PROCEEDINGS OF SCIENCE

Quantum Computing Future-Proofing What Lies Beyond Super-Computing

Scott L. Hamilton[1]
Atos
4851 Regent Blvd, Irving, TX, U.S.
E-mail: scott.hamilton@atos.net

The race is on as more computing power is required to solve some of the most complex, data-intensive problems in existence today. How will quantum computing overcome its inherent challenges - cost, footprint, power consumption, temperature requirements, instability - to leapfrog super computers as the platform for the future? See how quantum simulators are helping to overcome those challenges and enabling researchers and end users alike the ability to develop quantum algorithms that will solve problems faster, leading to new pharmaceuticals and medical treatments, improved financial modelling and weather forecasting, cheaper energy production and much more.

Artificial Intelligence for Science, Industry and Society, AISIS2019
October 21-25, 2019
Universidad Nacional Autónoma de México, Mexico City, México

[1]Scott L. Hamiltonach

PoS(AISIS2019)047

Introduction

As a Senior Hardware & HPC Expert at ATOS, I have several years of experience in software and hardware engineering and a desire to share my passion and knowledge of high-performance computing (HPC) with others. As lead for HPC research at the Missouri University of Science and Technology, I supported multidisciplinary research through computational simulation on HPC systems. At Atos, I design custom hardware and software solutions for computational sciences, data sciences, and scientific simulation and modeling. I frequently partner with research scientists in various fields, assisting them with software design for simulations and demonstrations.

As a result of these research-oriented activities I have come in contact with many of the top engineers in the fields of machine learning, artificial intelligence and quantum computing. Out of these relationships came the knowledge of how-to-future-proof our algorithms for the next generation of computation. In this paper I plan to discuss the basics of machine learning and artificial intelligence and show the advancements of hardware and algorithms over the past decade and into the future of Quantum Computing.

Human Learning

Before we can really talk about machine learning, we need to know a little about human learning see Figure 1. How does our brain work? How do we learn if we are safe or in danger? How do we learn what we can and cannot eat? How do we keep the information we learned for the future? The information is stored within chemical and electrical bonds between neurons in the brain. These neurons build pathways which control behavior and reactions. The more times a neuron fires, the stronger the link becomes, eventually making it the default flow of information through the brain.

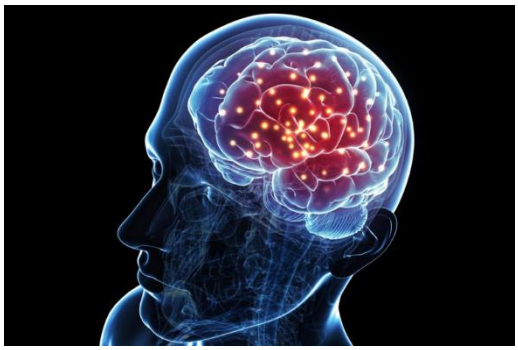

Figure 1: Human Leraning

The human brain consists of 100 billion neurons, 100 trillion synapses, and is massively parallel see Figure 2. The synapses for the connections for information flow to, through and from the brain. In essence, the synapses/neuron interaction is what controls our ability to learn. Modern Machine Learning (ML) Systems emulate learning at the neural level and are the basis for most modern machine learning algorithms.

Let's take a fictitious example from a cave man: learning to avoid the dangerous sabre-toothed tiger. To simplify the explanation, we will assume that the fight or flight response is controlled by a single neuron. This neuron can have several synapses based on sight, sound, smell, touch, or perception. On the first encounter with a sabre-toothed tiger the sight synapse sends a signal to the neuron saying, "That's a sabre-toothed

Figure 2: A nueron and the synapses connecting it to the nervous system.

tiger." The neuron says, "Run!" and remembers to listen more closely to this synapse next time. A few minutes later another synapse says, "Run! It's a fluffy bunny!" The neuron ignores this signal and remembers to listen to this synapse less in the future. The learning process continues as the neuron learns which signals are more or less important in the flight or fight decision.

Machine Learning

Modern machine learning techniques use the above model of the human brain as a starting point for "teaching" a computer to learn, like a human. We start with a neural network consisting of a series of neurons connected by synapses. The machine learning process is accomplished through modifying the weights of the synapses between the neurons. This network is trained by being provided with a data set of known outcomes. These input/output pairs are fed into the network and the weights are adjusted until for every known input state, the network provides the expected output state. Once the network is trained, it can then be utilized to categorize data that it has never seen before, in effect learning the expected output for the unknown input based on the history taken from the training process.

