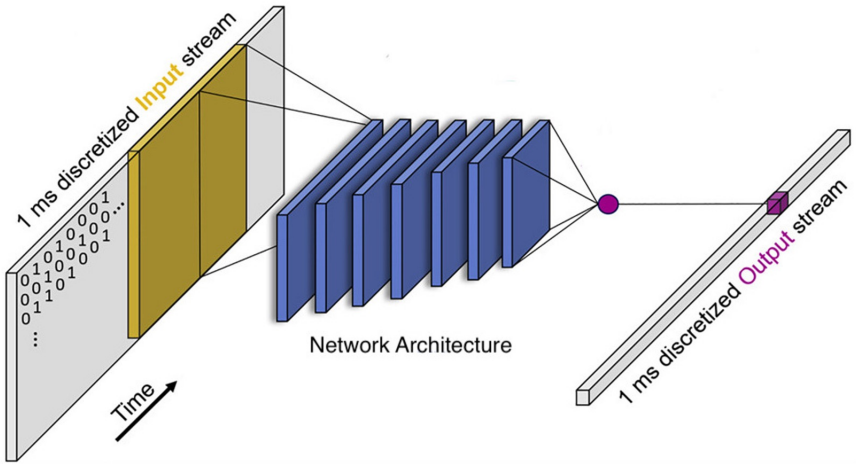


Figure 6: Model Architecture

5.3 Model Development and Training

The first objective was to replicate Beniaguev's temporal convolutional neural networks in PyTorch as shown in Figure 6. To translate the models from Tensorflow to PyTorch, a python script that extracts the existing parameters from a .h5 file to a .json file was created. The extracted parameters included weights, biases, size of channel inputs and channel outputs. To confirm accurate re-production of the models, a portion of the dataset was used to test the output prediction accuracy. The replicated models produced an output prediction accuracy of 90% which was less than

the 99% accuracy found in literature [9].

The current practice in Dr. Blohm's lab is to use the McCulloch-Pitts model, a fast, linear models that predict soma voltages using weighted sums of dendritic inputs. For the second objective, small FCNs, CNNs, and TCNs were built to explore how the varying architecture impacted model performance. The aim was for the models considered to perform better than the models currently in use by Dr. Blohm. Therefore the McCulloch-Pitts model was first trained and tuned on the dataset to achieve a baseline accuracy. Next, different network architectures were explored by varying parameters such as activation functions, number of layers, number of nodes, and connection of layers. Since the dataset is 100GB, the models tested were only trained on 3% of the dataset given the limited computational resources.

The purpose of training new models was to learn how output accuracy responds to changing neural network architecture. The training of each model utilized an Adam optimizer to minimize the loss function of the model. The Adam optimizer computes individual adaptive learning rates for different parameters from estimates and second moments of the gradients [16]. The algorithm calculates an exponential moving average of the gradient and the squared gradient where β_1 and β_2 control the decay rate of the moving averages during the training process [16]. Thus, an Adam optimizer was used as it produced the most accurate results.

Algorithm 1: ADAM Optimizer Algorithm

Data: Initialize input data x_0, α_t, Batch Size m
while x_t *not converged* **do**
  $t = t + 1$;
  Query m samples from the data x_0;
  for *sample in samples* **do**
    Forward Propagate to find the cost using $a^L = \sigma(W^L \cdot a^{L-1})$;
    Back-propagate to find the errors in each layer using $\delta_j = (W^T \cdot \delta_{j+1}) \odot \sigma(z_j)$;
  end
  Compute first moment $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
  Compute second moment $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
  Compute unbiased first moment $\hat{m}(t) = \frac{m(t)}{1-B_1^t}$
  Compute unbiased second moment $\hat{v}(t) = \frac{v(t)}{1-B_2^t}$
  Update the weights using computed moments: $w_{t+1} = w_t - \alpha_t \frac{\hat{m}(t)}{\sqrt{\hat{v}(t)+\epsilon}}$;
end

The models used a Mean Square Error (MSE) loss function as it ensured that the trained models did not contain outlier predictions with large errors that could contribute to an overall inaccurate output prediction. This is a result of MSE placing larger weight on such errors by squaring them. However, this highlights MSE main disadvantage of magnifying errors if a single poor prediction is made by the model. However, since the objective was to obtain a well-rouned model that could perform well on the majority of the inputs, this issue was resolved. The MSE is formally defined by the equation below.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

A combination of backpropagation and the Adam optimizer algorithm were used to update the

weights and biases of the neural networks. Backpropagation is split into two parts: the forward and the backward pass. Figure X demonstrates an example of a FCN that uses backpropagation to learn the weights and biases.

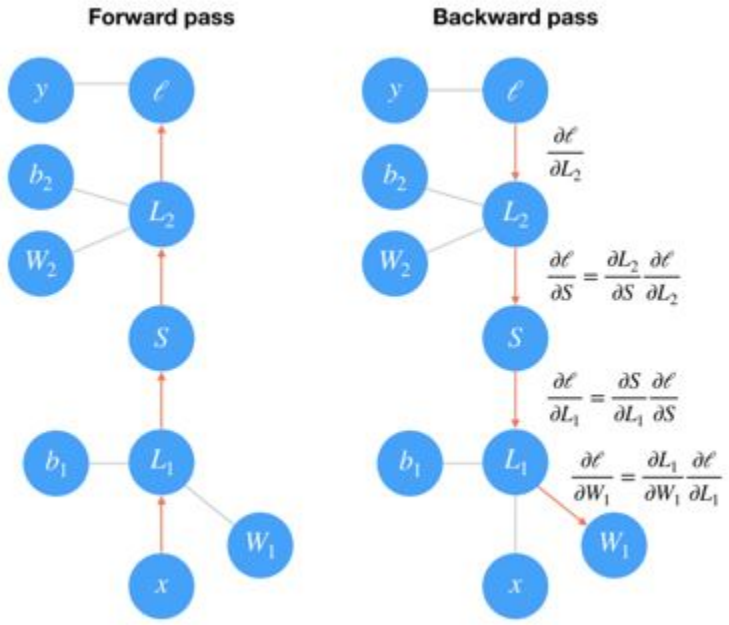


Figure 7: Backpropagation Example

For the case above, the input x is passed through a linear transformation (L_1) using weight (w_1) and bias ($b1$). Next, the output goes through an activation operation such as sigmoid (S) and another linear transformation (L_2).Lastly, the loss (l) is calculated. This is used to measure the quality of the network's predictions.

