A Formal introduction to Neural Networks: An Attempt to explain Whole Brain Emulation and Conscious Systems from Scratch

Sakshi Gupta, Abinesh Mathivanan

June 2022

1 Introduction

Brain simulation is the concept of creating a functioning computer model of a brain or part of a brain. Brain simulation projects intend to contribute to a complete understanding of the brain, and eventually also assist the process of treating and diagnosing brain diseases. Various simulations from around the world have been fully or partially released as open source, such as C. elegans, and the Blue Brain Project Showcase. In 2013 the Human Brain Project, which has utilized techniques used by the Blue Brain Project and built upon them,created a Brain Simulation Platform (BSP), an internet-accessible collaborative platform designed for the simulation of brain models. Simulation also aims to replicate work on animal models, such as the mouse. In addition, the computing environment used for simulation offers the possibility of studying disease processes electronically. Richard Feynman famously said, "What I cannot create, I do not understand." To truly understand the brain we need tools to create it, in brain atlases, computer models, and simulations. The platforms to be delivered are for Neuroinformatics, Medical Informatics, Brain Simulation, High Performance Computing, Neuromorphic Computing and Neurorobotics—each to be open for use by the global research community. These platforms are designed to bring together data about the brain, integrate it in unifying brain models, run simulations, analyze and visualize the results, and test hypotheses. The project aims to trigger a global, collaborative effort to understand the human brain, while enabling advances in neuroscience, medicine, and future computing. The primary objective is to provide the capability to build and simulate models of the entire human brain within ten years. The central question in next-generation artificial intelligence (AI) and developmental robotics is how to build an integrative cognitive system capable of lifelong learning and human-like behavior in various environments such as homes, offices, and outdoors. In this research, inspired by the whole brain architecture (WBA) approach, using a whole brain probabilistic generative model (WB-PGM), we introduce the idea of building an integrative cognitive system that can alternatively be referred to as artificial general intelligence. Adjacent research areas include biologically inspired cognitive architectures and cognitive computational neuroscience, which is an interdisciplinary field of cognitive science and computational neuroscience.

2 Abstract

Whole brain emulation (WBE) is the possible future one-to-one modeling of the function of the entire (human) brain. However, it is difficult for an individual to design a software program that corresponds to the entire brain because the neuroscientific data required to understand the architecture of the brain are extensive and complicated. The basic idea is to take a particular brain, scan its structure in detail, and construct a software model of it that is so faithful to the original that, when run on appropriate hardware, it will behave in essentially the same way as the original brain. This would achieve software-based intelligence by copying biological intelligence (without necessarily understanding it). The vastness of the design space created by the combination of a large number of computational mechanisms, including machine learning, is an obstacle to create an Artificial General Intelligence (AGI). The whole-brain architecture approach divides the brain-inspired AGI development process into the task of designing the brain reference architecture (BRA) - the flow of information and the diagram of corresponding components - and the task of developing each component using the (BRA). This is called BRA-driven development. Recently, proponents of WBE have suggested that it will be realized in the next few decades. In this paper, we investigate the plausibility of WBE being developed in the next 50 years (by 2063). We consider the cultural and social effects of WBE. We find the two most uncertain factors for WBE's future to be the development of advanced

miniscule probes that can amass neural data in vivo and the degree to which the culture surrounding WBE becomes cooperative or competitive. We identify four plausible scenarios from these uncertainties and suggest the most likely scenario to be one in which WBE is realized, and the technology is used for moderately cooperative ends.

