Research Proposal - A Question for the Machine

Intro

We are in the midst of a hardware renaissance and there are more chip startups than ever, while at the same time machine learning research hasn't progressed beyond backpropagation. Thousands of papers a year, each following the same research path of trying out new tricks with existing deep learning libraries. Experts understand that learning using the backpropagation algorithm has limitations - nothing in the framework can account for how to do long term credit assignment or continuous learning like the brain does. And it is easy to construct examples where gradient based optimization makes little sense. There should be something beyond it. So far, deep learning has swept away the objections of its detractors using practical success, but its successes as they are have come from using bigger, better and faster computers, rather than revolutionary algorithmic advancements. Attempts are being made, but nobody has discovered a principle to serve as the basis for engineering the next generation of learning systems.

I myself have tried to move beyond backpropagation and failed. It is strange that nobody else has managed it either, but at some point one has to accept that the problem is intrinsically difficult to reason about for humans. Moreover, not being able to do it is humiliating. Intelligence is something we are always surrounded by and see examples of all the time, and yet we cannot grasp it.

Maybe it would be best to just ask the machines how to do it? With the right setup and sufficiently powerful hardware, the hardware itself should be able to tell us how to best make use of it.

Goal

- Implement a game environment on an AI chip
- Implement a genetic programming system on an AI chip.
- Use it to infer a learning algorithm through game play.

The hope is that the learning algorithm would be a novel discovery we could study and think about, and mine for insights into the true principles of learning. The way ML research is currently done is inefficient. It relies very heavily on the ingenuity and imagination of the researcher itself who has to take great care in testing out a single one of his ideas at a time. It would be better to automate that in imitation of the ultimate scientist that is nature.

Genetic programming and evolutionary algorithms aren't a new thing by now and haven't produced particularly notable results. But then again, even in the early 20th century, there were Turing machines and lambda calculus, so programming existed - you could do it on a piece of paper. But it was not until decades later than the dawn of computers came and programming became a profession. I believe that if I had an oracle and asked it for an optimal algorithm for some game given realistic hardware constraints, the results would be quite enlightening. It certainly would not be backprop based.

About AI Chips

The way to think about them is that they are parallel computing devices, with many cores which communicate using message passing and only have access to their own local memory. They are not like GPUs which have a different parallelism model, but more like a cluster of instances on the cloud. Each instance has its own resources and cannot read the memory of another directly. Instead it has to communicate on the network. This kind of computing model will arise in AI chips and pave the way to the next generation of computing devices. Whereas CPUs' frequency scaling and now core count is stuck, these kinds of devices are the true spiritual successors to the CPUs of old, carrying on the will of Moore's Law.

Motivation For Using Them

Turing completeness being what it is, you could do what I propose on the cloud even without the use of exotic hardware, but it would be prohibitively expensive. Instead having the cloud right in your desktop rig is what would allow the costs of pursuing this path of research viable. I am being metaphoric. While in the future these chips will be available for purchase even to regular individuals for their PCs, the most likely avenue of research is to rent them over cloud due to the need to scale.

Why not GPUs?

Due to their computing model, GPUs aren't good when the task requires its individual cores to communicate. It would be difficult to implement an interpreter and an AST instancer and mutator on it. Furthermore, it would be difficult to implement a game directly on it for the same reason.

Then what about GPU/CPUs?

This is the dominant paradigm when doing RL research, but has significant efficiency issues. The communication overhead between the devices is harmful to performance. Even worse, unlike the brain, GPUs are poor at processing singular inputs. In supervised learning where you have a static dataset, this is less of an issue, but piping a simulator to a learning system hosted on the GPU is painful.

With the right hardware, all those problems will go away.

Given the benefits they could bring to their users, the essence of my proposal is that serious ML researchers should keep the hardware coming down the pipeline in mind, and even start preparing for them. It is an adventure as a new domain gets discovered. New hardware will bring with it new research and programming opportunities.

The Sketch of the System

In backpropagation, you could consider the individual parameters to be competing against each other. But it is not like the layers themselves have the freedom to change the way they learn as the training progresses. Instead of considering a system made out of layers like in deep learning, why not have each of the AI chip cores have its own individual programming that evolves according to the outcomes of gameplay? The static part of each core would be the genetic programming system evolving the AST defining its behavior, and an interpreter executing such an AST. Each core would have the ability of suppressing or accepting the messaging of others, in essence defining its own interaction protocol with the others. The hope is that through training, the individual units would converge to proper and efficient learning algorithms and also that efficient inter-core communication patterns would arise naturally through the training process.

I haven't done a full literature search on whether this has been done. Evolving learning algorithms has certainly been done. For example: AutoML-Zero: Evolving Machine Learning Algorithms From Scratch. Like in that paper, the research I've seen fundamentally tries to evolve a specific learning algorithm that can then be run on a single GPU and on a static dataset. Backprop seems to be good on static datasets and supervised learning tasks, so the resulting algorithms are just hacky variations on it rather than revolutionary gusts of insight the researcher would hope.

It could be done on the cloud, but you'd probably want 10k independent instances to even begin such research. Very few could afford to rent such computational power. But AI chips with 10k cores will not be rare. Doing this research proposal on current CPU/GPU systems would be quite difficult and it would not be natural by any means. You'd have to contort the system to make the fit. With AI chips, the research proposal pretty much falls out from the architecture of them.