In the backward pass process, the goal is to adjust the weights and biases to minimize the loss. Throughout this process, the gradient of the loss is propagated backwards through the network. There exists a gradient between each operation. To train the weights, the incoming gradient is multiplied with the gradient for the operation. Simply, this is calculating the gradient of the loss with respect to the weights using chain rule. The weights are then updated using a learning rate, alpha.
The equations that govern forward propagation for a fully connected network, given as following:

- $z_j^L = \sum_{k=1}^{n} w_{jk}^L \cdot a_k^{L-1}$ where n is the number of neurons in layer $L-1$
- $a_j^L = \sigma(z_j^L)$, describes the activation value of neuron j in layer L
- $a^L = \sigma(W^L \cdot a^{L-1})$ is the vectorized version

To retrieve the derivatives for the weights in a fully connected network, the following equations describe the process of backpropagating:

- $\delta_j = \frac{\partial C}{\partial z_j}$ is defined as the error in layer j
- $\delta_j = (W^T \cdot \delta_{j+1}) \odot \sigma(z_j)$ is a vectorized version of finding the errors in each layer

- $\frac{\partial C}{\partial w_{jk}^L} = a_k^{L-1}\delta_j$ is the error on each weight in the network

5.4 Design of Information Theoretic Metric

As mentioned in the problem statement, we wish to evaluate the computational complexity of the model. In this case we refer to complexity as a measure of how much information the system transfers from inputs to outputs. Many theoretical and experimental studies have utilized Shannon's information theory as a way to analyse the information transfer through neurons. We will use some of the methods which have been applied to similar neurological data, to provide an analysis and method for obtaining such metrics. The metrics derived from the simulation data will be compared to those from the models input and outputs. This will give an idea of how well the model emulates the transfer of information of the system it is modelling.

To develop the information transfer, the mutual information between input and output, and the entropy of the input and output is needed. This required developing a distribution for the inputs, a transition distribution and a distribution for the outputs. Equation 3 shows the mutual information and its required distributions, and 4 shows the metric for information transfer, a ratio of mutual information to entropy of the source. Note here X and Y refer to random variables of the input and output respectively.

The mutual information between two random variables is expressed below.

$$I(X;Y) = \sum_{x\in X, y\in Y} P(x)P(y|x)\log\frac{P(y|x)}{P(y)} \tag{3}$$

Below is the ratio between mutual information $I(X;Y)$ and the input entropy H_x, which we refer to as the information transfer. [17].

$$F = < \frac{I(X;Y)}{H_X} > \text{ samples} \tag{4}$$

Acquiring values for $I(X;Y)$ and F was not straightforward. There were prerequisite models and assumptions (i.e. the input distribution, conditional distribution, a method of analyzing inputs and outputs, etc) [18]. Hence, the analysis aligned with the following structure:

(1) Process the inputs
(2) Determine an input distribution
(3) Calculate input entropy
(4) Determine the conditional distribution
(5) Calculate mutual information and average transferred energy

The input vectors were encoded using a summation encoding, effectively this encodes the number of 1s in the vector [18]. These values were then analyzed to form the distribution for the inputs. By examining the mean and standard deviation of these summation encodings it was determined that they could be modelled as a Gaussian random variable. Figure 8 shows the observed summation encodings from the simulation data and a sample Gaussian with mean and variance of the data. It should be noted that using a purely empirical distribution (occurrences divided by total events) yielded similar results. Using a Gaussian model allowed for a simple calculation of the input entropy, as seen in Equation 5 and 6.

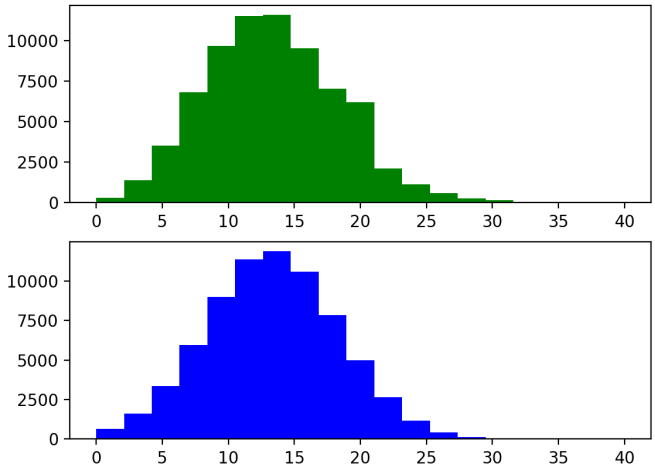


Figure 8: Sample Gaussian (blue) and Observed Summation Encodings (blue)

$$P(x) = \frac{1}{4.98\sqrt{2\pi}} e^{-(x-13.3)^2/2(4.98^2)} \tag{5}$$

$$H_X = -\sum_{x \in X} P(x) \log P(x) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2} = \frac{1}{2}\log(2\pi \times 4.98^2) + \frac{1}{2} = 4.14189 \text{ bits} \tag{6}$$

Therefore, the input entropy for this Gaussian distribution is 4.14189 bits.