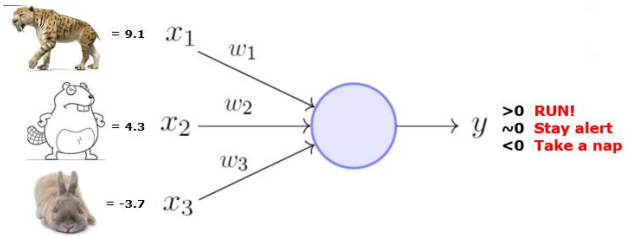

Figure 3: Diagram depicting the fight or flight nueron discussed in Human Learning as a computer model.

To understand a little more about machine learning, we are going to look at Figure 3 and recall the story of the caveman learning when to run or take a nap when faced with a creature. Imagine for a moment that what was learned was not correct. In the case when a caveman sees at the same time a badger, a fluffy bunny, and a sabre-thoothed tiger shown in Figure 3, the neuron with weights of (w1=1, w2=1, w3=5) would receive a 9.1 (it's a sabre-toothed tiger), a 4.3 (it's a badger) and a -3.7 (it's a bunny). This would result in a total of -5.1, so the model would take a nap and be eaten. It becomes obvious that this model is not correct for keeping the model safe. So adjustments need to happen; let's try a different weight (w1=5, w2=1 and w3=-5) and now we get 68.3, which tells the model to run, so this is good. However, the model must be tested with several different input cases before we can confirm correctness.

How the computer does this is to translate the weights into binary (W1 = 5 = 000101, W2 = 1 = 000001, and W3 = -5 = 100101); it then tests the training set with these weights and provides a scoring based on the match percentage. It then modifies a single bit in the weight string and tries the training set again. Each changed bit either improves or hurts the model's performance. This is a very long process; even in our simple example it requires testing the entire training set over 68 billion times to arrive at the optimal weights for training. Complex models contain more than one neuron and require very significant training times, changing single bits and computing a score repeatedly.

Speeding up the Process

Over the many years of machine learning software development, new technologies have come online to improve the training time of the algorithms. Once a model is trained, executing

the model is a simple and low-cost operation when it comes to computing power, but training can take months on complex deep learning models like the one pictured in Figure 4. The model here is one example that can take months of computer cycles to train, and every time new data is introduced to the system, benefits can be gained by retraining the model.

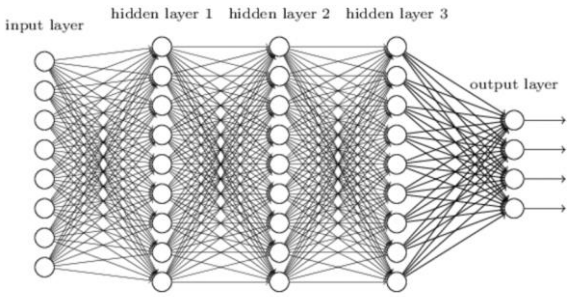

Figure 4: Dipiction of a deep learning network with many neurons and synapses where each can have a differen wieght and all the neuron interact with one another to different degrees.

The model we have described so far is very dependent on the number of simultaneous tests you can perform. Up until around 2001 you were limited to a single test per processor, resulting in needing multiple systems to process even relatively small machine learning tasks. In 2001 multi-core processor technology came in to play in the machine learning community. It resulted in being able to look at multiple weights simultaneously but was only a small step forward. In 2005 timeframe graphic processing units (GPUs) began to be used to train neural networks; this took the number of threads from around 12 to over 3000. This meant that 3000 weight vectors could be tested simultaneously resulting in a much faster training process.

The latest technology to coming to the market is quantum computing technology. It's not quite mainstream yet, but in effect, quantum processing units (QPUs) can process all possible weights simultaneously, at least as long as the number of bits in the weight vector is smaller than the number of qbits in the QPU, brining nearly infinite speedup to the training process on neural networks.

The concepts of modern machine learning techniques were described before the microprocessor-based computer. It was described in 1949 by Donald Hebb and was based on the neural learning mechanism within the human brain. Hebb wrote, "When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell."[2] Translating Hebb's concept to machines, we narrow it down to a weighting mechanism between artificial neurons and their relationship with each other.

