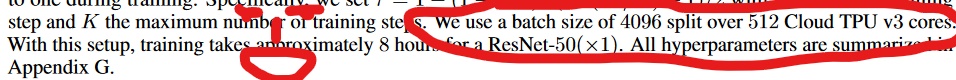
I wrote this as an explanation of how AI specific hardware (in fact also how any hardware) works in combination with the Software and how it might effect AI startups in short term and long term.

By AI specific hardware, I mean new chips being developed just for running AI algorithms in general. However, in this discussion I will speak about hardware being developed to run Deep Learning algorithms in particular. The reason is : A. Deep Learning algorithms require way more compute than any other AI algorithms and thus AI specific hardware is more desired for them. We already need special Hardware like GPUs to run them for all practical purposes. B. Other Machine Learning algorithms (and even normal data analytics) have started migrating to GPU only recently and not too much work seems to have been done to make specialized hardware for them.

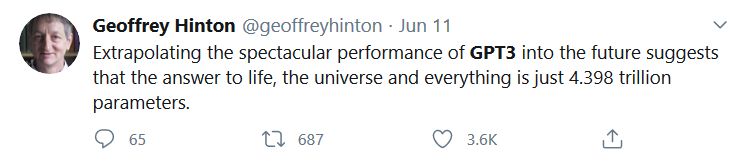
When I removed details specific to my startup (I am one of co-founders of [ParallelDots](https://www.paralleldots.com/)) from this answer, I found this had become a long essay! Before you read, please understand that this is an opinion/prediction document. I am not an expert at hardware and electronics by any means and there may be mistakes. I am just trying to make sense out of a related but unknown field to my field of work (My work is mostly in the field of Machine Learning and Software).

Introduction

One subtrend in the recent AI interest post 2012 is that of AI specific hardware. We all know that it takes lot of (costly) hardware to train Deep Neural Network. Every Nvidia-GPU worth training AI on is a costly machine and the recent mega models trained on large repositories data are trained on an insane amount of compute. An excerpt from the latest BYOL self supervised learning paper by DeepMind:

 Some reaction on size of the latest OpenAI GPT3 model:





So, I hope everyone agrees that research on Deep Learning specific hardware is not a bad investment. It can make deploying AI cheaper, reduce CO2 emissions and many do many other good things. More efficient Hardware will give better results. In fact, even the post AlexNet Deep Learning revolution happened because GPUs were made available. Convnets/LSTMs have been around since late 1980s- early 1990s, nothing substancial came out before the hardware was available.

Let’s talk now about Computer hardware and AI specific hardware in particular:

About Different Hardware Platforms

Microprocessors/chips in our computers

To understand these, we will have to understand how does a program written by a person work ? A programming language compiler converts a program written by a human into a machine readable format. What is a machine readable format ? While most people understand that this is zeros and ones (which is true), in reality, the zeros and ones are actually a low level program that can run on hardware directly. Basically low level instructions are fabricated into the hardware directly. These low level instructions are for example, mathematical operations like Sum, Product, And, Or, Not etc. (General term for this set of logic fabricated into hardware is called instruction set). Now when we are building hardware for a general laptop say, we need to keep the instruction set “Turing Complete”. Turing Complete is a set of instructions in which any program can be represented. This enables the laptop’s chips to be generic and run any possible logic. However, a chip having a generic instruction set is not the most efficient chip for any possible problem. The generic instruction set is build in a way it is good enough for most problems, but specialized hardware is often added for tasks which require more computation. For example, Real Number operations (1.0+1.0) are relatively slow on a generic CPU as compared to Whole Number operations (1 + 1) as they require more computation. A hardware which just runs Floating Point computations fast is present in most Computers by default to optimize (due to the prevalence of floating point ops). In fact, VLSI research has progressed enough to make sure the specialized that floating point hardware is small enough to fit on CPU itself.