3 Literature Review

Beyond the process of moving to other substrates, we are especially interested in enhancement of the mind. There are numerous technological proposals for the accomplishment of SIM. At present count, there are at least six main tracks. Our understanding is still limited with regards to the manner in which fundamental computational elements of the brain participate in the vast interaction of concurrent processes from which mind emerges. That approach is to faithfully re-implement by emulating the basic computational functions carried out by elements of the neurophysiology, while at the same time faithfully re-implementing the connectivity as it exists between those elements in the neuroanatomy. The problem is decomposed into much smaller physical pieces for which so-called system identification must be feasibly carried out. At that level, there are still suppositions about scope and resolution that need to be tested. E.g., do ensembles of neurons, individual spiking neurons, morphologically detailed neurons, or molecular processes in synaptic channels attain the requisite resolution for emulation? Still, the acquisition and recreation of function and structure are feasible with understanding at a level within reach of current neuroscience, and with tools that we can construct today. In 2000, I named this approach descriptively as Whole Brain Emulation (WBE). The term caught on and was eventually adopted by related research aspirations, where it is sometimes abbreviated to "brain emulation" if the whole brain is not the scope of a project. The core idea, that the significant aspects of a person's mental life can persist when properly transferred from body to machine, from machine to machine, or from body to body has been around for a long while. In those early forms, it was a fancy and a fantasy, the purview of magical transformations. With the scientific renaissance, modern philosophy and psychology, thought experiments along the same lines became more refined, and by the middle of the 20th century, serious science fiction writers were incorporating in their stories some ideas that quite closely resemble the current conceptions of mind uploading, whole brain emulation and substrate-independent minds. In the early 1990's, the rise of the Internet began to facilitate the formation of on-line interest groups and communities that would otherwise have had a very difficult time finding their peers on a global scale. One of these interest groups coalesced around the concept of mind uploading. A collection of web pages was maintained by Joe Strout, which was most notable for practical ideas about the reconstruction of brains by building compartmental models of neurons from structural scans. Joe Strout, at UCSD at the time, also established a mailing list called the mind uploading research group (MURG). There were some early fellow travelers with the same destination, who understood the difference between impossible projects and ambitious projects, and who were willing to devise practical plans and dedicate their efforts to the necessary actions. In 2007, the Future of Humanity Institute at Oxford University, and in particular Nick Bostrom and Anders Sandberg (a former computational neuroscientist) began to take a serious interest in Whole Brain Emulation. The first Whole Brain Emulation Workshop was convened at Oxford University. The result of this workshop was a technical report on the feasibility of WBE, a first attempt at a roadmap of sorts (A. Sandberg N. Bostrom, 2008). The report already included key technologies such as the Knife-Edge Scanning Microscope (KESM), the Automatic Tape-Collecting Lathe Ultramicrotome (ATLUM, now called an ATUM) and functional recording work by Peter Passaro. In 2010, WBE was for the first time included in the annual conference on Artificial General Intelligence (Lugano, Switzerland). As of this writing, most of those working on SIM are focused on research and tool development aimed at the initial challenge to gain sufficient access to the biological human brain. In those efforts, we now include the world-class expertise of Jeff Lichtman, Ted Berger, Henry Markram, Sebastian Seung, Ed Boyden, George Church, Anthony Zador, Konrad Kording, Clay Reid, their laboratories and many others.

4 Methodolgy

Iteratively, we can determine that "sweet spot" where our ability to solve a collection of connected and individually tractable system identification problems meets our ability to build new tools for high-resolution measurements. At that point, brain emulation at the scale of a human brain is a feasible project. We can categorize areas within a roadmap toward whole brain emulation according to four main pillars:

- 1. Hypothesis testing: iteratively evaluating proofs-of-concept on our way to the sweet spot.
- 2. Structure: the decomposition of the system identification problem into many smaller problems, largely by gathering so-called "connectome" data.

- 3. Function: characterizing each system, an area with tool-development needs that are addressed, for example, in the BRAIN Initiative (Obama, 2013).
- 4. Emulation: the mathematical representations and computational platforms needed.

