Go Wide, Get Deep: The Neocortex Is A Directed Graph Of Autoencoders

Version 1.0

Rafi Qumsieh
rafiqumsieh@gmail.com

July 28, 2021

Abstract

The main conjecture that is discussed in this paper is that the Neocortex is a directed graph of autoencoders. Each cortical column acts as an autoencoder whose purpose is to compress, encode, and later retrieve the information that passes through it. By having such a structure, the Neocortex can compute algorithms that have high levels of complexity. Its modular structure, its topology, and its plasticity allow it to form paths which correspond to composing functions together, which allows for computing complex algorithms.

1 INTRODUCTION

Humans use prior knowledge to effectively and robustly learn new tasks. State-of-the-art Machine Learning algorithms are effective in learning complex maps given enough data. These algorithms, however, need to be trained from scratch on new tasks. Many methods try to reduce the need for retraining like transfer learning and other few-shot learning techniques, but these systems are not equipped for online learning by design. In this paper, we will attempt to build a system that can combine prior knowledge with new data to solve the problems it is presented with.

The Universal Approximator Theorem shows that neural networks can approximate continuous real-valued functions when certain conditions are met. This implies that we can create a neural network to approximate any real-valued function to the degree that we wish by adjusting the network's size and parameters [1]. It was proved that Lambda Calculus is Turing-Complete [2], which means that it can perform any computation given enough resources. Functional programming is a process that uses the principles of Lambda Calculus to perform computations by using pure functions. Therefore, one can assume that, since neural networks are function approximators, a collection of neural networks can be Turing-Complete. The question becomes: What kind of architectures can be used to compose that collection of neural networks in a way that allows for any kind of computation, while maintaining the ability to learn?

One way to gain insights on this question is by looking at how nature implements computations. The Neocortex is the outermost part of the brain. It is a thin sheet of tissue that has roughly the size of a table cloth. It is hypothesized that the Neocortex, which is referred to as the seat of human intelligence, consists of modular repeating structures. These structures are referred to as the cortical columns [3].

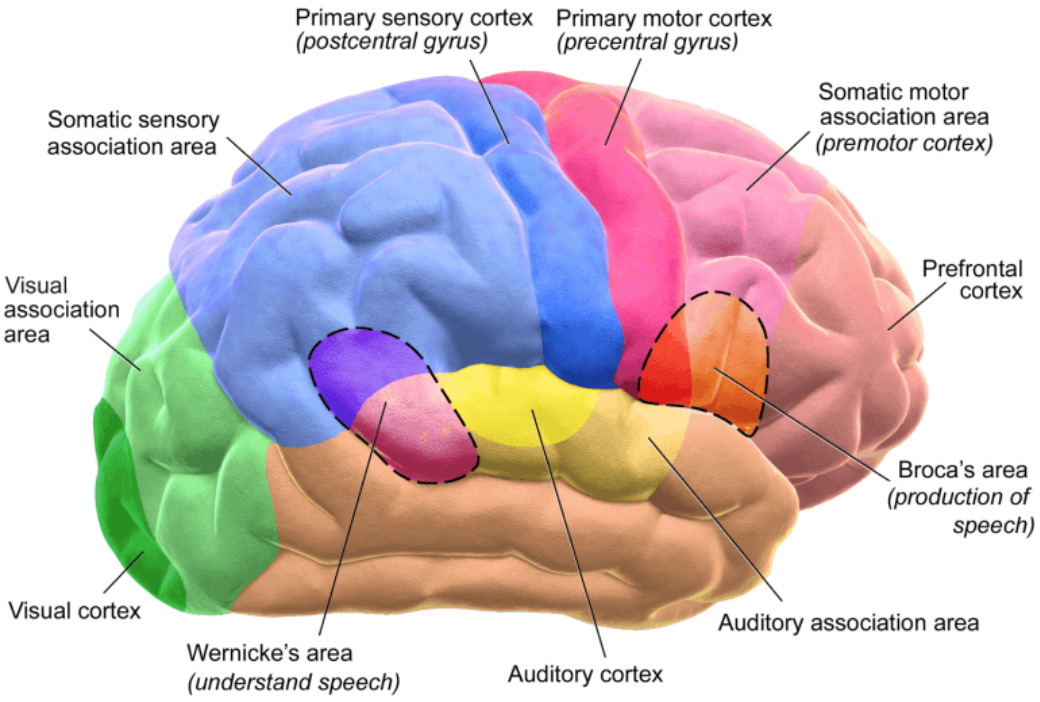

Figure 1: The neocortex (which means "bark" in Latin) is the wrinkly outermost part of the brain. It is a relatively uniform and modular structure. Each module is called a cortical column.