Next, the conditional distribution needed to be created. In other words, this is the probability of observing a specific soma voltage given an input encoding.

To create this distribution the somatic voltages were quantized. This was done because the voltages were precise floating point numbers, and thus would yield a purely uniform distribution if not quantized. It is important to note that the choice of quantizer results in different distributions. To contextualize the results, we will look at the mean squared error in relation to different quantizers. It is difficult to understand what is a good error in absolute terms, however it allows us to contextualize the results. The errors for 3 different quantizers can be seen in Figure 9 and a visualization of the quantizing process can be seen in Figure 10.

A decision to use the somatic voltages instead of spike trains as our output random variable was made due to the sparsity of 1s in output spike trains. This sparsity lead to the output distribution being nearly deterministic, which leads to difficulty in generating the entropy measure of the output.

MSE	Quantizer Cell Size
0.0052	0.25
0.0130	0.5
0.0243	0.75

Figure 9: MSE and Quantizer Cell Size

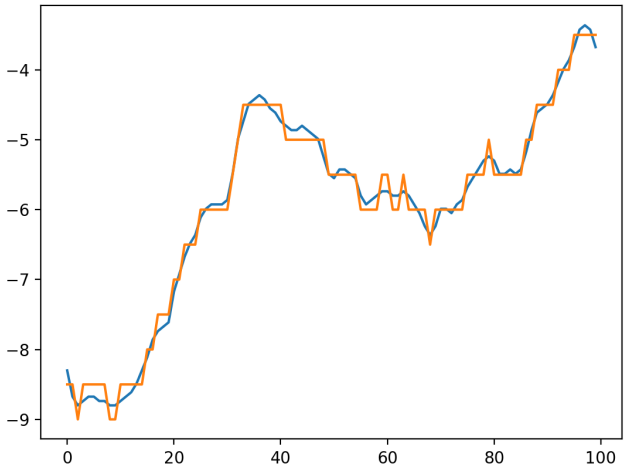


Figure 10: Sample of 100 Quantized Soma Voltages

Once the somatic voltages were quantized we developed the conditional distribution empirically by counting the number of input to output pairings divided by the total occurrences of the conditioned upon variable, as seen in Equation 7, where n term refers to the count of occurrences at index i and j.

$$P(y|x) = \frac{n_{ij}}{\sum_{k=1} n_{ik}} \tag{7}$$

With the necessary probability distributions developed we used this method on both the simulation data and on the models input/output mappings. With the goal of comparing the models information transfer to that of the simulation data. The process was run over multiple files using 10 iterations (60,000 inputs). The results are discussed in section 6.5.

6 Results

6.1 Model Accuracy Results

Several different model architectures were trained on the dataset to evaluate which artificial network could best predict the somatic voltage potential.

6.1.1 McCulloch-Pitts Model

The McCulloch-Pitts Model was first trained and tested to provide a baseline for current model performance. It was necessary that the new models proposed performed better than the weighted linear sum currently in use in Dr. Blohm's lab.

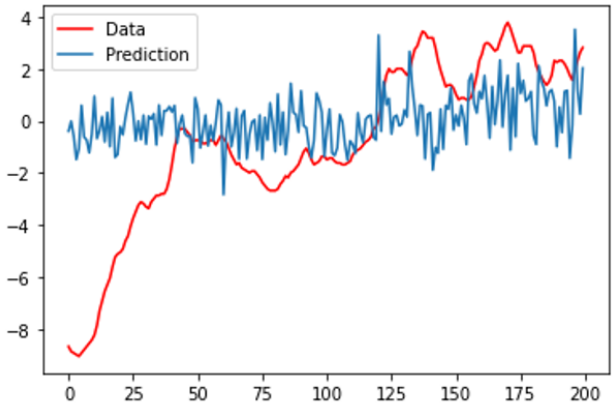


Figure 11: Prediction of Somatic Voltage with McCulloch Pitts Model

The weighted sum model did not accurately follow the trend of the true somatic voltage as shown in Figure 11 and resulted in a validation accuracy of 10.3% on the test set.

6.1.2 Beniaguev's Models

The team recreated Beniaguev's four temporal convolutional neural (TCN) networks which performed the best [9]. The smallest model tested had 7 hidden layers as shown in Figure 12 and consisted of hidden layers with causal convolutions, rectified linear unit (ReLU) activation functions and batch normalization. With the pre-trained parameters, the models were only able to achieve 90% accuracy on average on the test data.

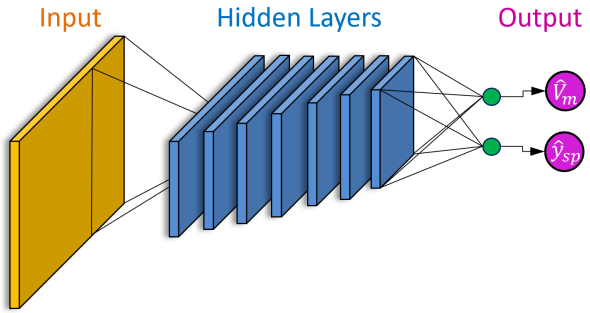


Figure 12: Smallest TCN architecture tested from Beniaguev's results [9]

6.1.3 Fully-Connected Neural Networks

Three different depth fully-connected neural networks (FCNs) with varying number of parameters per hidden layer were trained and tested to determine their performance on the dataset. The initial assumption was that networks with greater complexity would yield higher test accuracies.