In neural networks today we base the learning outcome on a scoring mechanism where the score determines the decision. For example, in a self-driving car, a high score means, "hit the brakes; there is a child in the road," and zero children die, and a low score means, "keep moving; everything is fine", and a child dies. We base this scoring on a series of artificial neurons that add together all the various inputs of the system to arrive at a score.

The neural network is trained by adjusting the weights of the links between the neurons through feeding in a known set of training data and adjusting the weights until the expected score for the training set is output correctly for every item in the training set. The training requires adjusting every neural link many times and processing the entire training set after each change; as a result the training can take weeks. This is where quantum computing comes in to play.

[2] Hebb, D.O. (1949). *The Organization of Behavior*. New York: Wiley & Sons.

Quantum Computing Overview

Quantum computers work off of a quantum mechanical bit called a qubit. What makes a qubit so special is that it can be in two different states simultaneously known as superposition. We can think of this like a coin tossed in the air spinning. While it is in the air it has equal probability of landing on heads or tails (0 or 1) and can be considered as being in superposition. Just like the coin cannot flip through the air continuously, a qubit cannot remain in superposition forever either. This is one of the first issues with producing stable quantum processors, the problem of decoherence, which can be solved to some degree by lowering the temperature of the qubit to near absolute zero, which greatly increases the amount of time it can remain in this superposition state.

The second strange property of a qubit is that it can become entangled with another qubit, meaning that the two entangled qubits will always react in the same way, if you read one qubit the other will instantly become the opposite value, or the same value depending on the initial entangled state, they stay related in the same way. I like to think of entanglement of ballroom dancing partners, one partner will always be facing you and the other will be facing away from you. When entanglement is perfect, they will follow one another exactly with no delay. Just like ballroom dancers cannot stay in perfect sync, neither can quantum systems, which results in a phase shift between the two qubits. With enough of a phase shift, the results of the process are not accurate.

There is no theoretical limitation for perfect entanglement, and eternal superposition, but the environmental noise cannot be suppressed completely resulting in limitations to the observation of perfect entanglement and eternal superposition.

Quantum Machine Learning

Quantum Computing, which is also not so new, was first conceptualized in the early 1980s by Paul Benioff[3][4], and very shortly afterward, Richard Feynman[5] and Yuri Manin[6] suggested that a quantum mechanical computer like Benioff proposed could perform calculations that are out of reach for classical computers. This comes about because of their ability to examine the results of multiple input scenarios simultaneously.

Quantum computers can be used to determine the weighting of the links between the artificial neurons in a fraction of the time because of their ability to test all weights simultaneously, or all input simultaneously, depending on the exact learning method employed. The first method is to use the quantum bits (qubits) to represent the neuron weights, allowing you to test all the input data in sequence and pick the best weights for the training data. The second method is to represent the data with the qubits and test all the possible weights in sequence. The fastest method

[3] "Quantum Mechanical Models of Turing Machines That Dissipate No Energy", Paul Benioff, Physical Review Letters, 48, 1581 (1982).

[4] " Quantum mechanical hamiltonian models of turing machines", Paul Benioff, Journal of Statistical Physics, Vol. 29, 515-546, 1982

[5] Feynman, Richard (1982). "Simulating Physics with Computers". International Journal of Theoretical Physics. 21 (6–7): 467–488. Bibcode:1982IJTP...21..467F. CiteSeerX 10.1.1.45.9310. doi:10.1007/BF02650179.

[6] Manin: Quantum groups and non commutative geometry, Montreal, Centre de Recherches Mathématiques, 1988

depends on the size of the neural network and training data. If your quantum system is large enough, you want to represent the larger of the two using qubits.

To make it a little easier to understand, let us think about a very simple test case with four neurons and a training data set with 32 values. This would require you to check weights of four links, requiring 16 total tests, for each of the 32 input values, for a total of 512 tests using classic machine learning. Because the qubits can set all possible values at once if we represent the four weights with four qubits, we run all weights at once and only perform 32 tests. If we represent the training data with the qubits, we can run all inputs at once and only need to run 16 tests. If we bring these values into current training model sizes, you begin to see the power. Modern quantum computers are limited to 52 qubits which allows for two to the 52^{nd} power weights that can be tested simultaneously, which will reduce the total number of steps for training a 52 neuron network by roughly the number of atoms in the universe if we assume 1-qubit weights. Actual machine learning algorithms would likely use 8-qubit weights reducing the size of the neural network to seven neurons, but the overall speed of computation would still be multiple orders of magnitude.