GPUs:

GP-GPUs ( General Purpose – Graphic Processing Units), commonly known as GPUs are specialized hardware to run a set of mathematical operations very efficiently. When the feature rich GPUs came out in late-2000s, they were like a super computer could be installed right on your laptop. The compute GPUs provide actually enabled two big trends : Blockchain and AI. Most new Supercomputers we have now days are clusters of GPU. This is in stark contrast of just one decade back, when they were all special purpose hardware. GPUs are still quite general purpose, in the sense they cannot run all programs in parallel, but can run many mathematical operations in parallel. Due to this, they are costly. A simple trend can be understood that most bitcoin mining (which was one of the biggest usecases of GPUs earlier), has migrated to other specialized hardware as new hashes are rarer to find and incrementally more compute is needed as the currency is used. The special purpose hardware has less generic but more optimized functionality for bitcoin mining bringing the compute cost down.

In AI, however, the compute cost is not climbing as much, so well GPUs still continue to be used. Also much more compute is needed at the time of training than deployment. However, there is almost always a debate about deployment as there is significant server costs to be incurred when algorithms are run at scale. While for some applications, the cost might be justified, it might be not for others. So, what do we do to get better bang-for-buck ? Same, less generic and more optimized hardware.

AI specific hardware:

There are two directions of work for AI specific chips :

1. **The bitcoin way :**

That is make more specific hardware than only target the GPU instruction set components aiding Neural Network training (GPUs have a broader instruction set that runs Graphics operations along with other compute). This is how most Bitcoin specific hardware was developed and this has been quite successful there(Technique is called ASICS – Application Specific ICs). In AI technology too, many people have tried this model successfully.

Google for example is already offering its specialized AI hardware on cloud : <https://en.wikipedia.org/wiki/Tensor_processing_unit> . We also have attempts by Alibaba (on its AWS competitor) : <https://techxplore.com/news/2019-09-alibaba-crowns-cloud-powerful-ai.html> and Intel : <https://techxplore.com/news/2018-11-usb-neural-debut-event.html> . These chips are very efficient for Vector Operations, Matrix Multiplications and graph structured data handling, two most common compute routines needed for Neural Network. AliBaba, for example claims its chip is 15 times more powerful than Nvidia’s T4 deployment GPUs and way more powerful than P4 when used for inference of product images. While the Alibaba device is offered on its cloud, just like Google’s device, the Intel device is a <$100 USB device to deploy AI algorithms on laptops or Raspberry Pies.





Both Google and Nvidia have also launched consumer AI deployment hardware like the Intel chip in $100 price range. <https://spectrum.ieee.org/geek-life/hands-on/the-coral-dev-board-takes-googles-ai-to-the-edge> and <https://spectrum.ieee.org/geek-life/hands-on/quickly-embed-ai-into-your-projects-with-nvidias-jetson-nano> respectively.

More research is being done to try and incorporate Spiking Neural Network operations on ASICS rather than normal Neural Network Operations. Spiking Neural Networks emulate the brain more closely than the Artificial Neural Networks we use more commonly. <https://techxplore.com/news/2018-12-hardware-software-co-design-approach-neural-networks.html>



There are also efforts to make extremely low power requirement hardware for small or miniature robots to perform Reinforcement learning on edge and learn online as they work in field. One method to do so is to merge traditional Digital and Analog devices for making ASICS. <https://techxplore.com/news/2019-03-ultra-low-power-chips-small-robots.html>



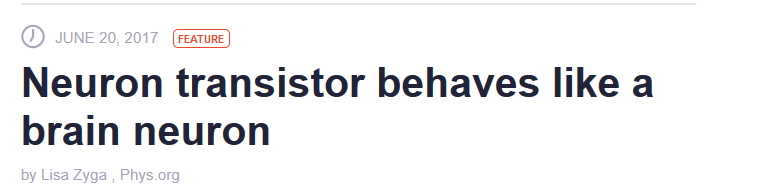
* + 1. 2. **The Brain Inspired Way :**

This is a different approach to build AI specific hardware. Also this is more “cool” in the way that it wants to change the hardware implementation quite fundamentally. While Matrix Multiplications and Vector Operations are layer by layer way to abstract a Neural Network operation, One can abstract it out in another way, at neuron level. Infact the Matrix Multiplication and non-linearity combination we use for every layer of Neural Network can be thought of as less efficient way of implementing artificial neurons in one layer of neural network. If we can represent the artificial neuron on the hardware, we don’t need to even do these costly matrix multiplications (and so hardware costs should go down even more). Essentially the instruction set for running neural networks will now change from Matrix multiplications to neurons. The method is more interesting also because one is building hardware that actually tries to emulate brain Neurons (We do it in software for a very long time, it will now be available in hardware directly). The long term aim here is that we should be able to pack much much more optimized compute for AI algorithms on one chip than what we have now, increasing the training and inference accuracy manifold.