Expectations

The proposal would work given sufficient computing power. Maybe a single chip of 10k cores would not be enough to make a good game playing agent or evolve particularly interesting algorithms, but with enough of it the proposal would certainly succeed. Right now, finding the next generations of learning algorithms is a big deal, but in the future the hardware will be powerful enough that anybody would be able to derive them from scratch. Intelligence of a system is so vital to its survival, so there is something about the structure of the universe itself that supports its evolution and development. Developing learning algorithms might be hard to do wrong, but might be easy to do right even though it seems simply hard all around to us now.

The big question is just how much computational power would be required to evolve revolutionary learning algorithms using a genetic programming system. Right we could spend 10k dollars on researching this, maybe if it fails go to 100k. But after that do we go to 1m, 10m or 100m just throwing money into a furnace?

I suggest that this proposal simply be kept in mind and repeated every few years on a small scale. Maybe today the project could fail, but in a few years the hardware will be better and could be tried more cheaply. Maybe in the future there will be material breakthroughs in memristors and we will have access to devices with terabytes of non-volatile memory on a single chip. A 10k of computation in 2032 might go for 1m today which would make success a lot more likely. Of course, by that point, maybe some genius will reason out the way learning works without relying on tools to do it, or neuroscientists will reveal the secrets of the brain. I'd prefer the insights came as soon possible regardless of the path taken to get to the source, but I'd prefer to be proactive. There is no need to trust or rely on the ML community at large to deliver insights and algorithms given its track record. There is no need to look for wisdom in the landfill of papers it produces annually.

What advantages I have in pursuing this line of research

This project would be difficult to an average ML researcher due to inadequate tooling. They might be versed in Python, how will they deal with programming a GP system on a novel computing device with sparse libraries and access to a low level programming language such as C? To me such a situation would be easy. I have my own programming language Spiral. Two weeks ago I did a ref counted C backend in anticipation of pursuing this line of research on my own. It will serve as a prototype that I can easily adapt to any kind of device that I want to program. It would take a few days to create a backend, but after that I would be able to program them in a high level functional programming language.

I've never made a genetic programming system, but I have significant PL experience and creating an interpreter is a cakewalk for me. Once I get access to a device suitable for implementing it, I'll have to go over the literature and study the papers from the field, but I am well versed in implementing ML algorithms by now. In the first place, the reason Spiral was created is specifically to implement ML libraries on esoteric devices. The motivation for creating is was the difficulty of creating a GPU-based library deep learning library in F#. I wanted a language that is high level and comfortable to program in, while at the same time being highly efficient for the sake of GPU kernel compilation as well as capable of easy interop between separate language backends. I succeeded in that goal, and today have something suitable to use on AI chips.

I simply just need access to such chips and the funds to support such research.

Current day (7/29/2022) suitable chip targets

Last year I did some research on companies in the AI chip field. There are dozens of startups, furthermore many bigs companies are making their own accelerator, but only 3 companies made an impression on me. Many startups focus on just inference for example and are insufficiently programmable to support a wide range of functionality, and/or their hardware is photonic or quantum and still in the labs so it is not something I'd be able to get my hands on any time soon. After going through the list, I pared it down to 3: TensTorrent, Groq and Graphcore.

Out of the 3, Tenstorrent has the strongest vision, and based on hearsay the best software stack.

Not all is rosy for them in their competition with Nvidia.

On the benchmarks they lead, their advantage is disappointing, and for the rest of the matches they don't even bother showing up. The 3 companies are being bodied by Nvidia's software muscle. But this does not matter to me as I am not interested in training large convolutional or transformer models in supervised learning regimes. Instead I want to investigate how suitable those chips are making simulators boosted by ML. There might be significant business opportunities in this, and personally I find this line work more interesting that the way ML is currently being done.

Personal anecdote

Last year I tried applying to Intel. The job was for a PL related position. The division at Intel was a company recently acquired and its business was making routers. The problem they have is that the router has limited memory, in fact it only had registers which needed to be completely pre--allocated at compile time. There is no leeway to make use of other kinds of memory if the registers were insufficient to hold the data. In other words, if the compiler could not assign enough registers at compile time, the compilation would fail even if otherwise the program was valid. Register assignment requires solving a graph coloring problem which in NP Hard. I found that interesting because I had taken a Discrete Optimization course back in 2015 and knew good ways of doing it. This story ended with the recruiter telling me that:

- They don't want a language like Spiral at Intel.
- Haven't exhausted the possibilities for deterministic compilation and don't want to consider randomized algorithms.
- Aren't interested in compilation on non-x64 chips.

I didn't get the job, but that episode made me aware that simulators boosted by NNs could have a stronger impact than is expected. Later, I studied 3d, and saw research on raytracing boosted by NNs. On the same channel there are examples of physics simulations being accelerated by NNs.

The point is, there are places where DL could be useful that you wouldn't think of. It is not supervised learning, but it is not reinforcement learning either. It is something in between. It is an opportunity that CPU + GPU combos are poorly suited to handle, but AI chips will find natural fit. This is even without significant algorithmic advances.