We can begin a formal description of mental processes, while taking care not to be overly restrictive about the underlying mechanisms to be considered. After-all, coming up with a satisfactory model or theory is an iterative process that is based both on conceptualization and on data evaluation. We can make some initial assumptions based on the presence or lack of certain evidential data at this time. For example, we might assume that the brain relies on particular biophysical mechanisms, which should be modeled to adequately predict and replicate the processes of the mind. Once we have a model of processes that act on signals, and once we acknowledge that boundaries are drawn around sub-systems in some way, then, when we focus on a specific set of sub-systems, we can identify the boundaries between them. The description can include hysteresis (memory) in the sub-system. The most promising results in high resolution connectome data are produced through volume microscopy in which electron microscope images are taken at successive ultra-thin layers of brain tissue. In electron micrographs at a resolution of 5-10nm it is possible to identify individual synapses and to reconstruct the 3D geometry of cell bodies of individual neurons with the detailed morphology of axon and dendrite branches. As for the functional data needed from each small sub-system, the most promising tool development is taking its inspiration from the brain's own approach: detection at close range in physical proximity to sources of interaction, namely via microscopic synaptic receptor channels. The brain handles a tremendous quantity of information by utilizing a vast hierarchy of such receptor connections. Similarly, to satisfy the temporal and spatial resolution requirements for in-vivo functional characterization, investigative tool development is looking primarily at ways to take the measurements from within. Modeling of thought processes is necessary for whole brain emulation and can be beneficial to efforts in AGI, but the goals and therefore the success criteria are different: AGI is successful if it manages to capture the general principles of a mind to the point where a machine can achieve a desired level of performance for a spectrum of possible tasks. Due to this difference, there will be points at which the level of investigation in biological brains will be chosen differently to best suit each goal. Another important realization is that both work on artificial intelligence and on WBE are mainly evaluated in terms of performance, and neither necessarily implies a full understanding of human intelligence. That said, whole brain emulation can provide readily accessible working mind functions that may rapidly facilitate insight and understanding.

5 Mathematical Modelling

Mathematical modeling of brain bioelectricity is a multidisciplinary science sharing contributors from both biomedical and engineering fields. This tutorial aims to help those newly introduced to the subject from both fields to grasp a global view of it, focusing on main concepts rather than on details. The article should help the reader understand why different modeling approaches are used in different applications and understand their limitations. This is especially important since the dominance and widespread of ready-made commercial/ open-source software packages has tempted many researchers to ignore dwelling into the mathematical basis of these packages and simply be content with using them. An old adage among computer scientists, however, states that ”garbage in -garbage out”. Motivated by this, the present article aims to help users of mathematics, as well as developers, recall essential concepts that can help them make wise use of existing models and related tools. The model problem of supervised learning which we focus on in this article can be formulated as follows: given a dataset

$$S = (x_i, y_i = f(x_i)), i[n], \tag{1}$$

approximate f as accurately as we can. If f takes continuous values, this is called a regression problem. If f takes discrete values, this is called a classification problem. We will focus on the regression problem. For simplicity, we will neglect the so-called “measurement noise” since it does not change much the big picture that we will describe, even though it does matter for a number of important specific issues. We will assume

$$x_i \in X := [0,1]d \tag{2}$$

and we denote by P the distribution of x_i.

We also assume for simplicity that

$$x \in X|f(x)| \leq 1 \tag{3}$$

The standard procedure for supervised learning is as follows:

