Why are GPUs well-suited to deep learning?

9 Answers

Ottokar Tilk, PhD student at Tallinn University of Technology
Written Dec 8, 2014

Deep learning involves huge amount of matrix multiplications and other operations which can be massively parallelized and thus sped up on GPU-s.

A single GPU might have thousands of cores while a CPU usually has no more than 12 cores. Although GPU cores are slower than CPU cores, they more than make up for that with their large number and faster memory if the operations can be parallelized. Sequential code is still faster on CPUs.

Chomba Bupe, I develop machine learning algorithms
Updated Aug 7, 2016

I think there are already good answers here but I will use a different perspective to answer this question but I am literally saying the same thing. If you want to play a game, for example on PC, you can choose a low-end machine with just a CPU or a high-end machine with a CPU and a GPU. It is very much possible to play some games on low-end machines but the frame rate is quite low compared to the frame rates obtained on a high-end machine.

The GPU speeds up or accelerates graphical computations very well but both a CPU and a GPU can handle graphical operations only that the latter performs faster because of the distributed/parallel nature of the architecture with many low-end processing nodes.

The parallel architecture in a GPU is well adapted for vector and matrix operations. In 3D computer graphics there are a lot of such operations, like computation of lighting effects from normal maps, 3D effects e.t.c. GPUs were designed to handle such vector and matrix operations in parallel unlike a single core CPU that would handle matrix operations in serial form processing one element at a time.

This makes it possible to play games at 60 fps with impressive real-time visuals. Now coming back to deep learning, there are a lot of vector and matrix operations in deep learning as well, so it's intuitive that deep

learning should run several times faster on a GPU than a CPU just like a game runs faster on a GPU compared to a CPU.

So the mere reason why GPUs are well-suited to deep learning is because of the nature of computations they were designed to accelerate and handle happens to be the same has those encountered in deep learning. Thus deep learning algorithms run several times faster on a GPU compared to a CPU, learning times can be reduced from months to weeks or even a day.

This speed up is important because researchers or people working with deep learning would want to experiment with multiple deep learning architectures like the number of layers, cost functions, regularization methods e.t.c.

Deep learning doesn't usually work well the first time so a lot of adjustments need to be done. For every adjustment to the network, it must learn again. Now imagine waiting for months every time you tweak your network, that's not practical. You would want a faster way to train your network so you don't have to wait for months every time you adjust it.

The answer is in gaming, why not use a GPU to accelerate deep learning just like in computer graphics?

Hope this helps.

23.4k Views · View Upvotes · Answer requested by Vo Viet Anh

Tim Dettmers, Studying deep learning
Updated Feb 28 · Upvoted by Kah Seng Tay, Master's and Bachelor's in Computer Science. and Yuval Feinstein, Algorithmic Software Engineer in NLP,IR and Machine Learning

As many have said GPUs are so fast because they are so efficient for matrix multiplication and convolution, but nobody gave a real explanation why this is so. The real reason for this is memory bandwidth and not necessarily parallelism.

First of all you have to understand that CPUs are latency optimized while GPUs are bandwidth optimized. You can visualize this as a CPU being a Ferrari and a GPU being a big truck. The task of both is to pick up packages from a random location A and to transport those packages to another random location B. The CPU (Ferrari) can fetch some memory (packages) in your RAM quickly while the GPU (big truck) is slower in doing that (much higher latency). However, the CPU (Ferrari) needs to go back and forth many times to do

its job (location A -> pick up 2 packages -> location B ... repeat) while the GPU can fetch much more memory at once (location A -> pick up 100 packages -> location B ... repeat).

So in other words the CPU is good at fetching small amounts of memory quickly (5 * 3 * 7) while the GPU is good at fetching large amounts of memory (Matrix multiplication: (A*B)*C). The best CPUs have about 50GB/s while the best GPUs have 750GB/s memory bandwidth. So the larger your computational operations are in terms of memory, the larger the advantage of GPUs over CPUs. But there is still the latency that may hurt performance in the case of the GPU. A big truck may be able to pick up a lot of packages with each tour, but the problem is that you are waiting a long time until the next set of packages arrives. Without solving this problem GPUs would be very slow even for large amounts of data. So how is this solved?

