Introduction to Deep Learning (IT3320E)

6 - Hardware and Software for Deep Learning

Hung Son Nguyen

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Sept. 29, 2022

Agenda

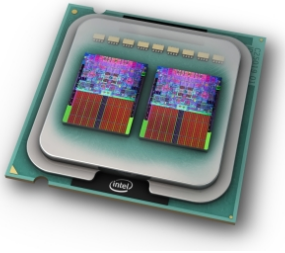
1. **Hardware for deep learning (CPU vs GPU)**
   - CPU vs. GPU comparison
2. **Deep learning frameworks**
3. **Accelerator and compression tools**

Section 1

Hardware for deep learning (CPU vs GPU)

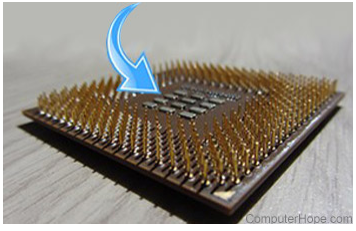
What are the differences between CPU and GPU?

- **CPU (central processing unit)** is a generalized processor that is designed to carry out a wide variety of tasks.
- **GPU (graphics processing unit)** is a specialized processing unit with enhanced mathematical computation capability, ideal for computer graphics and machine-learning tasks.



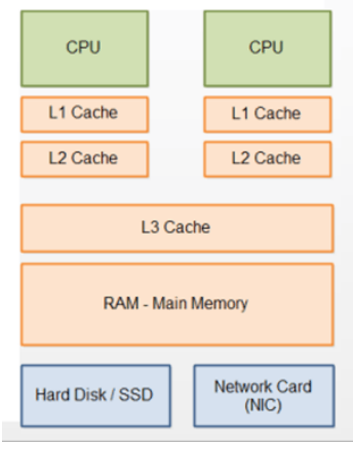
- CPU is the heart of any and every computer.
- The CPU handles the core processing tasks in a computer — the literal computation that drives every single action in a computer system.
- Computers work through the processing of binary data, or ones and zeroes.
- To translate that information into the software, graphics, animations, and every other process executed on a computer, those ones and zeroes must work through the logical structure of the CPU.
- That includes the basic arithmetic, logical functions (AND, OR, NOT) and input and output operations. The CPU is the brain, taking information, calculating it, and moving it where it needs to go.
- Within every CPU, there are a few standard components, which include the following: **Core, Cache, Memory Management Unit (MMU), CPU Clock and Control Unit.**

- A core, or CPU core, is the "brain" of a CPU. It receives instructions, and performs calculations, or operations, to satisfy those instructions. A CPU can have multiple cores.
- A core typically functions through what is called the "instruction cycle," where instructions are pulled from memory (fetch), decoded into processing language (decode), and executed through the logical gates of the core (execute).



- Initially, all CPUs were single-core, but with the proliferation of multi-core CPUs, we've seen an increase in processing power.
- As of 2019, the majority of consumer CPUs feature between two and twelve cores. Workstation and server CPUs may feature as many as 48 cores.

Cache is a small amount of memory which is a part of the CPU – closer to the CPU than RAM. It is used to temporarily hold instructions and data that the CPU is likely to reuse.



- Since CPUs work so fast to complete millions of calculations per second, they require ultra-fast (and expensive) memory to do it—memory that is much faster than hard drive storage or even the fastest RAM.
- In any CPU configuration, you will see some L1, L2, and/or L3 cache arrangement, with L1 being the fastest and L3 the slowest.
- The CPU will store the most immediately needed information in L1, and as the data loses priority, it will move out into L2, then L3, and then out to RAM or the hard disk.

- **Memory Management Unit (MMU):** The MMU controls data movement between the CPU and RAM during the instruction cycle.
- **CPU Clock and Control Unit:** Every CPU works on synchronizing processing tasks through a clock. The CPU clock determines the frequency at which the CPU can generate electrical pulses, its primary way of processing and transmitting data, and how rapidly the CPU can work. So, the higher the CPU clock rate, the faster it will run and quicker processor-intensive tasks can be completed.

