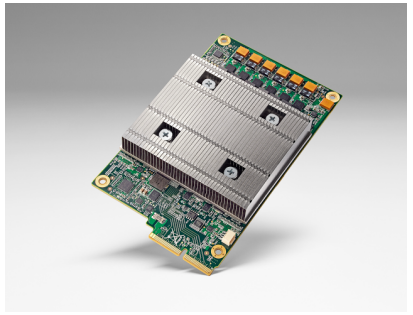
Hardware for Machine Learning

CS6787 Lecture 11 — Fall 2017

Recap: modern ML hardware

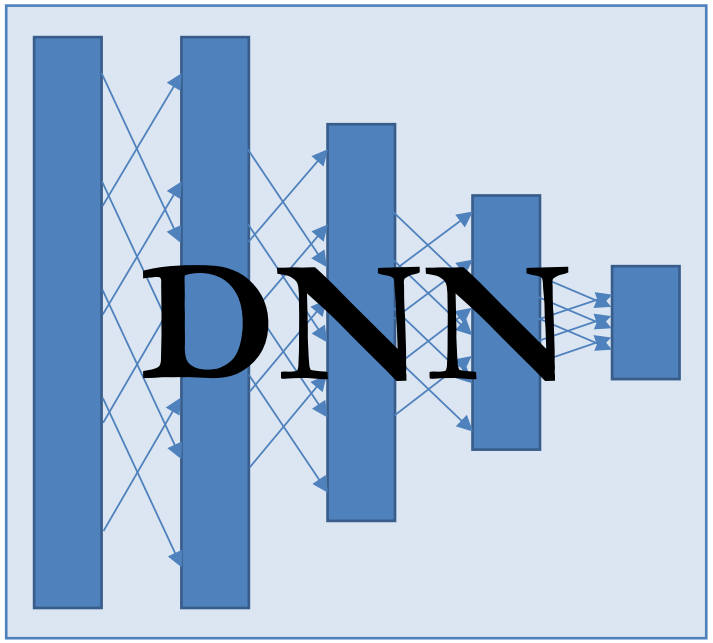
- **Lots of different types**
  - CPUs
  - GPUs
  - FPGAs
  - Specialized accelerators



- Right now, **GPUs are dominant**…we'll get to why later

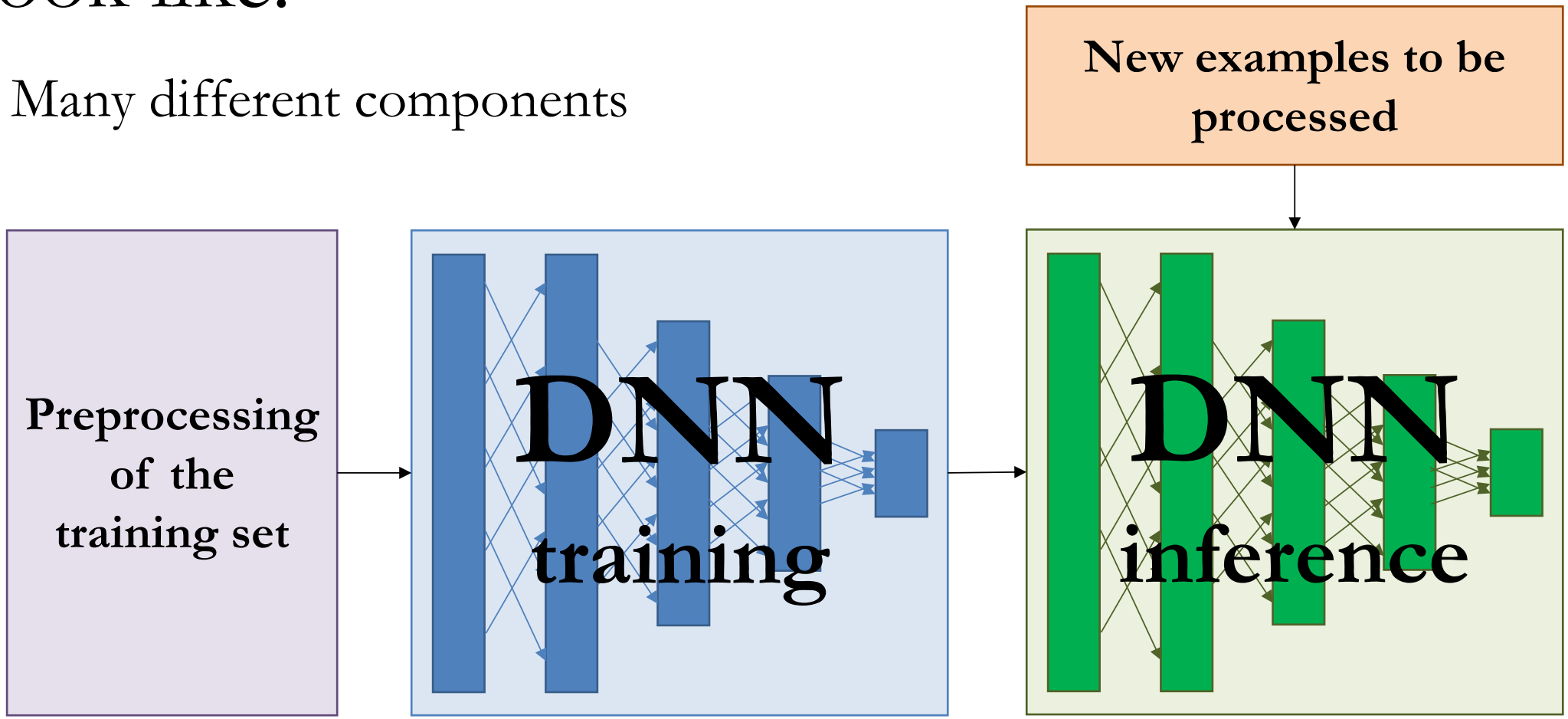
What does a modern machine learning pipeline look like?