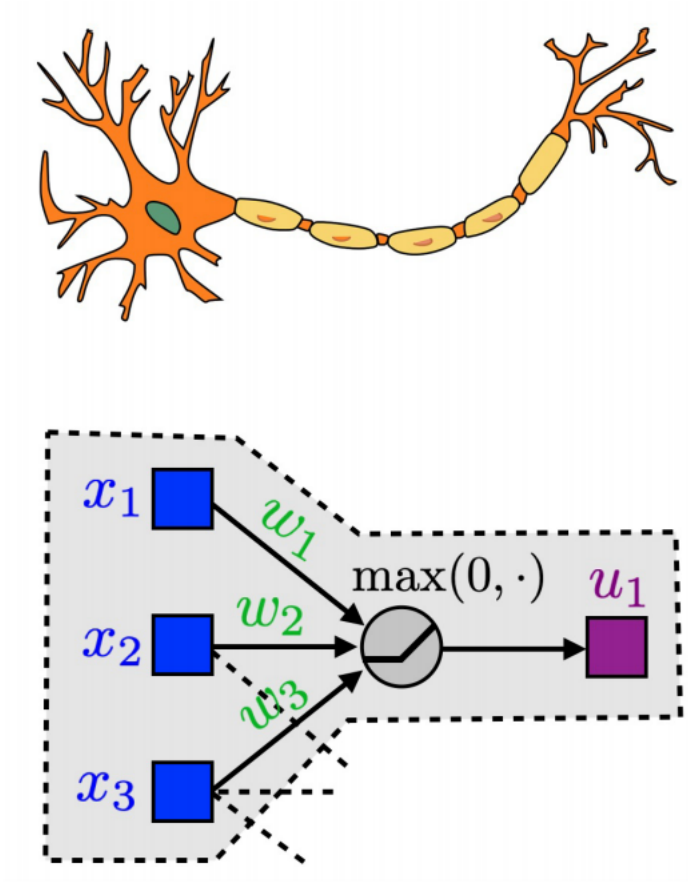

Figure 1: Structure of a Neural Network

- 1. Choose a hypothesis space, the set of trial functions, which will be denoted by Hm. In classical numerical analysis, one often uses polynomials or piecewise polynomials. In modern machine learning, it is much more popular to use neural network models. The subscript m characterizes the size of the model. It can be the number of parameters or neurons, and will be specified later for each model.
- 2. Choose a loss function. Our primary goal is to fit the data. Therefore the most popular choice is the "empirical risk":
$$R^n(f) = 1nX_i(f(x_i)y_i)2 = 1nX_i(f(x_i)f(x_i))2 \tag{4}$$
Sometimes one adds some regularization terms.
- 3. Choose an optimization algorithm and the hyper-parameters. The most popular choices are gradient descent (GD), stochastic gradient descent (SGD) and advanced optimizers such as Adam, RMSprop. The overall objective is to minimize the "population risk", also known as the "generalization error"
$$R(f) = E_x\ p(f(x)f(x))2 \tag{5}$$
- 4. "When we see a large function, we can see that it's composed of smaller functions and have some intuition about what the solution can be," Lample said. "We think the model tries to find clues in the symbols about what the solution can be." He said this process parallels how people solve integrals — and really all math problems — by reducing them to recognizable sub-problems they've solved before. Despite the results, the mathematician Roger Germundsson, who heads research and development at Wolfram, which makes Mathematica, took issue with the direct comparison. The Facebook researchers compared their method to only a few of Mathematica's functions —"integrate" for integrals and "DSolve" for differential equations — but Mathematica users can access hundreds of other solving tools.

A neuron is represented in the figure (1). It is the first neuron, the one that calculates the first value u1 which composes the layer u. This neuron connects a certain number of elements of the first layer (here three: x$_1, x_2, x_3$

but there can be more) to a single element of the second, so here u. The formula calculated by the neuron is

$$u_1 = max(w_1x_1 + w_2x_2 + w_3x_3 + w_4; 0) : \tag{6}$$

The neuron thus performs a weighted sum of the three inputs, with three weights $w_1; w_2; w_3$, and we also add w_4, which is a bias. Then the neuron calculates the maximum between this sum and zero. One can also use a function other than the maximum function, but this one is the most popular. It is a thresholding operation. We can compare it to biological neurons which let or not pass information according to whether they are sufficiently excited or not. So if the weighted sum

$$w_1x_1 + w_2x_2 + w_3x_3 + w_4 \tag{7}$$

is smaller than 0, then the neuron returns the value $u_1 = 0$, otherwise it returns the value of this sum and places it in u_1

5.1 Creating a Neural Network

- Step 1: For each input, multiply the input value x_i and w_i with weights and sum all the multiplied values. Weights — represent the strength of the connection between neurons and decides how much influence the given input will have on the neuron's output. If the weight w_1 have a higher value than the weight w, then the input x_1 will have a higher influence on the output than w_2.

$$\Sigma = (x_1 * w_1) + (x_2 * w_2) + ... + (x_n * w_n) \tag{8}$$

The row vectors of the inputs and weights are x = $[x_1, x_2, \ldots, x_n]$ and w =$[w_1, w_2, \ldots, w_n]$ respectively and their dot product is given by

$$x.w = (x_1 * w_1) + (x_2 * w_2) + ... + (x_n * w_n) \tag{9}$$

Hence, the summation is equal to the dot product of the vectors x and w

$$\Sigma = x.w \tag{10}$$

Step 2: Add bias b to the summation of multiplied values and let's call this z. Bias — also known as the offset is necessary in most of the cases, to move the entire activation function to the left or right to generate the required output values.