- All these components work together to provide an environment where high-speed task parallelism can take place.
- As the CPU clock drives activities, the CPU cores switch rapidly between hundreds of different tasks per second. That's why your computer can run multiple programs, display a desktop, connect to the internet, and more all at the same time.
- The CPU is responsible for all activity on a computer.
  - When you close or open programs, the CPU must send the correct instructions to pull information from the hard drive and run executable code from RAM.
  - When playing a game, the CPU handles processing graphical information to display on the screen.
  - When compiling code, the CPU handles all the computation and mathematics involved.

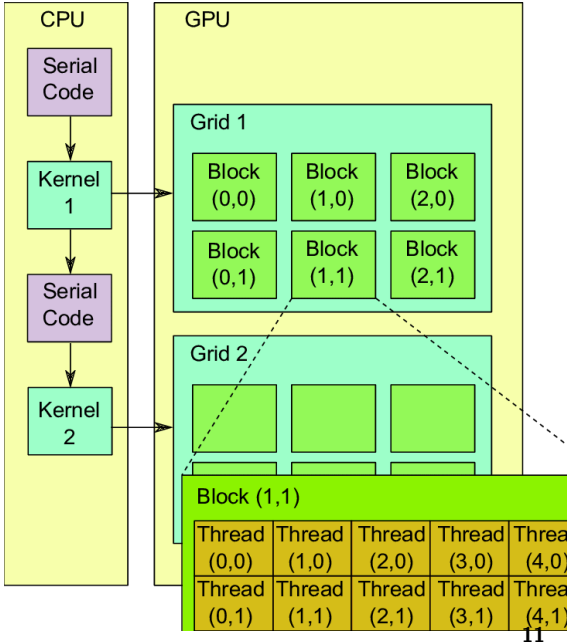
- One of these tasks, graphical processing, is generally considered one of the more complex processing tasks for the CPU. Solving that complexity has led to technology with applications far beyond graphics.
- The challenge in processing graphics is that graphics call on complex mathematics to render, and those complex mathematics must compute in parallel to work correctly.

- For example, a graphically intense video game might contain hundreds or thousands of polygons on the screen at any given time, each with its individual movement, color, lighting, and so on.
- CPUs aren't made to handle that kind of workload.
- That's where graphical processing units (GPUs) come into play.

- GPUs are similar in function to CPU: they contain cores, memory, and other components.
- Instead of emphasizing context switching to manage multiple tasks, GPU acceleration emphasizes parallel data processing through a large number of cores.
- These cores are usually less powerful individually than the core of a CPU. GPUs also typically have less interoperability with different hardware APIs and houseless memory.
- Where they shine is pushing large amounts of processed data in parallel.
- Instead of switching through multiple tasks to process graphics, a GPU simply takes batch instructions and pushes them out at high volume to speed processing and display.

- **Flexibility:** CPUs are flexible and resilient and can handle a variety of tasks outside of graphics processing. Because of their serial processing capabilities, the CPU can multitask across multiple activities in your computer. Because of this, a strong CPU can provide more speed for typical computer use than a GPU.
- **Contextual Power:** In specific situations, the CPU will outperform the GPU. For example, the CPU is significantly faster when handling several different types of system operations (random access memory, mid-range computational operations, managing an operating system, I/O operations).
- **Precision:** CPUs can work on mid-range mathematical equations with a higher level of precision. CPUs can handle the computational depth and complexity more readily, becoming increasingly crucial for specific applications.

- **Access to Memory:** CPUs usually contain significant local cache memory, which means they can handle a larger set of linear instructions and, hence, more complex system and computational operations.
- **Cost and Availability:** CPUs are more readily available, more widely manufactured, and cost-effective for consumer and enterprise use. Additionally, hardware manufacturers still create thousands of motherboard designs to house a wide range of CPUs.