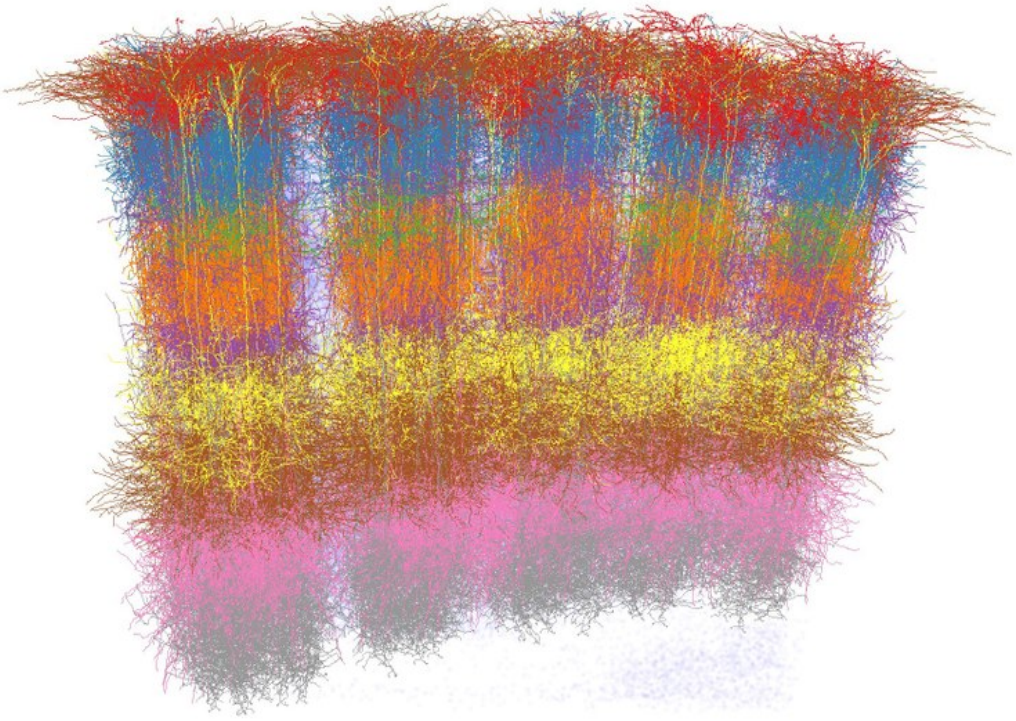

Figure 2: A depiction of a group of cortical columns, which make up the Neocortex. There are variations among these columns depending on the region of the Neocortex.

Modularity is a very powerful architectural design for several reasons:

1. It is efficient to store the information about a modular structure genetically, as you only need to store the information about one part along with some other meta data.

2. If each module contains information, then that information can be reused if the module is accessible by other parts and modules.

3. Modules can be composed to achieve more complex functionalities.

4. If each module can only learn a simple function, then the amount of data needed to train each module should be relatively small.

The conjecture is that each cortical column is an autoencoder. Autoencoders offer the advantage of using the input data both as input and and as target output. This means that a column encodes information and learns in a self-supervised manner. The hidden layers contain a compressed, latent representation of their input. The hidden layers act as content-addressable memory which can be approximated by some form of a Hopfield network. This can be useful since the Neocortex can use these representations to communicate information across its modules (columns) in an efficient way.

If we assume that the Neocortex is a modular structure, and we assume that each cortical column is an autoencoder neural network that can learn a simple function. Then, we can conclude that the Neocortex is a collection of function approximators. The columns in the cortex are connected to each other via lateral connections in Layers 2, 3 and 5. They are also connected to each other via sub-cortical regions like the Thalamus and the Basal Ganglia. The connections are altered via plasticity rules. So, if a certain module fires, it can cause another module to fire. This causal relationship between modules induces a direction in the connections. So, it is more appropriate to describe the Neocortex as a directed graph of modules rather than a collection of modules. These directed graphs can encode information in the different sequences and paths of activations.