- Not just a neural network:



What does a modern machine learning pipeline look like?

- Many different components



Where can hardware help?

- **Everywhere!**

- There's interest in using hardware everywhere in the pipeline
  - both **adapting existing hardware architectures**, and
  - **developing new ones**

- What improvements can we get?
  - **Lower latency inference**
  - **Higher throughput training**
  - **Lower power cost**

How can hardware help? Three ways

- Speed up the **basic building blocks** of machine learning computation
  - Major building block: **matrix-matrix multiply**
  - Another major building block: **convolution**

- Add **data/memory paths specialized** to machine learning workloads
  - Example: having a local cache to store network weights

- Create **application-specific functional units**
  - Not for general ML, but for a specific domain or application

Why are GPUs so popular for machine learning?

Why are GPUs so popular for *training deep neural networks*?

GPU vs CPU

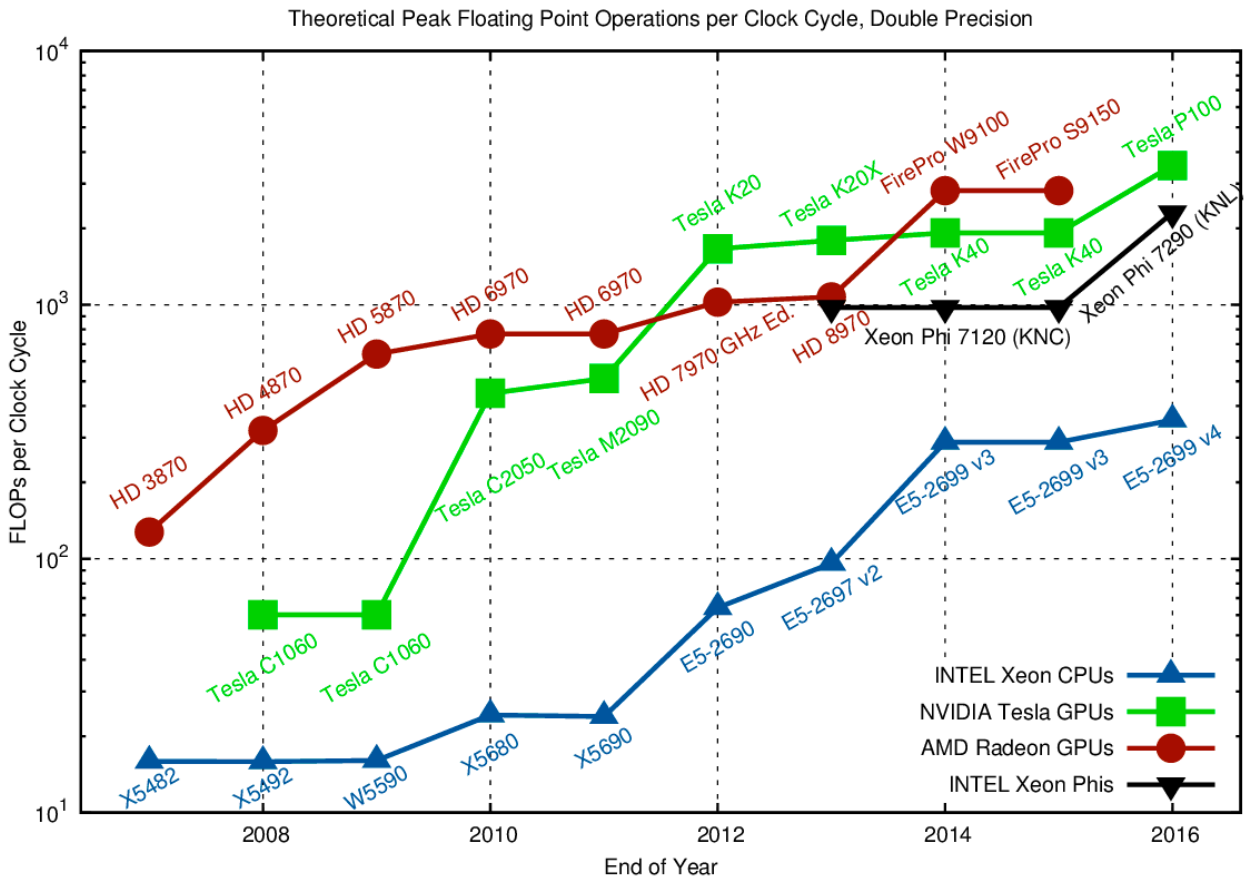
- **CPU is a general purpose processor**
  - Modern CPUs spend most of their area on deep caches
  - This makes the CPU a great choice for applications with random or non-uniform memory accesses