- **Parallel Processing:** CPUs cannot handle parallel processing like a GPU, so large tasks that require thousands or millions of identical operations will choke a CPU's capacity to process data.
- **Slow Evolution:** In line with Moore's Law, developing more powerful CPUs will eventually slow, which means less improvement year after year. The expansion of multi-core CPUs has mitigated this somewhat.
- **Compatibility:** Not every system or software is compatible with every processor. For example, applications written for x86 Intel Processors will not run on ARM processors. This is less of a problem as more computer manufacturers use standard processor sets (see Apple's move to Intel processors), but it still presents issues between PCs and mobile devices.

Some of the advantages of a GPU include the following:

- **High Data Throughput:** a GPU consist of hundreds of cores performing the same operation on multiple data items in parallel. Because of that, a GPU can push vast volumes of processed data through a workload, speeding up specific tasks beyond what a CPU can handle.
- **Massive Parallel Computing:** Whereas CPUs excel in more complex computations, GPUs excel in extensive calculations with numerous similar operations, such as computing matrices or modeling complex systems.

These two advantages were the main reasons GPUs were created because both contribute to complex graphics processing. However, the GPU structure quickly led developers and engineers to apply GPU technology to other high-performance applications:

- **Bitcoin Mining:** The process of mining bitcoins involves using computational power to solve complex cryptographic hashes. The increasing expansion of Bitcoin and the difficulty of mining bitcoins has led bitcoin mines to implement a GPU to handle immense volumes of cryptographic data in the hopes of earning bitcoins.
- **Machine Learning:** Neural networks, particularly those used for deep-learning algorithms, function through the ability to process large amounts of training data through smaller nodes of operations. GPUs for machine learning have emerged to help process the enormous data sets used to train machine-learning algorithms and AI.
- **Analytics and Data Science:** GPUs are uniquely suited to help analytics programs process large amounts of base data from different sources. Furthermore, these same GPUs can power the computation necessary for deep data sets associated with research areas like life sciences (genomic sequencing).

- **Multitasking:** GPUs aren't built for multitasking, so they don't have much impact in areas like general-purpose computing.
- **Cost:** While the price of GPUs has fallen somewhat over the years, they are still significantly more expensive than CPUs. This cost rises more when talking about a GPU built for specific tasks like mining or analytics.
- **Power and Complexity:** While a GPU can handle large amounts of parallel computing and data throughput, they struggle when the processing requirements become more chaotic. Branching logic paths, sequential operations, and other approaches to computing impede the effectiveness of a GPU.

TPU stands for tensor processing unit and is a designated architecture for deep learning or machine learning applications.

- Invented by Google, TPUs are application-specific integrated circuits (ASIC) designed specifically to handle the computational demands of machine learning and accelerate AI calculations and algorithms.
- Google began using TPUs internally in 2015, and in 2018 they made them publicly available to others.
- When Google designed the TPU, they created a domain-specific architecture. What that means is that instead of designing a general purpose processor like a GPU or CPU, Google designed it as a matrix processor that was specialized for neural network work loads.
- By designing the TPU as a matrix processor instead of a general purpose processor, Google solved the memory access problem that slows down GPUs and CPUs and requires them to use more processing power.

- Here's how a TPU works:
  - TPU loads the parameter from memory into the matrix of multipliers and adders.
  - TPU loads the data from memory.
  - As multiplications are executed, their results are passed on to the next multipliers while simultaneously taking summation at the same time.
- The output from these steps will be whatever the summation of all the multiplication results is between the data and parameters.
- No memory access at all is required throughout the entire process of these massive calculations and data passing.

