Neurons

BCI Signal Decoding

Raihan Abdul Vaheed
Jakeb Chouinard

October 14, 2024

1 Introduction

Neurons are Neat!

While not everyone has had an opportunity to gain experience using, developing, or testing Spiking Neural Networks, it is very likely that they have come into contact with the notion of *neurons* at some point in time. Neurons are the building-blocks of our Central Nervous System, and they can be found throughout the body – though non-brain neurons are often referred to as nerves. Neurons serve a variety of different purposes, but they all contribute in some way to our conscious experiences.

In this document, we would like to introduce you all to beurons in a basic way across their presences in medical science, traditional neural networks, and spiking neural networks. To begin, we'll consider biological aspects of neurons and their behaviour – this will provide a segue into the development of Perceptrons and Multi-Layer Perceptron models as well as more modern AI developments. From here, we will return more to biological considerations and see how spiking neural network models serve as a better analogue to brains and their internal processes.

This is largely intended to be an internal document. If you would like to distribute or share this document with anyone outside of the BCI Signal Decoding project team and/or WAT.ai Admin, please ask the TPMs for approval first.

2 What is a Neuron?

2.1 The Neuroscience Perspective

As mentioned in the Introduction, Neurons are the building-blocks of the Central Nervous System (CNS). The CNS is composed of a large variety of neurons, though there are three easily identifiable families [1]:

1. ***Sensory Neurons***
   These neurons are responsible for how we perceive our environment. This includes the rods and cones present in our eyes, nerves in the skin that communicate hot and cold, etc. Simply put, if it gathers sense information, it's a sensory neuron.

2. ***Motor Neurons***
   Motor neurons are responsible for us being able to interact with our environment. These include the nerves that extend from the brain to the spine and the nerves that extend from the spine to the muscles. If it's related to motion in some way, it's a motor neuron.

3. ***Interneurons***
   Connecting the sensory and motor neurons are interneurons. Interneurons can similarly connect with other interneurons and create complex circuits of neural behaviour. These neurons are typically associated with reflexes that do not require thought – such as a jerk in the event of a needle prick.

You may be thinking to yourself: none of these seem to be in the brain! And you'd be right for the most part. Brain neurons are significantly more difficult to be categorized, as their individual contributions to behaviour are not well-understood. Brain structures are innately complex in their architectures as well as their functions – one good example of this is the Cerebral Cortex. The Cerebral Cortex is the outer-most brain structure (the wrinkly one) and is composed of multiple layers of neurons that are complicatedly connected both vertically and laterally – see Figure 1[2]. Despite being the biggest and heaviest part of the human brain at approximately 80% of the brain's mass, the Cerebral Cortex accounts for just shy of 20% of the neurons present in a typical human brain – clocking in at *only* 100 billion neurons [3].

The basic structure of a neuron is in three parts: the dendrites, the soma, and the axon. Dendrites act as branching input channels for a neuron to receive some current input. The current makes its way to the soma (the nucleus-containing cell body) where the neuron stores its internal charge. Once a sufficient charge has been reached, the neuron emits a spike (or action potential) that travels out through the axon. Axons are sheathed in a substance known as myelin – myelin provides electrical insulation and ensures that signals can move between neurons without a significant amount of loss. Communication from one neuron's axon to other neurons' dendrite(s) is facilitated through synapses. Synapses can be either chemical or electrical, where chemical synapses rely on neurotransmitters like