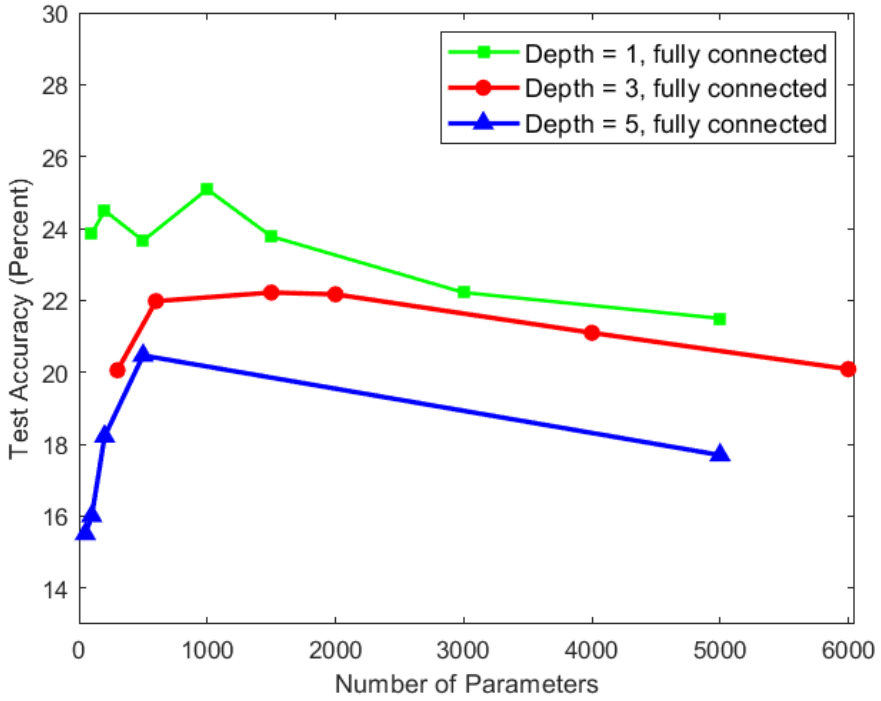


Figure 13: Accuracy of Tested FCNs with Different Depths and Parameters

The best performance was achieved with one hidden layer with 1000 units as shown in Figure 15. Overall, performance decreased with increasing number of parameters and increasing number of layers.

6.1.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) performed similarly to the base-line McCulloch-Pitts model despite varying the size of the kernels. Figure 14 summarizes the network architectures tested and their validation accuracy.

Layers	Structure	Validation Accuracy
1	1278 channels → 1 output → Rectified Linear Unit Function (ReLU)	15.98%
2	1278 channels → 128 outputs → 1 output → ReLU	16.63%
	1278 channels → 150 outputs → 1 output → ReLU	14.47%
	1278 channels → 64 outputs → 1 output → ReLU	14.69%
3	1278 channels → 128 outputs → 64 outputs → 1 output → ReLU	15.80%
	1278 channels → 128 outputs → 32 outputs → 1 output → ReLU	16.06%
	1278 channels → 64 outputs → 32 outputs → 1 output → ReLU	15.54%
4	1278 channels → 128 outputs → 64 outputs → 32 outputs → 1 output → ReLU	16.45%

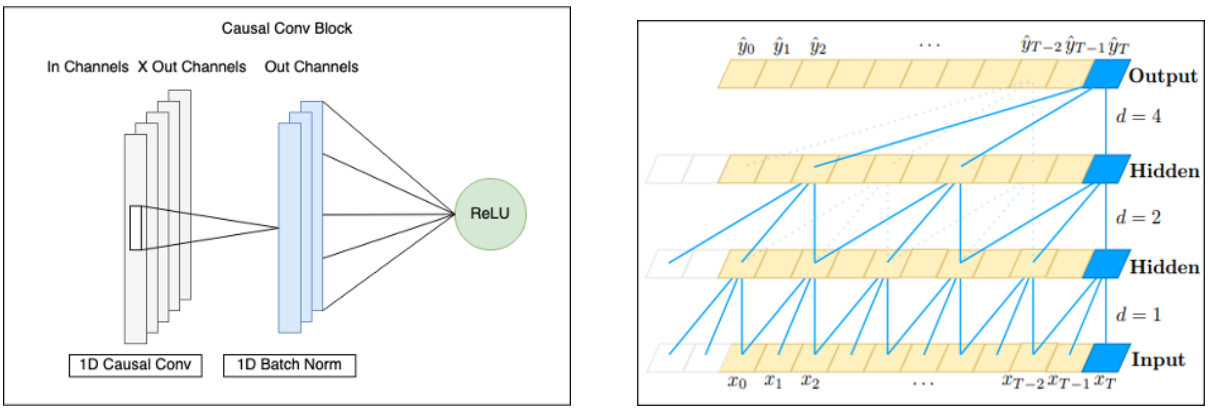
Figure 14: 1D Convolutional Neural Network Models and Accuracy

6.1.5 Temporal Convolutional Neural Networks

Architecture:

Temporal Convolutional Networks (TCN) offer the following properties that are useful for modeling neuronal activity:

- Temporal modelling of the dendritic tree function using convolutional layers.
- Use of causality in convolutions to ensure no information leakage from future events in current predictions.
- Gives flexibility of large receptive fields through dilation whilst maintaining the same memory and computational cost.



(a) Convolutional Blocks for TCN Models Models

(b) Overall TCN architecture

Figure 15: TCN Modelling Results

Each layer in a TCN is composed of a convolutional block shown in Figure 15 (a). This block has three components:

- A 1-dimensional causal dilated convolutional layer.
- A 1-dimensional batch normalization layer.
- A Rectified Linear Unit (ReLU) activation function.