$$z = x.w + b \tag{11}$$

Step 3.Pass the value of z to a non-linear activation function. Activation functions — are used to introduce non-linearity into the output of the neurons, without which the neural network will just be a linear function. Moreover, they have a significant impact on the learning speed of the neural network. Perceptrons have binary step function as their activation function. However, we shall use sigmoid — also known as logistic function as our activation function.

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{12}$$

where σ denotes the sigmoid activation function and the output we get after the forward prorogation is known as the predicted value ŷ.

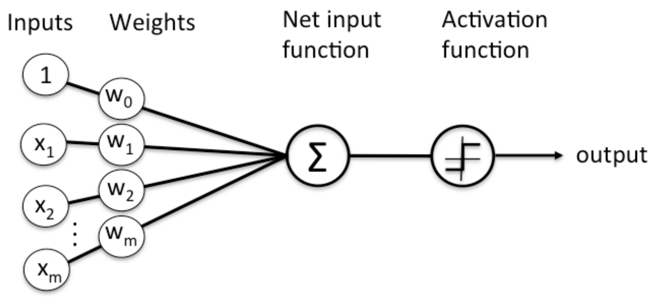

Fig 2: Representation of Neural network

Optimization: Optimization is the selection of the best element from some set of available alternatives, which in our case, is the selection of best weights and bias of the perceptron. Let's choose gradient descent as our optimization algorithm, which changes the weights and bias, proportional to the negative of the gradient of the cost function with respect to the corresponding weight or bias. Learning rate (α) is a hyperparameter which is used to control how much the weights and bias are changed. The weights and bias are updated as follows and the backpropagation and gradient descent is repeated until convergence.

$$w_i = w_i - (\alpha * \frac{\delta C}{\delta w_i}) \tag{13}$$

$$b = b - (\alpha * \frac{\delta C}{\delta b}) \tag{14}$$

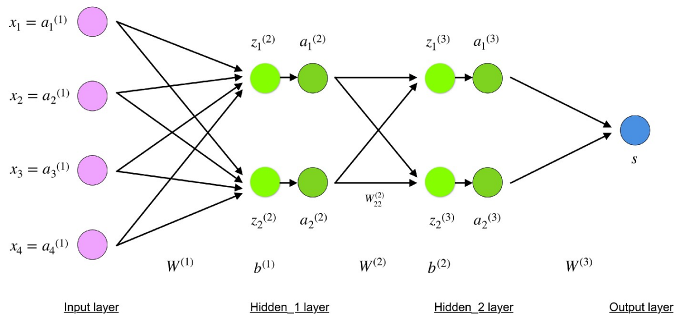

Fig 3: Representation a Backpropagation Neural network

Here we have the following entities:

- 1.1. Data — Here we are going to use the famous IRIS dataset. Here is the link to the dataset
- 2. Network — This network is a fully connected network where each neuron is connected to the other. This network has 6 layers including the input and output layers.
- 3. Layers — The first layer is the input layer. In the current architecture, we have 6 inputs. Let us call them x$_1, x_2, x_3, x_4, x_5$ and x$_5$. The next 4 layers are the hidden layers with 6, 6, 4 and 5 neurons each. The last layer is the output layer which has only one neuron. The output of this neuron is going to be the probability of the output variable 'y' being 1 or 0.
- 4. Weights — Each neuron has a set of weights associated to it. The nomenclature of each weight is defined by the input neuron in its layer, the current neuron in its layer and the layer.
- 5. Biases — Each neuron has a bias associated to it. For simplicity, I have assumed bias = 0 for all the neurons.
- 6.The nomenclature for inputs to a neuron is:

  $$A_x^y \tag{15}$$

  Here x is the layer and y is the neuron in its layer.
- 7.The nomenclature for the weights of a neuron is:

  $$W_{ab}^x \tag{16}$$

  Here x is the layer, a b are the neuron in their respective layers.
- 8.The nomenclature for the activation function is:

  $$\sigma_y^x \tag{17}$$

  Here y is the layer and x is the neuron in its layer. Having mentioned that, we have defined activation functions at the level of a layer and not at the level of a neuron.