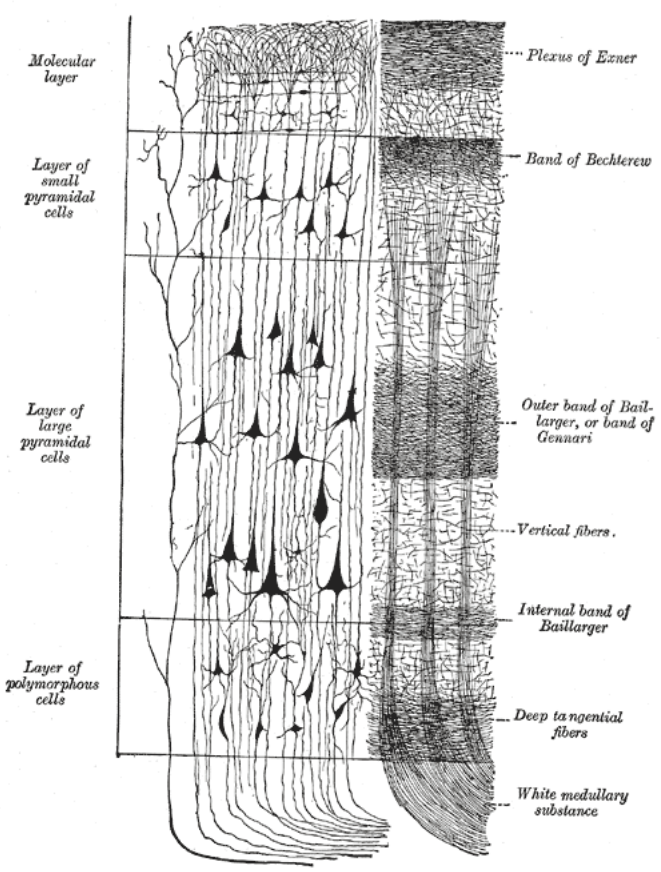


Figure 1: A Generic Cerebral Cortex Column [2]

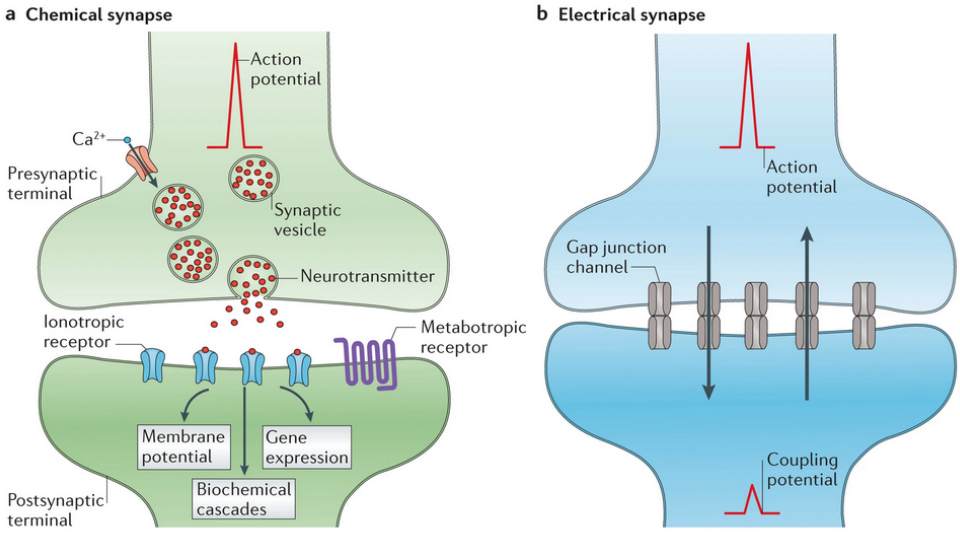
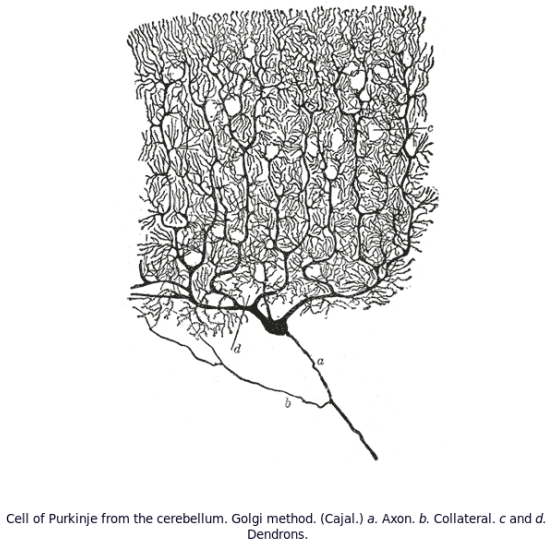


Figure 2: Generic Synapse Depictions [4]

dopamine, serotonin, GABA, etc., and electrical synapses couple action potentials. Figure 2 shows generic depictions of both synapse types.

Despite not having a terribly confusing structure in theory, neurons found in the human body can be egregiously complex in their connectivities. As an example, consider a Purkinje cell typically found in the Cerebellum (shown in Figure 3. Purkinje cells have extremely dense and flat dendrite branches, allowing them to perform complex computations over an exceedingly large input space. Further, imagine this cell connected to tens of thousands of other similarly-complex cells all localized in the same place – how could you even determine what any one cell corresponds to? The pure variety in neurons alone makes it a uniquely difficult challenge to understand how everything is connected – much less understand the brain.