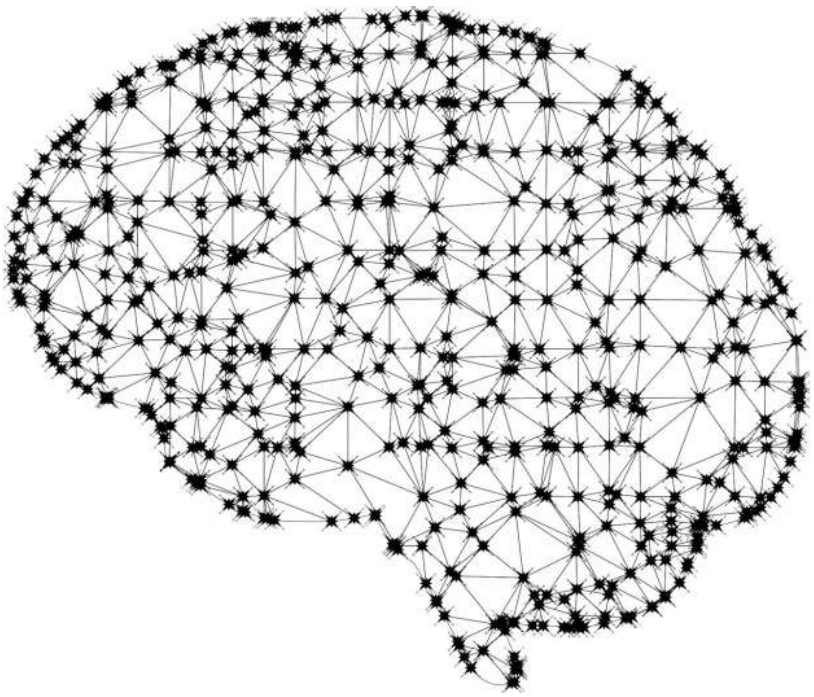

Figure 3: A depiction of the cortical columns of the Neocortex forming a graph. Different paths of activations can lead to different computations.

To complete the picture, let us briefly discuss algorithms because humans can not only learn simple functions, but they can also learn complex algorithms. Algorithms have several definitions, but one of them is that an algorithm is a function that can be broken down into sub-functions, which can also be algorithms. Algorithms can be written down as compositions of functions. Therefore, an algorithm can be represented by a directed graph of neural networks. So putting everything together, the Neocortex is a directed graph of modules, where each module is a shallow autoencoder neural network that can encode some information. Information learned in each module can be re-used. The modules are connected to each other via Hebbian learning to allow for sequential and causal relationships among the modules. This allows the Neocortex to approximate different algorithms using different activation paths on the graph. These paths can be deep or short, and are similar to what state-of-the-art deep neural networks represent. The only difference here is that each layer is an actual network instead of a single layer of independent neurons.