Figure 15 (b) shows the complete architecture where each layer represents a convolutional block with an increasing dilation factor of 2. The training process for TCN's is identical to FCN's with the exception of the forward propagation rule in the convolutional layer. For kernel size k, convolutional filters $f(i)$, inputs x, and dilation factor d_i, the forward propagation in a dilated convolutional layer is:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) * x_{s-d_i}$$

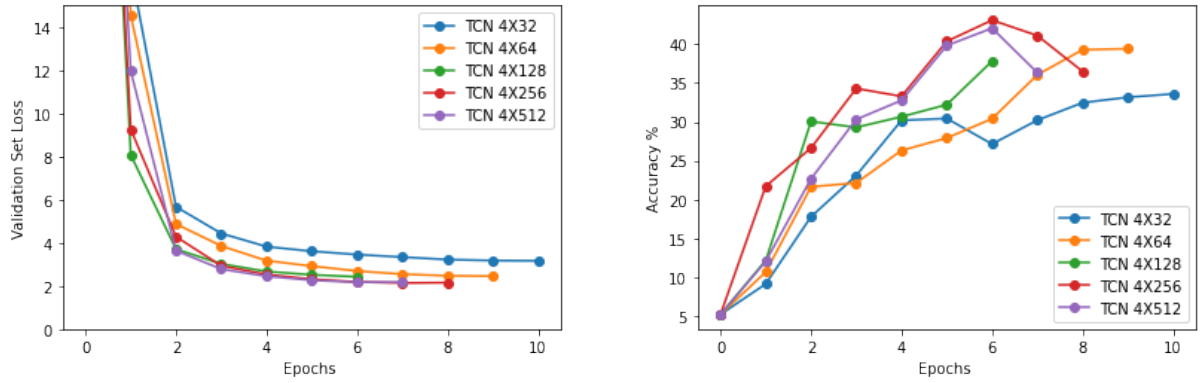
A variety of TCN architectures were explored to yield different results. For the data inputs, a window size of 400ms was used with an overlap of 150ms between each consecutive data sample. The dataset was split into 85% for training and 15% for validation.

Results:

Model	No of Filters	1% Accuracy
TCN 4X32	32	30.52%
TCN 4X64	64	35.27%
TCN 4X128	128	39.88%
TCN 4X256	256	42.04%
TCN 4X512	512	43.79%

Figure 16: TCN Model Results

Figure 16 shows the extracted results from the TCN modelling. A clear dependence was established between network width and modelling accuracy. There no establishe dependence between network depth and accuracy. Experimentation showed that a network depth of 4 yielded the best modelling results on our dataset. The TCN widths ranged from 32 to 512 features, multiplying by a factor of 2 for each consecutive model. The highest accuracy achieved was by the $TCN4X512$ model achieving 43.79% using a 1% accuracy metric.



(a) Validation Loss of TCN Models

(b) 1% Accuracy of TCN Models

Figure 17: TCN Modelling Results

In Figure 17 (a) and Figure 17 (b) the respective validation loss and 1% accuracy measure for all the tested TCN models is shown. The figures show that as the network width increases, the performance improvement starts to plateau both for validation loss and the accuracy metrics. Based on these results, the pursue of wider TCN architectures for modelling is not advantageous since that would significantly increase complexity for minor performance improvements.

6.2 Information Metric Results

As can be seen in Figures 18 and 19, the mutual information and transferred entropy was compared between the simulation data and the best performing TCN models. The simulation data served as the ground truth whereas the TCN models attempted to replicate the true dendritic mechanism. Evidently, the simulation data had a larger mutual information between input and outputs as well as a larger transferred entropy than any models evaluated.However, as the model accuracy increases, the metrics for the models approach the metrics for the simulation data. This can be

seen moving from top to bottom in Figure 19.

Although these numbers are hard to interpret on their own, comparing simulation metrics with model metrics would allow researchers to see how well a new biologically-inspired artificial neuron replicates the information flow from inputs to outputs. This comparison would allow researchers to comment how well the model represents the dendritic tree function in the artificial neuron.

Mutual Information (bits)	Transferred Entropy (bits)
0.331	0.079

Figure 18: Simulation Data Mutual Information and Transferred Entropy

Model Name	Mutual Information (bits)	Transferred Entropy (bits)
4x64	0.253	0.0611
4x128	0.284	0.068
4x256	0.293	0.071

Figure 19: Model Mutual Information and Transferred Entropy

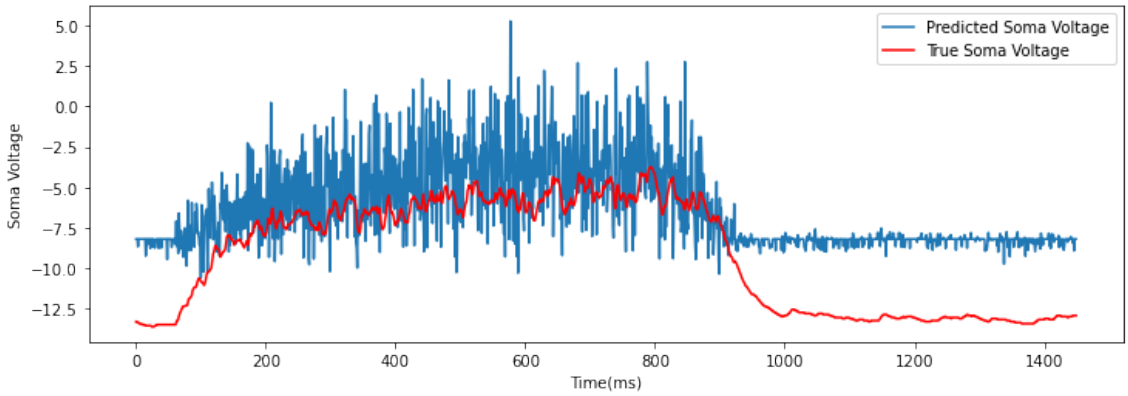
The largest limitation of this metric was the use of the summation encoding, this contains no temporal information and thus loses signification amounts of information. Secondly, the choice of quantizer introduces a large bias in the results, since the cell sizes influence the developed distributions. Lastly, without similar systems to compare to these metrics provide only a valuable comparison between the models information transfer and that of the actual simulation data. For future improvements of the encoding, ideas such as measuring the mean distance between ones or using the binary sequence itself as the input symbol would contain more information about the timing and placement of 1s, and thus provide a more accurate metrics.