**So how do these hardware designs work you ask ? Let’s do a mini survey :**

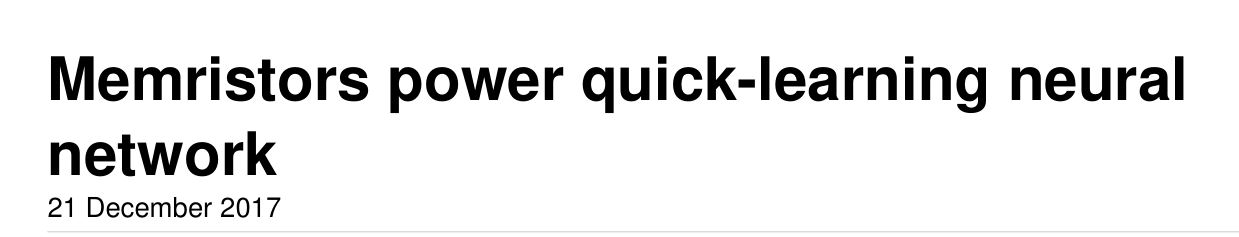
One thing that you should probably think about is that these approaches are of two types, A. having programmable weights and dumb thresholding, like what our software based Neural Networks have and B. Working using smart learnable thresholding (synaptic).

Simplest way you can think of the change happening, is at the Transistor layer, where instead of traditional transistor, we use a Neuron Transistor. That is, it takes input charges and can do what a basic artificial neuron does with its input, weighted sum followed by thresholding (thresholding determines whether a Neuron fires or stays inactive for an input). <https://phys.org/news/2017-06-neuron-transistor-brain.html> developed a Neuron Transistor in form of a Molybdenum Disulphide flake that works like an artificial neuron.

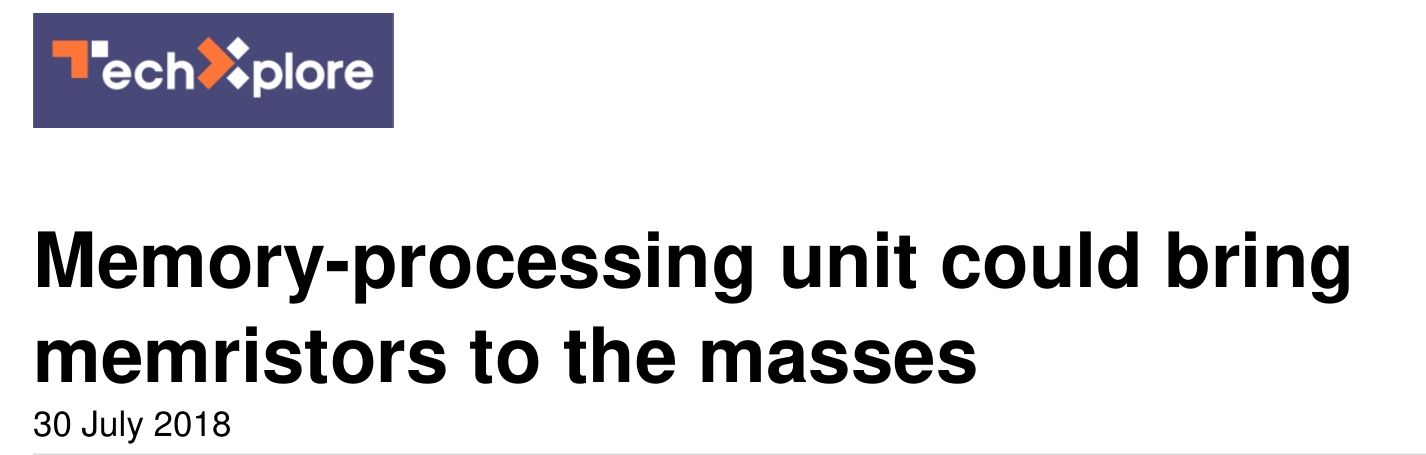
This however models one Neuron, we have to also build a chip that can model Dense connection between Neurons so that multiple layers of a Deep Artificial Neural Network can be emulated on the chip. In human brain, the process of the dense connections is done by Synapses. These synapses threshold the signal and fire (ie pass information) for a strong signal and stay put for a weak one. <https://phys.org/news/2018-01-artificial-synapse-brain-on-a-chip-hardware.html> have designed this signal carrier layer (or artificial synapses) for use on a chip using a 2D Silion-Germanium material. They have even trained an MNIST classifier using the new synpases hardware.