One important clarification: **neurons are not the only cells present in the brain**. The brain is full of *glia* as well as neurons. Glia are currently the subject of ongoing neuroscience research to determine exactly what their functions are and how glia malfunctions can impact brain processes – leading or contributing to diseases such as Alzheimer's Disease, Parkinson's Disease, and multiple sclerosis [5]. While glia is thought to play a role in the maintaining of proper brain function, it is not widely believed that they are active participants in cognitive processes – this job belongs to neurons. Hence, glia do not typically appear as a consideration for computational modeling of neurobiological systems.



Cell of Purkinje from the cerebellum. Golgi method. (Cajal.) *a.* Axon. *b.* Collateral. *c* and *d.* Dendrons.

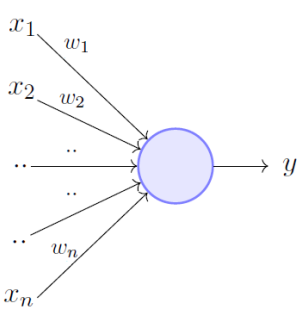
Figure 3: Purkinje Cell Diagram [2]

2.2 The AI Engineering Perspective

The advent of artificial neural networks dates back to 1943 between professors at University of Illinois at Chicago and University of Chicago – Warren McCulloch and Walter Pitts respectively. The first hardware implementation did not arrive until 1957 courtesy of Frank Rosenblatt – a professor at Cornell University. This early implementation was known as a "Perceptron". Rosenblatt's Perceptron was a supervised-learning approach to solving arbitrary classification tasks. The basic architecture involved the weighted summation of values acting as an input offset by some bias factor. The output was then binarized based on if the weighted input met the threshold that the bias factor implemented. This is shown in Figure 4. Based on the observed output of a training trial, Rosenblatt would adjust the weights of the inputs by varying potentiometer resistances. Multiple of these networks implemented successively was known as a Multi-Layer Perceptron, and Rosenblatt believed strongly that an MLP was capable of performing any arbitrary classification task, though other scholars of the time disagreed. In the early 60s, Rosenblatt also aided in the development of a simplified version of back-propogation – a generalizable way to calculate the weighting changes needed to be made to improve performance [6].

The Perceptron is generally understood as one very specific implementation of a neural network based on the McCulloch-Pitts neuron model, though its implementation is slightly more computational than their original model.

Typical artificial networks still follow a similar basic architecture to the Perceptron neuron model: a linear weighting of the input values from one or many neurons, an activation function, and an output to one or many neurons. Layers are commonly *densely connected* – one neuron in the N^{th} layer outputs to **all** the neurons in the N^{th+1} layer. There is also



$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

Figure 4: Simple Perceptron Neuron [7]

the potential for *sparsely connected* layers where one neuron in the N^{th} layer outputs to **some** of the neurons in the N^{th+1} layer. There are also *skip* and *recursive* connections where a neuron in the N^{th} layer outputs to some $N^{th+\alpha}$ $(\alpha > 1)$ layer or to itself.

In these more traditional layer structures, activation functions can vary significantly. The most common activation function is *ReLU*, with *Sigmoid* and *tanh* being similarly prevalent. ReLU – **Re**ctified **L**inear **U**nit – performs a weighted summation of its inputs and compares it to 0. If the weighted summation is less than 0, then the output of the neuron is 0, otherwise the output is the weighted summation value:

$$y_i = \begin{cases} 0 & \sum_{j=0}^{n} w_{ji} y_j + b_i \leq 0 \\ \sum_{j=0}^{n} w_{ji} y_j + b_i & \text{else} \end{cases}$$

The resultant behaviour is that of $y = x$ – a linear output – for any input value above zero and that of $y = 0$ for any other value. This introduces a clear non-linearity in the neuron's output; however, this is not a bad thing. By introducing internal non-linearity, we can increase the neural network's ability to calculate non-linear relationships between inputs and outputs of the model. Sigmoid maps the output to a $(0, 1)$ range using an inverse decaying exponential. This behaviour can be useful for classification tasks where the categorization is binary. Tanh maps the output to a $(-1, 1)$ range using a combination of exponentials – this is generally used in hidden layers when provided data is zero-meaned. Tanh's zero-mean makes it particularly apt in handling zero-meaned data. Their definitions are show below where x is the weighted input from the previous layer:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