7 Discussion

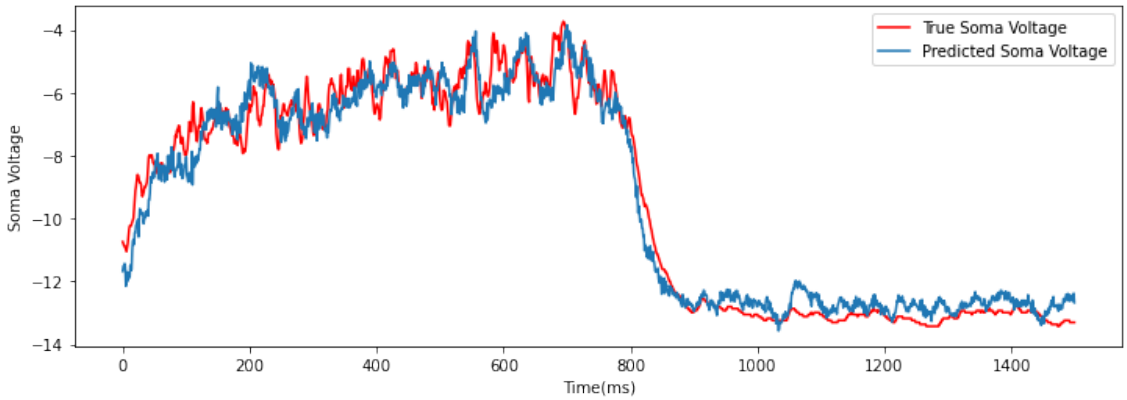
The current model used in Dr. Blohm's lab achieved an accuracy of 10.3%, indicating that it is not able to predict the somatic voltage given the input spike trains. However, this model is widely used due to its computational efficiency, especially when the full brain model is being tested.

When experimenting with FCNs, the validation accuracy did not necessarily increase as the depth and number of parameters increased. It was interesting to note that a small FCNs with only 100 tunable parameters had a validation accuracy of 23.8% whereas one with 200 parameters had 24.5%. Contrary to the initial assumption that was made, networks with a higher degree of complexity, whether they be through number of layers or number of parameters, did not produce more accurate results. Figure 15 demonstrates that each of the networks reached their peak accuracy at roughly 1000 parameters, after which it steadily decreased. Furthermore, the more layers a network had, the less accurate the results were overall. This information can be used to train an efficient network with the lowest computational complexity. Overall, these models provide a slight accuracy increase over the McCulloch-Pitts model and can be feasible to implement due to their small size.

When testing CNNs, both 1D and 2D convolutions were tried however 2D convolutions performed worse than the base-line model. Although CNNs may take a longer time to train, the architecture allows for computational efficiency when testing due to parameter sharing and the sparsity of connections. The kernel/ filter weights must be tuned but they are shared among several inputs. Therefore, it was highly desirable for good model performance, however traditional 1D CNNs performed just as good as the McCulloch-Pitts model.



(a) McCulloch-Pitts Prediction



(b) $TCN4X512$ Prediction

Figure 20: TCN Modelling Results

Figure 20 shows a comparison between the prediction quality of the McCulloch-Pitts model verses the best performing TCN architecture. The comparison is across a randomly chosen $1500ms$ interval from the validation set. The McCulloch-Pitts model is noisy with high frequency oscillations and fails to respond steep changes in the somatic voltage amplitude. The TCN model is much more stable, less noisy, and captures the amplitude of the somatic voltage accurately throughout the interval.

Category	Weighted Sum Model	TCN Model
1% Accuracy	10.3%	43%
20% Accuracy	15.4%	73.4%

Figure 21: McCulloch-Pitts verses TCN comparison

Figure 21 shows the accuracy comparison between the McCulloch-Pitts model and the TCN model. The TCN model about quadruples the accuracy of the McCulloch-Pitts model using a 1% accuracy measure. Using a 20% accuracy measure, the TCN model has a 30% performance gain while the McCulloch-Pitts model only has a 5% performance gain. This further exemplifies how much better the TCN models are at capturing the function of the dendritic tree.

7.1 Limitations and Trade-offs

One limitation is that all the given model types fail to incorporate the relative position of the dendritic synapses. Capturing this information as part of the the modelling might have yielded different result that provided better insight on the true function of the dendritic tree. Another limitation was the restriction of the model sizes by the available GPU memory we had on our devices. Access to more compute resources would have allowed for more experimentation with various model types. The dataset utilized contained almost 100GB of simulation data. Due to limited compute resources and time constraints, only 3% of this dataset was used for the training process.

The biggest trade off observed on the modelling aspect is the empirical nature of training machine learning models. The models' parameters were experimentally determined or chosen based on past research experimentation there is no confirmation optimality of these parameter choices. Another issue that arises is extensbility. Training on only a small portion of the dataset makes it unclear whether these results will extend if trained on the whole dataset, or different simulation data. It might be the case that the optimal architecture varies as the data size varies or the simulation data changes.