What if the key to unlocking artificial general intelligence is a pattern that already exists, but has yet to be discovered? Artificial general intelligence, also referred to as “strong artificial intelligence” or “full artificial intelligence” is the ability of a machine to perform human-like cognition. What seems to be a straight-forward philosophical question is actually quite nuanced. Clues to the answer can be found in an interdisciplinary examination of computer science, math, philosophy, physics, and neuroscience.

Artificial intelligence (AI) is a term that lacks a single unifying definition. The simplest explanation is that AI is machine intelligence versus biological human intelligence. AI is in the early stages of development, despite being a concept that is over 60 years old—the term originated in a Darmouth Publication circa 1955. The resurrection of AI is largely due to recent trends such as the falling cost of computing, the rise of powerful cloud-based decentralized computing, the availability of big data for machine learning, and the increasing sophistication of computing algorithms. Today computer science technology enables machines to perform functions such as problem solving, learning, planning, reasoning, and recognition of speech, voice, images, and handwriting. Currently AI is more of a tool for point solutions—far from strong artificial intelligence.

Is math merely discovered, like an excavation by an archeologist on a dig, or invented, like an inspired poet? Mathematical Platonism is a metaphysical view that mathematical truths are discovered, not invented mathematical objects are abstract and exist independently of our having the ability to think or describe it. Metaphysics is a branch of philosophy that is concerned with the fundamental nature of reality and being, which includes ontology (the study of the nature of existence), cosmology (the study of the origin and evolution of the universe), and epistemology (the study of knowledge and justified beliefs). If an object has an associated mathematical formula, then it is theoretically possible to express it in a computer algorithm. If math is a reality unto itself that awaits identification, would that imply that everything has a corresponding mathematical formula? Critics of Mathematical Platonism would argue that numbers are concepts that exist when the mind conceives of them. Technological singularity is the concept where machine intelligence exceeds the capability of human intelligence. If this can be achieved, what does this mean for the future of humanity? The answer to this question has profound implications for the future. Whether or not the universe is inherently mathematical, humans are advancing toward unlocking the mysteries of physics, consciousness, artificial intelligence, neuroscience, and life itself.

6 Programming

There are two ways to create a neural network in Python:

- 1) From Scratch – this can be a good learning exercise, as it will teach you how neural networks work from the ground up
- 2) Using a Neural Network Library – packages like Keras and TensorFlow simplify the building of neural networks by abstracting away the low-level code

If you’re already familiar with how neural networks work, this is the fastest and easiest way to create one. No matter which method you choose, working with a neural network to make a prediction is essentially the same:

- . Import the libraries. For example: import numpy as np
- 2. Define/create input data. For example, use numpy to create a dataset and an array of data values.
- 3. Add weights and bias (if applicable) to input features. These are learnable parameters, meaning that they can be adjusted during training.
  - 1) Weights = input parameters that influences output
  - 2) Bias = an extra threshold value added to the output
- 4. Train the network against known, good data in order to find the correct values for the weights and biases.
- 5. Test the Network against a set of test data to see how it performs.
- 6. Fit the model with hyperparameters (parameters whose values are used to control the learning process), calculate accuracy, and make a prediction.

If you’re just starting out in the artificial intelligence (AI) world, then Python is a great language to learn since most of the tools are built using it. Deep learning is a technique used to make predictions using data, and it heavily relies on neural networks. Today, you’ll learn how to build a neural network from scratch. In a production setting, you would use a deep learning framework like TensorFlow or PyTorch instead of building your own neural network. That said, having some knowledge of how neural networks work is helpful because you can use it to better architect your deep learning models.