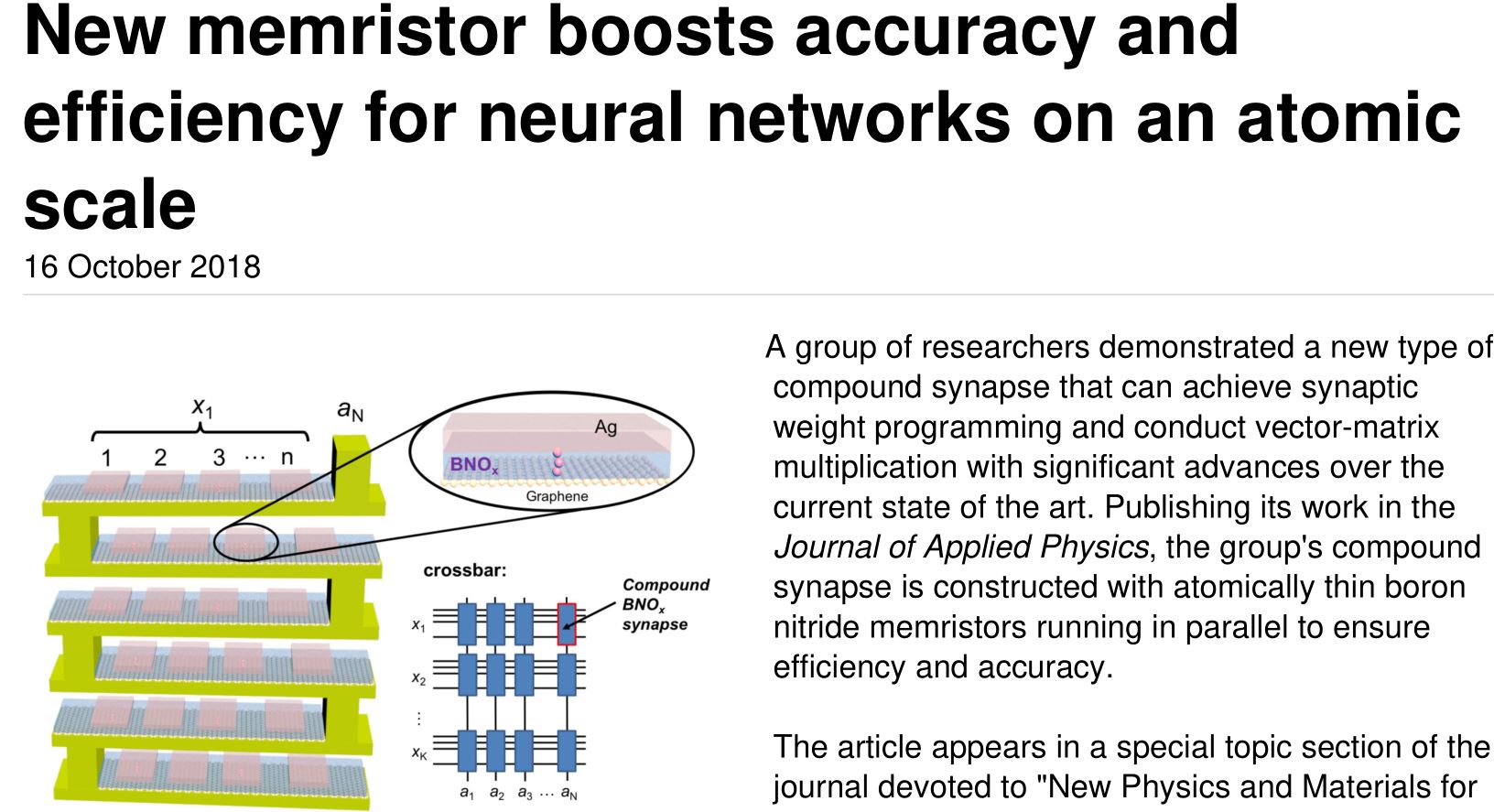
Another direction of research for brain inspired computing is the one using Memristors. Memristors are electronic components which can both perform computations and store data. The data is stored by programming the resistance of the memristor. If you have seen functioning of any Neural Network, you will be able to relate memristors with Neurons. Neurons both store data (that is weights) and perform computations (forward propagation/ backward propagation) on inputs and weights both. There have always been attempts to learn Neural Networks through memristor elements, as the programmable resistance component help memristors save data as a continuum (like Neural Network weights), rather than binary states like 0/1. In speech recognition/generation, Neural Networks running on memristor can run realtime with respect to Human Speech. <https://phys.org/news/2017-12-memristors-power-quick-learning-neural-network.html>