7.2 Next Steps

In the future, the best models should be trained and tested on the entire dataset to see if the result can be generalized. Once the model optimizing both accuracy and computational efficiency is selected, researchers may train the model on the entire dataset to tune the appropriate parameters and may implement it in their neuronal model.

The group plans on making the remaining code readable and modifiable by Dr. Blohm's research group. When reaching out to Beniaguev and his colleagues, they suggested that we try the following modifications to the networks to see if they improve modelling:

- Use of locally connected networks to capture information with regard to the relative position of dendritic synapses in modelling.
- Use of skip connections in deep network models to avoid vanishing gradients.

- Usage of graph neural networks which identically model the morphology of the neurons as part of the modelling process.

8 Engineering Impacts

8.1 Application

One of the many strides that the scientific community has been making with the design of deep neural networks has been Brain-Computer Interface (BCI). BCI analyzes brain signals and encodes intention from a person, decodes the intended output, and then translates the intention to a device to complete an action. The models and networks designed for this initiative do exactly this. Inputs from surrounding neurons are weighted and an appropriate output is generated as a prediction. The relationship between these models and BCI can have beneficial effects to people suffering from neuromuscular disease, which causes dysfunction of muscles due to nerve and muscle problems. This affects over 50,000 Canadians and is incurable, however significant developments have been made to rehabilitate with the use of adaptive devices.

Modelling a cortical pyramidal neuron's dendritic tree to analyze the relationship between inputs and outputs can help researchers to create neural networks capable of detecting and diagnosing psychiatric disorders on an individual level. By changing how the LSM model reacts to certain NTs and receptors, researchers will be able to predict whether these modifications lead to specific mental illnesses, and may also determine ways to diagnose patients with these diseases. With the knowledge of how mental illnesses affect the brain chemistry, the effects of new medicine can be modelled prior to human trials using the LSM model. will be able to visualize the expected and the unexpected effects of their new medical drugs on the brain and may spur drug discovery for neurophysiological disorders. The model may also be used to determine the effectiveness of psychosocial therapies such as cognitive behavioural therapy (CBT), or to determine the effectiveness of these therapies when administered together to resolve mental illnesses.

8.2 Triple Bottom Line

Understanding single neuronal interactions and their impact on people and the environment can greatly increase the quality of life of people suffering from neuromuscular disease, decrease chances of injury, and relieve burden on caregivers. Traditional adaptive devices can become greatly improved upon, reducing the need to change them based on the function will be performed. Furthermore, a reduced reliance on primate and rodent subjects for scientific testing will carry a positive doctrine among the scientific community.
Computational complexity varies greatly when training neural networks. For initiatives such as modelling single cortical neurons, the computations can be performed on any machine in a reasonable amount of time. Once the complexity of the neural network is increased to the point where a single machine can no longer be used, the computation must be outsourced to clusters contained in large datacenters, which are often still powered by fossil fuels. Reducing the complexity of neural networks can have a massive impact on the environment, as demonstrated by the fact that if 10 million people were to each delete 200 spam emails, 20000 metric tons of carbon emissions would be reduced[19]. This may not seem like a lot compared to other sources of emissions, however it can have a significant impact with the amount of data in circulation.

8.3 Economic Analysis

The cost of finding a suitable model is mainly associated with the cost required to train a neural network. This cost will be either associated with cloud computing units or energy costs from

using a graphical processing unit (GPU) to train the network. GPU usage has significant energy consumption when training neural networks. Therefore, when scaling this project to model millions of neurons, there is substantial environmental impact and monetary cost. It is important to find the proper balance between simplicity and accuracy. As demonstrated in the Results section, models with higher complexity do not necessarily increase accuracy. Such observations can have significant financial impact when optimizing neural networks, as finding the simplest model with the most accurate results is crucial to reducing cost.

8.4 Standards, Codes, and Ethical Considerations

While recent developments in AI have been accelerating at an unprecedented pace, it is important to consider both the positive and negative effects of implementing such a technology. An engineer must understand that poorly designed projects built on data that is inadequate, biased, or faulty can have unintended and potentially harmful consequences. Furthermore, rapid advancement in machine learning means that in many cases, engineers who designed the neural networks are themselves completely unaware of how conclusions are being made. In many cases AI that even the most experienced designers cannot explain are making decisions that could affect society. It is for these reasons that engineers must treat developing artificial systems with the same scrutiny as any other project.[20]

9 Conclusion

The team explored different neural network architectures to model the nonlinear processing of dendritic trees in L5 coritical pyramidal neurons including FCNs, CNNs, and TCNs in PyTorch. Information theory techniques were also used to develop a way to quantify the entropy transfered from the dendritic branches to the soma. TCNs performed the best out of the architectures tested, achieving 43.8% accuracy on the test data. Both FCNs and CNNs did not significantly improve the somatic voltage prediction from the McCulloch-Pitts model. In FCNs, increasing number of hidden layers or parameters decreased model performance. The entropy transfered metric can be a useful way to compare the biological accuracy of information flow from inputs to outputs in newly developed biologically-inspired artificial neurons. In the future, locally connected networks or causality in convolutions should be tested to see if this improves somatic voltage prediction accuracy and the entropy transferred metric. These models and metrics can help Dr. Blohm implement a more biologically-accurate and computationally efficient artificial neuron model to be used in LSM, in hopes of better modeling the computational circuits in the brain.

References

[1] Yong Zhang et al. "A Digital Liquid State Machine with Biologically Inspired Learning and Its Application to Speech Recognition". In: *IEEE Transactions on Neural Networks and Learning Systems* 26.11 (Nov. 2015), pp. 2635–2649. ISSN: 21622388. DOI: 10.1109/TNNLS.2015.2388544.