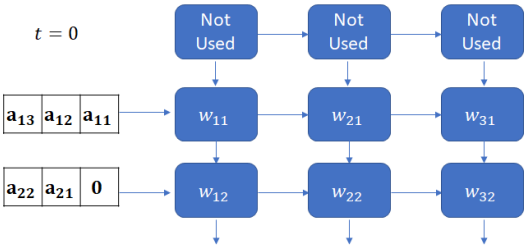
Consider product of $N \times M$ and $M \times N$ matrices, where $N > M$. Product has dimensions $N \times N$. Let's consider $N = 3, M = 2$

w_{11}	w_{12}
w_{21}	w_{22}
w_{31}	w_{32}

✖

a_{11}	a_{12}	a_{13}
a_{21}	a_{22}	a_{23}

$y_{11} = w_{11}a_{11} + w_{12}a_{21}$ etc



2N-1

$t = 3$ y^a_{13}

$t = 4$ y^a_{12} y^a_{23}

$t = 5$ y^a_{11} y^a_{22} y^a_{33}

$t = 6$ y^a_{21} y^a_{32}

$t = 7$ y^a_{31}

# of clock cycles between when

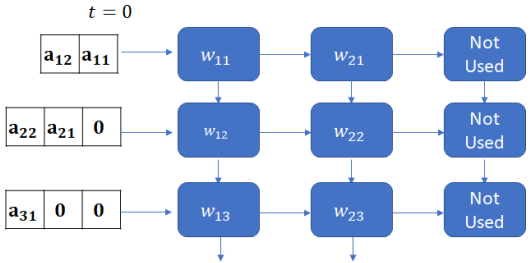
Now consider the case where N < M. Product has dimensions $N \times N$. Let's consider $N = 2, M = 3$

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}

✖

a_{11}	a_{12}
a_{21}	a_{22}
a_{31}	a_{32}

$y_{11} = w_{11}a_{11} + w_{12}a_{21} + w_{13}a_{31}$



2N-1

$t = 4$ y^a_{12}

$t = 5$ y^a_{11} y^a_{22}

$t = 6$ y^a_{21}

# of clock cycles between when the first activation is fed into the systolic array ($t =$

- TPUs are extremely valuable and bring a lot to the table. Their only real downside is that they are more expensive than GPUs and CPUs.
- Their list of pros highly outweighs their high price tag.
- TPUs are a great choice for those who want to:
  - Accelerate machine learning applications
  - Scale applications quickly
  - Cost effectively manage machine learning workloads
  - Start with well-optimized, open source reference models

	Cores	Clock Speed	Memory	Price	Speed
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	$385	~540 GFLOPs FP32
GPU (NVIDIA RTX 2080 Ti)	3584	1.6 GHz	11 GB GDDR6	$1199	~13.4 TFLOPs FP32
TPU NVIDIA TITAN V	5120 CUDA, 640 Tensor	1.5 GHz	12GB HBM2	$2999	~14 TFLOPs FP32 ~112 TFLOP FP16
TPU Google Cloud TPU	?	?	64 GB HBM	$4.50 per hour	~180 TFLOP

Section 2

Deep learning frameworks

https://github.com/397090770/spark-ai-summit-europe-2018-10/blob/master/ppt/a-tale-of-three-deep-learning-frameworks-tensorflow-keras-pytorch-with-br pdf

- Keras is an effective high-level neural network Application Programming Interface (API) written in Python. This open-source neural network library is designed to provide fast experimentation with deep neural networks, and it can run on top of CNTK, TensorFlow, and Theano.
- Keras focuses on being modular, user-friendly, and extensible. It doesn't handle low-level computations; instead, it hands them off to another library called the Backend.
- Keras was adopted and integrated into TensorFlow in mid-2017. Users can access it via the tf.keras module. However, the Keras library can still operate separately and independently.

What is PyTorch?

- PyTorch is a relatively new deep learning framework based on Torch.
- Developed by Facebook's AI research group and open-sourced on GitHub in 2017, it's used for natural language processing applications.
- PyTorch has a reputation for simplicity, ease of use, flexibility, efficient memory usage, and dynamic computational graphs.
- It also feels native, making coding more manageable and increasing processing speed.