However, the problem with memristors is that the resistance value stored on them is not precise enough (like the float point values F16/32/64 we train/use our Neural Networks at). To counter this, one approach is basically discretize the values of these memory resistances (thus arriving at multiple states like 0,1,2....) and then use this to store floating points precisely. Afterall, as we saw earlier, even CPUs store all floating points as binary values. This helps us have precise floating points with some tradeoffs. The method is called Memory Processing Units.

Link: <https://techxplore.com/news/2018-07-memory-processing-memristors-masses.html>

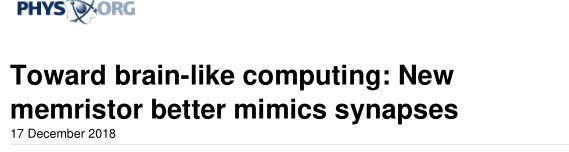
Just like in software, it seems in hardware too, larger the number of memristor devices one can support, the better is the accuracy. However, too many Memristors would also mean too much need of energy. To solve this, researchers have built atomic level memristors which are energy effcient. Silver and Boron-Nitride topped on Graphene have been used to build such microscopic memristors that can work in parallel. These memristors help model the weighted averaging property of Neurons.

Link :: <https://phys.org/news/2018-10-memristor-boosts-accuracy-efficiency-neural.html>



Another way to make molecular level memristors is to use Molybdenum Disulphide (our friend whom we have already seen previously) with Lithium. These memristors can be a way to model thresholding in Artificial Neurons. Molybdenum Disulphide behaves like a semi-conductor generally, allowing less current flow, but in presence of Lithium ions, it changes its structure starting to behave like a conductor. Thus, the amount of Lithium varying can build the thresholding of firing of Neurons, doing what synpases do in real brains. Electric field can control flow of Lithium ions, helping us program the system.

Link: <https://phys.org/news/2018-12-brain-like-memristor-mimics-synapses.html>



Another cool work is doing Associative Learning on a memristor based hardware. Associative Learning for Software based Machine Learning practitioners is what Reinforcement Learning is. While most Memristor based technologies need input-output together to learn, in real world, action and reward are often not happening at the same time. A time-delayed input based Memristor Synapse was invented for this purpose. Link :: <https://techxplore.com/news/2019-12-memristor-based-neural-network-notion-associative.html>

 <https://techxplore.com/news/2019-07-programmable-memristor-aims-ai-cloud.html> is a programmable memristor circuit which can be used in real world for two-layered Neural Networks. It can still work on toy problems only, but is generic to be trained across whatever domain you want. Big step indeed.



More recent innovations include training 3D memristors for the purpose. These enable even more densely connected (deeper due to more possible paths ?) memristor Neural Networks to be run. 3D electronic devices are hard to make from what I can gather and this seems like another big achievement. The unique “local connections” based topology these inventors used not only allows them to build a 3D memristor circuit but also train Convnets, in contrast to old memristor technologies which can only use FC layers. <https://techxplore.com/news/2020-05-d-memristor-based-circuit-brain-inspired.html>



In a similar attempt to put many-many memristors together to make the devices capable of inference on real world problems, researches solved the problem of diluted voltage for ion movement in case of many memristors on a chip. They did this by allowing the Silver + node with Copper and using a Silicon -ve node for ion movement voltage. This let them put 100s of thousands of memristors on a single chip. The aim is to do inference on the edge based on small devices rather than GPUs or supercomputers.