There are also a few more specialized layers, such as *convolutional* and *pooling* layers. Convolutional layers aim to extract spatially patterned information from previous layers – depending on its level in the hierarchy, this patterned information could be specially-curved edges or entire shape silhouettes. Pooling layers are generally used immediately

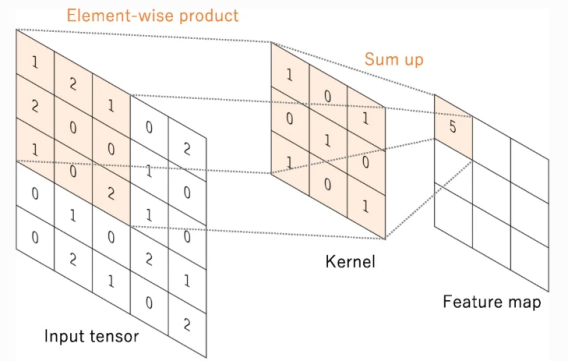


Figure 5: Basic Example of a Convolutional Layer Operation Given a 5x5 Tensor, a 3x3 Kernal, and Stride 1 [8]

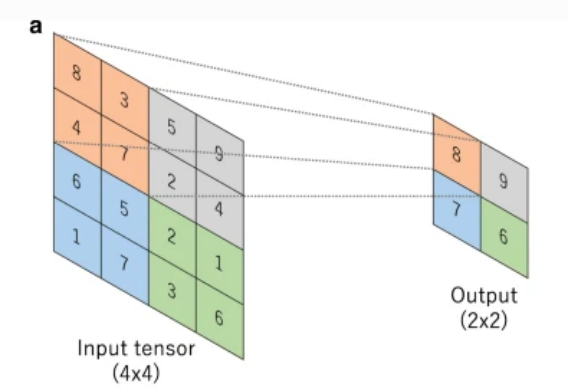


Figure 6: Basic Example of a Maximum-Pooling Layer Operation Given a 4x4 Tensor, a 2x2 Filter, and Stride 2 [8]

after a convolutional layer in an attempt to consolidate the features that the different convolutional kernals extracted. This can similarly help reduce the effects of localization that convolutional layers can induce. Pooling can be done by extracting the most prevalent feature over a patch of the convolutional layer (*maximum-pooling*) or by averaging the presence of features over a patch of the convolutional layer (*average-pooling*). A simple example of a convolutional layer operation is shown in Figure 5. Similarly, a simple example of maximum-pooling is shown in Figure 6.

More complex structures can be implemented using these layers as building-blocks, such as LSTMs, GRUs, Auto-Encoders, etc.. If you are interested in learning more about those specific implementations, I encourage you to investigate them further and ask questions as needed. :)

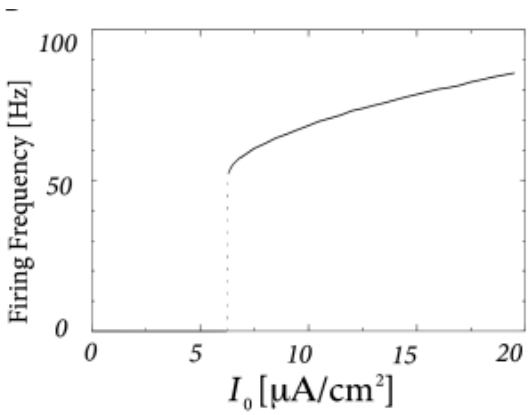


Figure 7: Hodgkin-Huxley Estimated Firing Rates for a Large Squid Axon [9]

2.3 The Neural Engineering Perspective

One intuition when we compare the Neuroscience and AI Engineering perspectives is that the AI Engineering "neurons" don't do a great job at capturing more biological aspects of neurons. Sure, neurons observe an increased firing rate with stronger stimulus, but it isn't a linear increase. On top of that, it isn't an infinite increase – there are biological limitations to the rate at which neurons can spike (release action potential. Similarly, if we consider a layer like maximum-pooling, it is impossible to trivially implement something that executes a $\max(x)$ function over several patches of some representational space – we have an entire brain structure dedicated to action selection, in fact (Basal Ganglia), and that is effectively just an implementation of a $\max(x)$ (I am unintentionally making the Basal Ganglia sound a lot less cool than it is; it is involved in all sorts of cool brain stuff, not just a trivial max pool).