6.1 Creating the Neural Network Class

Now you know how to write the expressions to update both the weights and the bias. It's time to create a class for the neural network. Classes are the main building blocks of object-oriented programming (OOP). The Neural Network class generates random start values for the weights and bias variables. When instantiating a Neural Network object, you need to pass the . You'll use predict() to make the prediction. The methods

```
    _compute_derivatives() and _update_parameters()
```

have the computations you learned in this section. This is the final Neural Network class

```
    class Neural Network:
        def__init__(self, learning_rate)
        self.weights = np.array([np.random.randn(),
np.random.randn()])
self.bias = np.random.randn()
self.learning_rate = learning_rate
def _sigmoid(self, x):
return 1 / (1 + np.exp(-x))
def _sigmoid_deriv(self, x):
return self._sigmoid(x) * (1 - self._sigmoid(x))
def predict(self, input_vector):
layer_1 = np.dot(input_vector, self.weights) + self.bias
layer_2 = self._sigmoid(layer_1)
prediction = layer_2
return prediction

def _compute_gradients(self, input_vector, target):
layer_1 = np.dot(input_vector, self.weights) + self.bias
layer_2 = self._sigmoid(layer_1)
prediction = layer_2
derror_dprediction = 2 * (prediction - target)
dprediction_dlayer1 = self._sigmoid_deriv(layer_1)
dlayer1_dbias = 1
dlayer1_dweights = (0 * self.weights) + (1 * input_vector)
derror_dbias = (
derror_dprediction * dprediction_dlayer1 * dlayer1_dbias
)
derror_dweights = (
derror_dprediction * dprediction_dlayer1 *
dlayer1_dweights
)
return derror_dbias, derror_dweights
def _update_parameters(self, derror_dbias, derror_dweights):
self.bias = self.bias - (derror_dbias * self.learning_rate)
self.weights = self.weights - (
derror_dweights * self.learning_rate
)
```

6.2 Source code

```
    import numpy as np
```

```
class NeuralNetwork():
def __init__(self):
# seeding for random number generation
np.random.seed(1)
#converting weights to a 3 by 1 matrix with values from -1 to 1 and mean of 0
self.synaptic_weights = 2 * np.random.random((3, 1)) - 1
def sigmoid(self, x):
#applying the sigmoid function
return 1 / (1 + np.exp(-x))
def sigmoid_derivative(self, x):
#computing derivative to the Sigmoid function
return x * (1 - x)
def train(self, training_inputs, training_outputs, training_iterations):
#training the model to make accurate predictions while adjusting weights continually
for iteration in range(training_iterations):
#siphon the training data via the neuron
output = self.think(training_inputs)
#computing error rate for back-propagation
error = training_outputs - output
#performing weight adjustments
adjustments = np.dot(training_inputs.T, error * self.sigmoid_derivative(output))
self.synaptic_weights += adjustments
def think(self, inputs):
#passing the inputs via the neuron to get output
#converting values to floats
inputs = inputs.astype(float)
output = self.sigmoid(np.dot(inputs, self.synaptic_weights))
return output
if __name__ == "__main__":
#initializing the neuron class
neural_network = NeuralNetwork()
print("Beginning Randomly Generated Weights: ")
print(neural_network.synaptic_weights)
#training data consisting of 4 examples--3 input values and 1 output
training_inputs = np.array([[0,0,1],
[1,1,1],
[1,0,1],
[0,1,1]])
training_outputs = np.array([[0,1,1,0]]).T
#training taking place
neural_network.train(training_inputs, training_outputs, 15000)
print("Ending Weights After Training: ")
print(neural_network.synaptic_weights)
user_input_one = str(input("User Input One: "))
user_input_two = str(input("User Input Two: "))
user_input_three = str(input("User Input Three: "))
print("Considering New Situation: ", user_input_one, user_input_two, user_input_three)
print("New Output data: ")
print(neural_network.think(np.array([user_input_one, user_input_two, user_input_three])))
print("Wow, we did it!")
```

7 Discussion

Structure and behavior - In BRA-driven development, we decided to use HCD as the main source of design information because it expresses the structural aspects of functions. Moreover, the accumulation of findings has progressed because the mesoscopic level structure in the brain, which is the basis of an HCD, embodies invariant information that is not task-dependent. However, not only the structural aspect but also the behavioral aspect is important in software design. Therefore, an important task in the future is to develop behavior-related evaluations that are similar to functional similarity and activity reproducibility used for fidelity evaluation.