If you ask a big truck to make a number of tours to fetch packages you will always wait for a long time for the next load of packages once the truck has departed to do the next tour — the truck is just slow. However, if you now use a fleet of either Ferraris and big trucks (thread parallelism), and you have a big job with many packages (large chunks of memory such as matrices) then you will wait for the first truck a bit, but after that you will have no waiting time at all, because unloading the packages takes so much time that all the trucks will queue in unloading location B so that you always have direct access to your packages (memory). This effectively hides latency so that GPUs offer high bandwidth while hiding their latency under thread parallelism — so for large chunks of memory GPUs provide the best memory bandwidth while having almost no drawback due to latency via thread parallelism. This is the second reason why GPUs are faster than CPUs for deep learning. As a side note, you will also see why more threads do not make sense for CPUs: A fleet of Ferraris has no real benefit in any scenario.

But the advantages for the GPU do not end here. This is the first step where the memory is fetched from the main memory (RAM) to the local memory on the chip (L1 cache and registers). This second step is less important for performance, but still adds to the lead for GPUs. All computation that ever is executed happens in registers which are directly attached to the execution unit (a core for CPUs, a stream processor for GPUs). Usually, you have the fast L1 and register memory very close to the execution engine and you want to keep these memories small, so that access is fast. Increased distance to the execution engine dramatically reduces memory access speed, so the larger the distance to access it the slower it gets. If you make your memory larger and larger, then in turn it gets slower to access its memory (on average, finding what you want to buy in a small store is faster than finding what you want to buy in a huge store, even if you know where that item is). So the size is limited for register files - we are just at the limits of physics here and every nanometer counts, we want to keep them small.

The advantage of the GPU is here, that it can have a small pack of registers for every processing unit (steam processor, or SM), of which it has many. Thus we can have in total a lot of register memory, which is very small and thus very fast. This leads to the aggregate GPU registers size being more than 30 times larger compared to CPUs and still twice as fast which translates to up to 14MB register memory that operates at a whooping 80TB/s. As a comparison, the CPU L1 cache only operates at about 5TB/s which is quite slow and has the size of roughly 1MB; CPU registers usually have sizes of around 64-128KB and operate at 10-20TB/s. Of course, this comparison of numbers is a bit flawed because registers operate a bit differently than GPU

registers (a bit like apples and oranges), but the difference in size here is more crucial than the difference in speed and it does make a difference.

As a side note, full register utilization in GPUs seems to be difficult to achieve at first because it is the smallest unit of computation which needs to be fine-tuned by hand for good performance. But NVIDIA has developed good compiler tools here which exactly indicate when you are using too much or too few registers per stream processor. It is easy to tweak your GPU code to make use of the right amount of registers and L1 cache for fast performance. This gives GPUs an advantage over other architectures like Xeon Phis where this utilization is difficult to achieve and difficult to debug which in the end makes it difficult to maximize performance on a Xeon Phi.

What this means in the end is that you can store a lot of data in your L1 caches and register files on GPUs to reuse convolutional and matrix multiplication tiles. For example the best matrix multiplication algorithms use 2 tiles of 64x32 to 96x64 numbers for 2 matrices in L1 cache, and a 16x16 to 32x32 number register tile for the outputs sums per thread block (1 thread block = up to 1024 threads; you have 8 thread blocks per stream processor, there are 60 stream processors in total for the entire GPU). If you have a 100MB matrix, you can split it up in smaller matrices that fit into your cache and registers, and then do matrix multiplication with three matrix tiles at speeds of 10-80TB/s — that is fast! This is the third reason why GPUs are so much faster than CPUs, and why they are so well suited for deep learning.

Keep in mind that the slower memory always dominates performance bottlenecks. If 95% of your memory movements take place in registers (80TB/s), and 5% in your main memory (0.75TB/s), then you still spend most of the time on memory access of main memory (about 6 times as much).

Thus in order of importance: (1) High bandwidth main memory, (2) hiding memory access latency under thread parallelism, and (3) large and fast register and L1 memory which is easily programmable are the components which make GPUs so well suited for deep learning.

Carlos E. Perez, Software Architect - Design Patterns for Deep Learning Architectures
Written Nov 18, 2015

CPUs are designed for more general computing workloads. GPUs in contrast are less flexible, however GPUs are designed to compute in parallel the same instructions. Deep Neural Networks (DNN) are structured in a very uniform manner such that at each layer of the network thousands of identical artificial neurons perform the same computation. Therefore the structure of a DNN fits quite well with the kinds of

computation that a GPU can efficiently perform.

GPUs have additional advantages over CPUs, these include having more computational units and having a higher bandwidth to retrieve from memory. Furthermore, in applications requiring image processing (i.e. Convolution Neural Networks) GPU graphics specific capabilities can be exploited to further speed up calculations.