[2] Gideon Gbenga Oladipupo. "Research on the Concept of Liquid State Machine". In: (Oct. 2019). arXiv: 1910.03354. URL: https://arxiv.org/abs/1910.03354v1.

[3] S. Hayman. "The McCulloch-Pitts model". In: *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*. Vol. 6. 1999, 4438–4439 vol.6. DOI: 10.1109/IJCNN.1999.830886.

[4] Wulfram Gerstner et al. *Neuronal Dynamics- From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. URL: https://neuronaldynamics.epfl.ch/online/Ch1.S4.html.

[5] Eugene M Izhikevich. "Simple Model of Spiking Neurons". In: *IEEE TRANSACTIONS ON NEURAL NETWORKS* 14.6 (2003). DOI: 10.1109/TNN.2003.820440. URL: www.izhikevich.com..

[6] Leonardo L. Gollo, Osame Kinouchi, and Mauro Copelli. "Single-neuron criticality optimizes analog dendritic computation". In: *Scientific Reports 2013 3:1* 3.1 (Nov. 2013), pp. 1–9. ISSN: 2045-2322. DOI: 10.1038/SREP03222. URL: https://www-nature-com.proxy.queensu.ca/articles/srep03222.

[7] Songting Li et al. "Dendritic computations captured by an effective point neuron model". en. In: *Proceedings of the National Academy of Sciences* 116.30 (July 2019), pp. 15244–15252. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1904463116. URL: http://www.pnas.org/lookup/doi/10.1073/pnas.1904463116 (visited on 10/13/2021).

[8] David C. Van Essen, Chad J. Donahue, and Matthew F. Glasser. "Development and Evolution of Cerebral and Cerebellar Cortex". In: *Brain, behavior and evolution* 91.3 (Aug. 2018), pp. 158–169. ISSN: 1421-9743. DOI: 10.1159/000489943. URL: https://pubmed.ncbi.nlm.nih.gov/30099464/.

[9] David Beniaguev, Idan Segev, and Michael London. "Single cortical neurons as deep artificial neural networks". In: *Neuron* 109.17 (Sept. 2021), 2727–2739.e3. ISSN: 0896-6273. DOI: 10.1016/J.NEURON.2021.07.002.

[10] Terrence J. Sejnowski. "The unreasonable effectiveness of deep learning in artificial intelligence". en. In: *Proceedings of the National Academy of Sciences* 117.48 (Dec. 2020). Publisher: National Academy of Sciences Section: Colloquium Paper, pp. 30033–30038. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1907373117. URL: https://www.pnas.org/content/117/48/30033 (visited on 10/13/2021).

[11] Christof Koch and Idan Segev. "The role of single neurons in information processing". en. In: *Nature Neuroscience* 3.11 (Nov. 2000). Bandiera_abtest: a Cg_type: Nature Research Journals Number: 11 Primary_atype: Reviews Publisher: Nature Publishing Group, pp. 1171–1177. ISSN: 1546-1726. DOI: 10.1038/81444. URL: https://www.nature.com/articles/nn1100_1171 (visited on 10/13/2021).

[12] Guoping Feng et al. "Opportunities and limitations of genetically modified nonhuman primate models for neuroscience research". en. In: *Proceedings of the National Academy of Sciences* 117.39 (Sept. 2020). Publisher: National Academy of Sciences Section: Perspective, pp. 24022–24031. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2006515117. URL: https://www.pnas.org/content/117/39/24022 (visited on 10/13/2021).

[13] Constança Carvalho et al. "Ethical and Scientific Pitfalls Concerning Laboratory Research with Non-Human Primates, and Possible Solutions". In: *Animals : an Open Access Journal from MDPI* 9.1 (Dec. 2018), p. 12. ISSN: 2076-2615. DOI: 10.3390/ani9010012. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6356609/ (visited on 10/13/2021).

[14] Paula Duhatschek · CBC News ·. *Growing demand for mental health services linked to long wait times — CBC News*. en. Jan. 2020. URL: https://www.cbc.ca/news/canada/kitchener-waterloo/growing-demand-for-mental-health-services-linked-to-long-wait-times-1.5444702 (visited on 10/13/2021).

[15] Jerome C Wakefield. "The concept of mental disorder: diagnostic implications of the harmful dysfunction analysis". In: *World Psychiatry* 6.3 (Oct. 2007), pp. 149–156. ISSN: 1723-8617. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2174594/ (visited on 10/13/2021).

[16] Jason Brownlee. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. Jan. 2021. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (visited on 04/10/2022).

[17] Fleur Zeldenrust et al. "Estimating the information extracted by a single spiking neuron from a continuous input time series". In: *Frontiers in Computational Neuroscience* 11 (June 2017), p. 49. ISSN: 16625188. DOI: 10.3389/FNCOM.2017.00049/BIBTEX.

[18] Gianni Pola et al. "A Practical Guide to Information Analysis of Spike Trains". In: *Neuroscience Databases* (2003), pp. 139–154. DOI: 10.1007/978-1-4615-1079-6_10.

[19] SEDNA. *The Carbon Footprint of Carbon Copies: How Does Email Affect the Environment?* 2021. URL: https://sedna.com/blog/the-carbon-footprint-of-carbon-copies-how-does-email-affect-the-environment/ (visited on 04/10/2022).

[20] George Lawton and Ivy Wigmore. *What are AI Ethics (AI Code of Ethics)?* URL: https://www.techtarget.com/whatis/definition/AI-code-of-ethics (visited on 04/10/2022).