We can account for some of these intuitive issues by looking for a better model by which to implement neurons. One option is the Hodgkin-Huxley Model. The Hodgkin-Huxley model implements three different state parameters to simulate the behaviour of a neuron based on its neurotransmitter interactions. This is based on analysis of equilibrium-pursuit at synaptic clefts when neurotransmitters are released – particularly Sodium-Potassium interactions. Figure 7 shows a tuning plot based on the Hodgkin-Huxley model where parameters were chosen based on the neurobiology of a large squid axon. [9]

We do see an approaching saturation in the Hodgkin-Huxley plot shown in Figure 7, though the stark discontinuity at the point of stimulus increase is a little bit concerning. An additional potential problem is that the complex dynamic relationship between the equilibrium variables that would be needed for each neuron in a network results in a significant amount of parameters needing to be taken into consideration and computed upon.

Instead of focusing on the neurotransmitters, we can take a more abstracted approach by just looking at the electrical signals being passed around neurons. The input to any

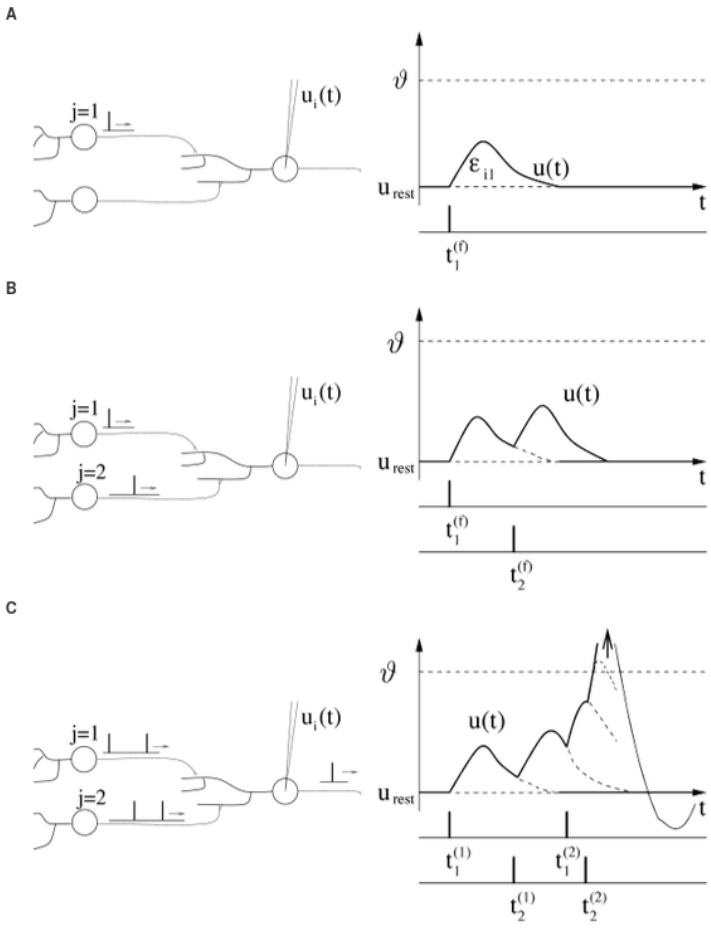


Figure 8: Post-Synaptic Current

neuron's dendrites is called a post-synaptic potential (A in Figure 8). If multiple post-synaptic potentials are provided to a neuron, they are superimposed as demonstrated in B and C of Figure 8. Differing from the Hodgkin-Huxley mode, the synapses are generally modeled as low-pass filters that apply to pre-synaptic potentials. Post-synaptic potentials accumulate in a neuron's soma until a voltage threshold is reached, causing a reset in voltage and an action potential to travel down its axon. This accumulation is modeled as an integration process with some leaking behaviour, as once a neuron's input is removed, its soma's potential returns to a baseline potential.