The primary weakness of GPUs as compared to CPUs is memory capacity on GPUs are lower than CPUs. The highest known GPU contains 24GB of RAM, in contrast, CPUs can reach 1TB of RAM. A secondary weakness is that a CPU is required to transfer data into the GPU card. This takes place through the PCI-E connector which is much slower than CPU or GPU memory. One final weakness is GPU clock speeds are 1/3rd that of high end CPUs, so on sequential tasks a GPU is not expected to perform comparatively well.

In summary, GPUs work well with DNN computations because (1) GPUs have many more resources and faster bandwidth to memory (2) DNN computations fit well with GPU architecture. Computational speed is extremely important because training of Deep Neural Networks can range from days to weeks. In fact, many of the successes of Deep Learning may have not been discovered if it were not for the availability of GPUs.

Rohan Saxena, Computer Science undergrad at BITS Pilani
Written Mar 30

Deep learning consists of deep neural networks. These neural networks have lots of weights and biases, which are basically large matrices of floating point numbers.

Now consider your display system. It is basically built by operating on pixels and computing floating-point operations. Due to the demand for high-end graphics (movies, games, etc) the GPU industry had to pump up their throughput (the number of instructions a processor can execute in unit time). They did this in part by parallelising operations, that is, maintaining multiple execution lines which executed instructions.

Incidentally, deep learning also requires floating-point operations, many of which can be done parallelly. It was natural then, that GPUs were sighted quickly by deep learning enthusiasts.

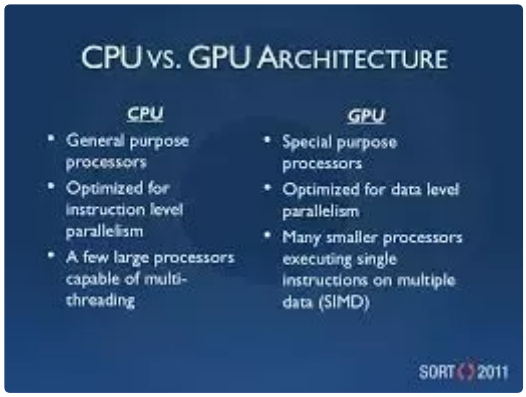

Image credits

Note: SIMD means that the processor executes a **s**ingle **i**nstruction on **m**ultiple **d**ata (useful in working with arrays and matrices).

354 Views · View Upvotes

Upvote 3 Downvote Comment

Yam Peleg, Deep learning researcher in a private sector laboratory.
Written Jun 9, 2016

Simple, GPU's are build specifically for matrix multiplications. It is the way 3d graphics are represented.

Deep learning at the end is represented by weight matrices and the learning phase is just some (alot) matrix multiplications.

3.6k Views · View Upvotes

Upvote 1 Downvote Comment

Roman Trusov, Facebook AI Research Intern 2016
Written Dec 8, 2014

Deep Learning requires A LOT of computations. It typically involved a neural network with a great amount of nodes, and every node has many connections, which need to be updated multiple times during the learning.
For example, simple backpropagation for 100 nodes on a single CPU core with 100000 iterations(which is sometimes not enough) requires about 15 minutes. When you have 100000 nodes and 1000000 stages...
At the meantime, GPUs have great clock rate comparing with CPUs. It can perform a tremendous amount of FLOPs in a second, which boosts the performance of the fitting enormously.

6.1k Views · View Upvotes

Upvote 15 Downvote Comment 1

Kevin Cameron, I like to program in C++ & Perl on Linux
Written Feb 12

They aren't.

However they are better than X86 type Intel processors that suck equally at everything.

GPUs are stream processors that handle (unidirectional) data flow problems well, and are good if you can fit your problem into their architecture - usually alternating banks of CPUs and memory. If you overrun their resources and need to swap data in and out of the banks to DRAM then performance drops off significantly, and there's limited support for doing that transparently, so you need to learn CUDA or OpenCL to program them.

Since they aren't particularly well suited there's a new crop of machines coming along that are designed specifically for AI tasks.

679 Views · View Upvotes

Upvote 2 Downvote Comment

Matthew Lai, Research Engineer @ Google DeepMind

Written Dec 6, 2015

They can multiply large matrices quickly. The performance bottleneck in deep learning is usually matrix multiplications.

3.9k Views · View Upvotes · Answer requested by Vo Viet Anh