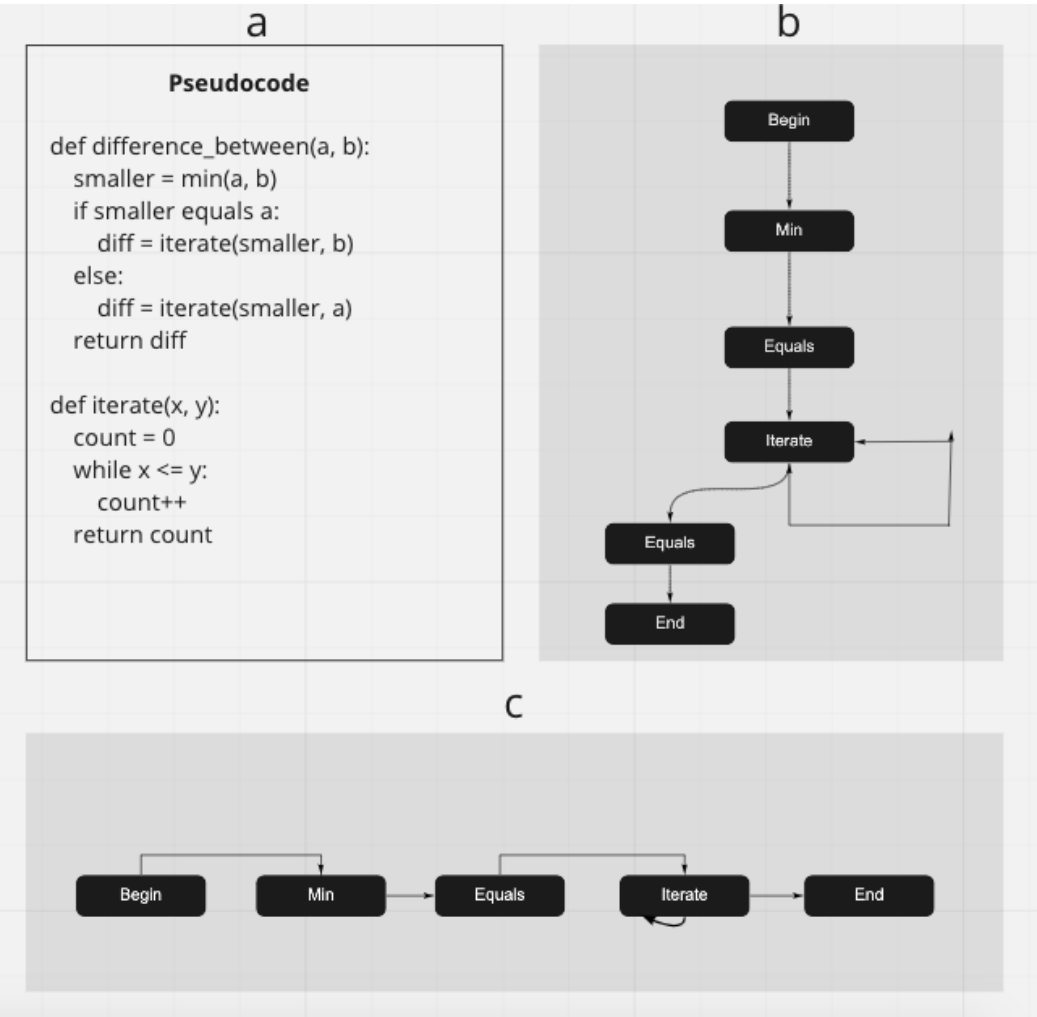

Figure 4: Part (a) describes a pseudocode of a program to calculate the difference between two integers. Part (b) shows a flow of that same program as a composition of predefined modules. Part (c) shows the modules in a flat hierarchy where each module is free to connect to any other module.

How can such a structure learn temporal information and patterns? We will assume that patterns that occur simultaneously or patterns that occur causally will be associated together. Formally, let us start with temporal patterns that are received from multiple sensory inputs P. Let us define P

as a relation in the cross product between the sets of time T and state vectors S. That is:

$$S : \{s \in R^n \mid n \in 1, 2, 3, ...\}$$

and:

$$P : T \times S$$

Where each element in the relation is a tuple of a timestamp and an associated state vector. The reason the above is defined as a relation instead of a function is that because at a certain time, many inputs can be simultaneously present. Now consider the two patterns $p_1 = (t_1, s_1)$, $p_2 = (t_2, s_2) \in P$, where $t_1 \leq t_2$. We will consider the two states s_1 and s_2 to be causally related if the inputs occur close to each other temporally. That is, the corresponding times of the inputs t_1 and t_2 are close to each other up to a real-valued threshold δ, so they must satisfy this condition:

$$|t_2 - t_1| \leq \delta$$

If this condition is satisfied, we say that state s_1 caused state s_2. We can associate a real-valued score for each pair states to indicate the strength of causality between the states. This pair-wise relation between the states induces a causality matrix CM between the states, where each entry is the causality score between the states. This matrix can also be thought of as a form of an incidence matrix between the states. The introduction of this matrix allows us to think of the states as nodes in a directed graph.

2 POST SCRIPTUM

This paper is intended to be a motivational paper for researching and developing hypotheses that can be tested based on this conjecture.

3 REFERENCES

1. Cybenko, G. (1989) "Approximation by superpositions of a sigmoidal function", Mathematics of Control, Signals, and Systems, 2(4), 303–314. doi:10.1007/BF02551274

2. Turing, Alan M. (December 1937). "Computability and Lambda-Definability". The Journal of Symbolic Logic. 2 (4): 153–163. doi:10.2307/2268280. JSTOR 2268280.

3. Kurzweil R (2012). How to Create a Mind: The Secret of Human Thought Revealed. New York: Viking Penguin. p. 36. ISBN 978-0670025299.