A simple implementation of this behaviour is achieved by the Leaky Integrate-and-Fire model. This model treats the approximates the soma as a parallel RC circuit as shown in Figure 9. Measuring the voltage in the soma as the voltage across the neuron's capacitor, an application of Kerchoff's Current Law provides the following:

$$C_n \frac{dv_n(t)}{dt} = \frac{v_{\text{rest}} - v_n(t)}{R} + J_{in}, \quad \text{iff}: \quad v_n(t) < v_{\text{th}}$$

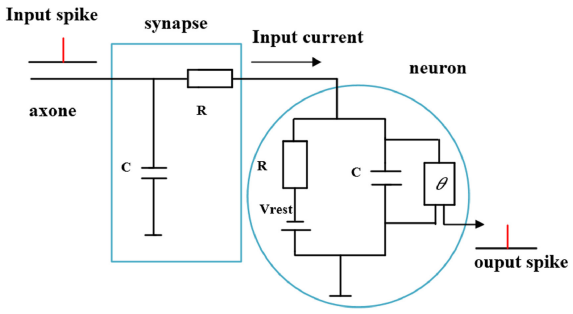


Figure 9: LIF Circuit Exquivalence Diagram [10]

Some simplifying assumptions can be made: $v_{\text{rest}} = 0$ and $v_{\text{th}} = 1$. Similarly, a simplifications of notation can be used: $C_n R = \tau_{RC}$. These result in:

$$\frac{dv_n(t)}{dt} = \frac{1}{\tau_{RC}} \left(R J_{\text{in}} - v_n(t) \right), \quad \text{iff}: \quad v_n(t) < 1$$

This effectively describes the behaviour of the voltage in the soma when it does not exceed the threshold voltage. When the voltage meets the threshold voltage, the voltage becomes a delta-dirac function centered at the time of firing, t_f: $\delta(t_f)$. For those unfamiliar, the delta-dirac function is an infinite spike when its input zero and is 0 elsewhere. Immediately after the spike, the voltage resets to its reset value (assumed 0 in our case) and stays at 0 until the absolute refractory period is finished. In this period, the neuron is unable to fire or increase in voltage due to the limited amount of time since the previous spike. The duration of this period is notated as τ_{ref}.

Without going into an egregious number of derivations and mathematical details, we can approximate the number of spikes that a neuron would exhibit per second under a constant input as follows:

$$G(J) = \begin{cases} \frac{1}{\tau_{\text{ref}} - \tau_{\text{RC}} \log\left(1 - \frac{1}{J}\right)} & J > 1 \\ 0 & \text{else} \end{cases}$$

This provides a metric by which we can qualitatively analyze neural behaviour – and it's not terribly difficult to do so. I would challenge anyone who things this is difficult to perform the same sort of simplification for the Hodgkin-Huxley model.

3 Summary

We've seen the different ways that people have characterized neurons in the past as well as the direction that we can work towards. The exact means by which we will use this information may not be obvious yet, but we will discuss things more in depth as we move forward. Keep an eye out for any resources or packets of information that we'll be sending your way.

All the best,

Jake and Rai

References

[1] *Types of neurons*, 2022. [Online]. Available: https://qbi.uq.edu.au/brain/brain-anatomy/types-neurons.

[2] H. Gray and W. H. Lewis, *Anatomy of the human body*. Philadelphia, Lea and Febiger, 1918, 1918, p. 1396, https://www.biodiversitylibrary.org/bibliography/20311 — First ed. published in London in 1858 under title: Anatomy, descriptive and surgical. [Online]. Available: https://www.biodiversitylibrary.org/item/60234.

[3] S. Herculano-Houzel, "The human brain in numbers: A linearly scaled-up primate brain," en, *Front. Hum. Neurosci.*, vol. 3, p. 31, Nov. 2009.

[4] A. E. Pereda, "Electrical synapses and their functional interactions with chemical synapses," *Nature Reviews Neuroscience*, vol. 15, no. 4, pp. 250–263, Apr. 2014, ISSN: 1471-0048. DOI: 10.1038/nrn3708. [Online]. Available: https://doi.org/10.1038/nrn3708.

[5] Dana Foundation, *Cells of the brain*, 2023. [Online]. Available: https://dana.org/resources/cells-of-the-brain-grades-9-12/.

[6] B. Copeland, *Perceptrons*, 2024. [Online]. Available: https://www.britannica.com/technology/perceptrons.

[7] A. Chandra, *Perceptron learning algorithm: A graphical explanation of why it works*, 2022. [Online]. Available: https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975.

[8] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. [Online]. Available: https://doi.org/10.1007/s13244-018-0639-9.

[9] R. N. Wulfram Gerstner Werner M. Kistler and L. Paninski, *Neuronal dynamics*, 2014. [Online]. Available: https://neuronaldynamics.epfl.ch/online/.

[10] G. Zhang, B. Li, J. Wu, *et al.*, "A low-cost and high-speed hardware implementation of spiking neural network," *Neurocomputing*, vol. 382, pp. 106–115, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.11.045. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219316479.