<http://news.mit.edu/2020/thousands-artificial-brain-synapses-single-chip-0608>



Short term impact on Machine Learning companies (tech/businesswise) :

Cloud infrastructure bill for an AI company is comparatively higher than what it should be for a SaaS company of the same scale and revenue. The reason is that they right now use GPUs a lot. There are two types of GPUs in usage :

A. Cheaper low power GPUs to deploy AI models. On cloud generally Nvidia T4/P4 are used. This is one area where alternative AI hardware has made inroads. TPUs are pretty common alternative on cloud here and devices like Nvidia Jetson nano , Google Coral and Intel Neural Compute Stick in IoT.

B. Costlier high powered GPUs for training AI algorithms. In fact, for training many large algorithms, you need a GPU cluster. OpenAI trained its recent GPT3 model on one of world’s most powerful GPU based supercomputers.

At least on paper for both A and B, TPUs seem to be very very cost effective. While my startup has not yet started adapting models to deploy on TPUs yet (we use cheap inference GPUs on cloud for deployment and in office large GPUs for training), AI blogs praise it as a effective device for both purposes in cloud. <https://medium.com/bigdatarepublic/cost-comparison-of-deep-learning-hardware-google-tpuv2-vs-nvidia-tesla-v100-3c63fe56c20f>

You can assume that in short run, deployment and training of AI will get cheaper with respect to model size. This will have two effects :

1. Low complexity models will start to be deployed for cases where they are assumed to be too costly right now.

2. Larger and more accurate models will be trained. At least for now, it seems “Bigger is Better” is true in Neural Networks.

*One requirement for all AI specific hardware platforms is to make the cost to switch very low. The more code needs to be rewritten to adapt software to the new platforms, the bigger gains in performance they will have to show to the people to make the investment to switch.*

1. Long Term impact of AI hardware – Making AutoML feasible
   1. [I was actually answering the question “Is the difference between AI specific hardware vs traditional GPU like manual shift and automatic shift in cars?”]

A good analogy for GPUs and AI specific hardware is IC Engine vs Electric Vehicle hardware. Its a new way of doing the same thing efficiently. For example, no changes in code need to be done to deploy a model trained on Nvidia GPUs and to make it work on Google's AI specific hardware(TPUs). You won't even feel the difference while working on an AI problem as an engineer, but the hardware that deploys code is more efficient and costs less. Free lunch in some ways (you do lose some flexibility of writing more generic programs with the general purpose hardware as I said earlier, but it doesn't matter for most AI algorithms). What is happening with the AI specific hardware is the AI programs we write burn less compute/context in getting translated to hardware code to run (because of some software instructions now directly runnable on hardware without translation) and hence more compute to run the programs themselves.

However, there are other technologies, which involve "AI making AI" and are apt to the automatic-shift vs Manual Shift analogy in AI. If you want to read more about these technologies, good terms to look for are *AutoML*, *Automated Machine Learning* (both these terms used in automation of traditional machine learning) and *Neural Architecture Search* (specific term for Deep Learning). These techniques are not really making writing AI easier for all engineers, but are making writing all AI easier for just a few smart engineers. So just like we are automating some parts of jobs of Software Developers, BPOs, web designers, cab drivers, we are automating low skill AI tasks too.

These AutoML techniques are still AI algorithms, which instead of solving a problem directly, learn to develop a problem solver. Just like some humans can better look at an image and describe it as compared to an AI algorithm, there are Machine Learning practitioners who can beat AutoMLtechniques easily as of now (Otherwise, AutoML would win all Machine Learning competitions). There is still art in smart Human Observation and Intuition which these algorithms are far from beating. What AutoML is good at right now is finding out good enough solutions, so you don't need an engineer for every single Data Learning problem. However, this is where cheap and efficient hardware (specially very efficient ones like Neurons on Chip hardware) can come in handy. One limiting factor of AutoML (even today) is compute. Efficient Hardware can give computers extreme power to do what computers are really good at for cheap money in the field of AutoML, that is, trial-and-error for learning what humans try to solve by wisdom and intuition. Apart from making AI technology cheaper, this is where I see the greatest potential of AI hardware, scaling up AutoML to actually start getting "superb" results from "good" results like right now.