- TensorFlow is an end-to-end open-source deep learning framework developed by Google and released in 2015. It is known for documentation and training support, scalable production and deployment options, multiple abstraction levels, and support for different platforms, such as Android.
- TensorFlow is a symbolic math library used for neural networks and is best suited for dataflow programming across a range of tasks. It offers multiple abstraction levels for building and training models.
- A promising and fast-growing entry in the world of deep learning, TensorFlow offers a flexible, comprehensive ecosystem of community resources, libraries, and tools that facilitate building and deploying machine learning apps. Also, as mentioned before, TensorFlow has adopted Keras, which makes comparing the two seem problematic. Nevertheless, we will still compare the two frameworks for the sake of completeness, especially since Keras users don't necessarily have to use TensorFlow.

- Both TensorFlow and PyTorch offer useful abstractions that ease the development of models by reducing boilerplate code. They differ because PyTorch has a more "pythonic" approach and is object-oriented, while TensorFlow offers a variety of options.
- PyTorch is used for many deep learning projects today, and its popularity is increasing among AI researchers, although of the three main frameworks, it is the least popular. Trends show that this may change soon.
- When researchers want flexibility, debugging capabilities, and short training duration, they choose PyTorch. It runs on Linux, macOS, and Windows.

- Thanks to its well-documented framework and abundance of trained models and tutorials, TensorFlow is the favorite tool of many industry professionals and researchers. TensorFlow offers better visualization, which allows developers to debug better and track the training process. PyTorch, however, provides only limited visualization.
- TensorFlow also beats PyTorch in deploying trained models to production, thanks to the TensorFlow Serving framework. PyTorch offers no such framework, so developers need to use Django or Flask as a back-end server.
- In the area of data parallelism, PyTorch gains optimal performance by relying on native support for asynchronous execution through Python. However, with TensorFlow, you must manually code and optimize every operation run on a specific device to allow distributed training. In summary, you can replicate everything from PyTorch in TensorFlow; you just need to work harder at it.
- If you're just starting to explore deep learning, you should learn PyTorch first due to its popularity in the research community. However, if you're familiar with machine learning and deep learning and focused on getting a job in the industry as soon as possible, learn TensorFlow first.

- Both of these choices are good if you're just starting to work with deep learning frameworks. Mathematicians and experienced researchers will find PyTorch more to their liking. Keras is better suited for developers who want a plug-and-play framework that lets them build, train, and evaluate their models quickly. Keras also offers more deployment options and easier model export.
- However, remember that PyTorch is faster than Keras and has better debugging capabilities.
- Both platforms enjoy sufficient levels of popularity that they offer plenty of learning resources. Keras has excellent access to reusable code and tutorials, while PyTorch has outstanding community support and active development.
- Keras is the best when working with small datasets, rapid prototyping, and multiple back-end support. It's the most popular framework thanks to its comparative simplicity. It runs on Linux, MacOS, and Windows.

- TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high-level APIs used for easily building and training models, but Keras is more user-friendly because it's built-in Python.
- Researchers turn to TensorFlow when working with large datasets and object detection and need excellent functionality and high performance. TensorFlow runs on Linux, MacOS, Windows, and Android. The framework was developed by Google Brain and currently used for Google's research and production needs.
- The reader should bear in mind that comparing TensorFlow and Keras isn't the best way to approach the question since Keras functions as a wrapper to TensorFlow's framework. Thus, you can define a model with Keras' interface, which is easier to use, then drop down into TensorFlow when you need to use a feature that Keras doesn't have, or you're looking for specific TensorFlow functionality. Thus, you can place your TensorFlow code directly into the Keras training pipeline or model.
- At the end of the day, use TensorFlow machine learning applications and Keras for

	Keras	**PyTorch**	**TensorFlow**
API Level	High	Low	High and Low
Architecture	Simple, concise, readable	Complex, less readable	Not easy to use
Datasets	Smaller datasets	Large datasets, high performance	Large datasets, high performance
Debugging	Simple network, debugging is not often needed	Good debugging capabilities	Difficult to conduct debugging
Does It Have Trained Models?	Yes	Yes	Yes
Popularity	Most popular	Third most popular	Second most popular
Speed	Slow, low performance	Fast, high-performance	Fast, high-performance
Written In	Python	Lua	C++, CUDA, Python

Section 3

Accelerator and compression tools