- **GPU is optimized for**
  - more compute intensive workloads
  - streaming memory models

Machine learning applications look more like this

FLOPS: GPU vs CPU

- FLOPS: **f**loating **p**oint **o**perations **p**er **s**econd



GPU FLOPS consistently exceed CPU FLOPS

Intel Xeon Phi chips are compute-heavy manycore processors that compete with GPUs

From Karl Rupp's blog https://www.karlrupp.net/2016/08/flops-per-cycle-for-cpus-gpus-and-xeon-phis/

This was the best diagram I could find that shows trends over time.

Memory bandwidth: CPU vs GPU

- GPUs have **higher memory bandwidths** than CPUs
  - E.g. new NVIDIA Tesla V100 has a claimed **900 GB/s memory bandwidth**
  - Wheras Intel Xeon E7 has only about **100 GB/s memory bandwidth**

- But, this **comparison is unfair!**
  - GPU memory bandwidth is the bandwidth to GPU memory
  - E.g. on a PCIE2, bandwidth is only **32 GB/s for a GPU**

What limits deep learning?

- **Is it compute bound or memory bound?**

- Ideally: it's **compute bound**
  - Why? Matrix-matrix multiply takes $\mathbf{O(n^2)}$ memory but $\mathbf{O(n^3)}$ compute

- Sometimes it is memory/communication bound
  - Especially when we are running at **large scale on a cluster**

Challengers to the GPU

- More **compute-intensive CPUs**
  - Like Intel's Phi line — promise same level of compute performance and better handling of sparsity
- **Low-power devices**
  - Like mobile-device-targeted chips
  - Configurable hardware like FPGAs and CGRAs
- Accelerators that **speed up matrix-matrix multiply**
  - Like Google's TPU

Will all computation become dense matrix-matrix multiply?

Deep learning and matrix-matrix multiply

- Traditionally, the most costly operation for deep learning for both training and inference is dense **matrix-matrix multiply**

- Matrix-matrix multiply at $\mathbf{O(n^3)}$ scales worse than other operations
  - So should expect it to **become even more of a bottleneck** as problems scale

- Deep learning is **still exploding** and capturing more compute cycles
  - Motivates the question: **will most computation in the future become dense matrix-matrix multiply?**

What if dense matrix multiply takes over?

- Great opportunities for **new highly specialized hardware**
  - The TPU is already an example of this
  - It's a glorified matrix-matrix multiply engine
- **Significant power savings** from specialized hardware
  - But not as much as if we could use something like sparsity
- It might put us all out of work
  - Who cares about researching algorithms when there's **only one algorithm anyone cares about?**

What if matrix multiply doesn't take over?

- Great opportunities for designing new **heterogeneous, application-specific hardware**
  - We might want one chip for SVRG, one chip for low-precision
- Interesting systems/framework opportunities to give users **suggestions for which chips to use**
  - Or even to **automatically dispatch work** within a heterogeneous datacenter
- **Community might fragment**
  - Into smaller subgroups working on particular problems

The truth is somewhere in the middle

- We'll probably see **both**
  - a lot of dense matrix-matrix multiply compute
  - a lot of opportunities for faster more specialized compute
- New models are being developed every day
  - And we **shouldn't read too much into** the current trends
- But this means we get the **best of both worlds**
  - We can do research on either side and still have impact

Recent work on hardware for machine learning

Abstracts from papers at architecture conferences this year

Questions?

- Conclusion
  - Lots of interesting work on hardware for machine learning
  - Lots of **opportunities for interdisciplinary research**

- Upcoming things
  - Paper Review #10 — **due today**
  - **Project proposal — due today**
  - Paper Presentation #11 **on Wednesday** — TPU