Roadmap for reaching AGI - Given the characteristics of this development methodology, the following four milestones need to be achieved on the way in the roadmap leading to the realization of a brain inspired AGI. The first milestone is the construction of a BIF for almost the whole brain. The second milestone is the construction of an HCD that covers almost the whole brain, and the third milestone is the construction of integrated software that covers almost the whole brain. Finally, the fourth milestone is to achieve a situation where AGI solutions can be explored in a design space where constraints continue to be updated as neuroscience progresses, primarily by running and testing integrated software in a virtual environment. However, something may still be lacking even when all the computational mechanisms that make up the brain seem to work together. Applications of AI systems based on BRA AI systems developed - based on BRA can be expected to almost exactly replicate human cognitive and behavioral capabilities. Therefore, it has several practical applications, including the following: Its greatest use is that it allows us to construct an AI that has familiarity with humans when communicating with them. Conversely, findings regarding human cognitive impairment may be used for the problematic behavior seen in brain-inspired AI. Moreover, we believe this can also be used as a computational model that will serve as a device for mind uploading.

8 Result and Conclusion

The main contribution of this research is the establishment of a methodology for accumulating data on brain constraints in a form that can be used for software development.

- 1. Separation of design information: The data can be used in a variety of development projects because they are described in a standard format for software development that does not depend on any particular development environment.
- 2. Standardization of description granularity: As a rule, describing BRA data at a coarser granularity than the mesoscopic level reduces the possibility that development will get caught up in details that are unnecessary for the realization of the target cognitive behavioral level.
- 3. BRA design: The method of designing computational functions according to anatomy allows BRAs to be created while compensating for the lack of neuroscientific knowledge in a wide range of brain areas.
- 4. Tolerance of diversity: Even BRAs that contain mutually contradictory HCDs can be registered if they have a certain level of validity, thereby reducing the risk of overly narrowing the design space to be considered. The above features of the BRA will provide a scaffold for large-scale whole brain software development as the comprehensiveness of its data increases.

This will allow the brain architecture to exert a centripetal force as an anchor that can efficiently bring about the convergence and eventual completion of the development of human-like AGI, when currently the development results in this field have a tendency to diverge.

9 References

[1] https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
[2] https://wiki.pathmind.com/neural-network
[3] https://becominghuman.ai/mathematics-of-neural-network-13d204ebfe1
[4] https://becominghuman.ai/understanding-neural-networks-2-the-math-of-neural-networks-in-3-equations-6085fd3f09df
[5] "The Prospects of Whole Brain Emulation within the next Half- Century", by Daniel Eth, Juan-Carlos Foust, Brandon Whale
[6] https://github.com/neurolib-dev/neurolib
[7] https://realpython.com/python-ai-neural-network/
[8] https://towardsdatascience.com/deep-learning-with-python-neural-networks-complete-tutorial-6b53c0b06af0
[9] https://www.activestate.com/resources/quick-reads/how-to-create-a-neural-network-in-python-with-and-without-keras/
[10] https://link.springer.com/chapter/10.1007/978-3-642-31674-$6_1$9

[11] https://sciendo.com/pdf/10.2478/jagi-2013-0012
[12] https://sites.google.com/site/carboncopiesproject/the-history-of-sim-whole-brain-em
[13] https://www.degruyter.com/document/doi/10.1515/9781400851935-015/pdf
[14] https://medium.com/@camirosso/the-beauty-of-the-bayes-theorem-the-trending-of-bayesian-analysis-9f0d1af55f62
[15] "A COGNITIVE ARCHITECTURE FOR KNOWLEDGE EXPLOITATION" by GEE WAH NG, YUAN SIN TAN, LOO NIN TEOW, KHIN HUA NG, KHENG HWEE TAN, and RUI ZHONG CHAN.
[16] "The whole brain architecture approach: Accelerating the development of artificial general intelligence by referring to the brain", by Hiroshi Yamakawa.