ACCELERATED COMPUTING: REDUCE LATENCY OF IDEA GENERATION

Research as a sequential, cyclic process

Limit: Ingenuity

Limit: Programmability

Limit: Throughput
WHY IS DEEP LEARNING SUCCESSFUL

Big data sets

New algorithms
Computing hardware

Focus of this talk

Accuracy

Deep Learning

Many previous methods

Data & Compute

@ctnzr
MORE COMPUTE: MORE AI
https://blog.openai.com/ai-and-compute/
DEEP NEURAL NETWORKS 101

Simple, powerful function approximators

\[ y_j = f \left( \sum_i w_{ij} x_i \right) \]

One layer: nonlinearity \( \circ \) linear combination

\[ f(x) = \begin{cases} 
0, & x < 0 \\
0, & x \geq 0 
\end{cases} \]

nonlinearity
TRAINING NEURAL NETWORKS

\[ y_j = f \left( \sum_i w_{ij} x_i \right) \]

Computation dominated by dot products
Multiple inputs, multiple outputs, batch means it is compute bound

Train one model: 20+ Exaflops
LAWS OF PHYSICS
Successful AI uses Accelerated Computing

20X gap and growing...
20X in 10 years

Volta

General Purpose Performance
Accelerated Performance
MATRX MULIPLICATION

Thor’s hammer

\[ \begin{array}{c|c}
\text{m} & \text{k} \\
\hline
\text{n} & \text{m} \\
\end{array} \]

\[ O(n^2) \] communication

\[ O(n^3) \] computation
TENSOR CORE
Mixed Precision Matrix Math
4x4 matrices

\[
D = AB + C
\]

FP16 or FP32  FP16  FP16  FP16 or FP32

A\_0,0 A\_0,1 A\_0,2 A\_0,3
A\_1,0 A\_1,1 A\_1,2 A\_1,3
A\_2,0 A\_2,1 A\_2,2 A\_2,3
A\_3,0 A\_3,1 A\_3,2 A\_3,3

B\_0,0 B\_0,1 B\_0,2 B\_0,3
B\_1,0 B\_1,1 B\_1,2 B\_1,3
B\_2,0 B\_2,1 B\_2,2 B\_2,3
B\_3,0 B\_3,1 B\_3,2 B\_3,3

C\_0,0 C\_0,1 C\_0,2 C\_0,3
C\_1,0 C\_1,1 C\_1,2 C\_1,3
C\_2,0 C\_2,1 C\_2,2 C\_2,3
C\_3,0 C\_3,1 C\_3,2 C\_3,3

@ctnizr
**CHUNKY INSTRUCTIONS AMORTIZE OVERHEAD**

Taking advantage of that $O(n^3)$ goodness

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy**</th>
<th>Overhead*</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFMA</td>
<td>1.5pJ</td>
<td>2000%</td>
</tr>
<tr>
<td>HDP4A</td>
<td>6.0pJ</td>
<td>500%</td>
</tr>
<tr>
<td>HMMA</td>
<td>110pJ</td>
<td>27%</td>
</tr>
</tbody>
</table>

*(Overhead is instruction fetch, decode, and operand fetch – 30pJ)

**Energy numbers from 45nm process

Tensor cores yield efficiency benefits, but are still programmable

Bill Dally
TESLA V100

21B transistors
815 mm²

80 SM*
5120 CUDA Cores
640 Tensor Cores

32 GB HBM2
900 GB/s HBM2
300 GB/s NVLink

*full GV100 chip contains 84 SMs
## GPU PERFORMANCE COMPARISON

<table>
<thead>
<tr>
<th></th>
<th>P100</th>
<th>V100</th>
<th>Ratio</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training acceleration</td>
<td>10 TFLOPS</td>
<td>120 TFLOPS</td>
<td>12x</td>
<td>65 TFLOPS</td>
</tr>
<tr>
<td>Inference acceleration</td>
<td>20 TFLOPS</td>
<td>120 TFLOPS</td>
<td>6x</td>
<td>130 TOPS</td>
</tr>
<tr>
<td>FP64/FP32</td>
<td>5/10 TFLOPS</td>
<td>7.5/15 TFLOPS</td>
<td>1.5x</td>
<td>0.25/8 TFLOPS</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>720 GB/s</td>
<td>900 GB/s</td>
<td>1.2x</td>
<td>320 GB/s</td>
</tr>
<tr>
<td>NVLink Bandwidth</td>
<td>160 GB/s</td>
<td>300 GB/s</td>
<td>1.9x</td>
<td>--</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>4 MB</td>
<td>6 MB</td>
<td>1.5x</td>
<td>4 MB</td>
</tr>
<tr>
<td>L1 Caches</td>
<td>1.3 MB</td>
<td>10 MB</td>
<td>7.7x</td>
<td>6 MB</td>
</tr>
<tr>
<td>Power</td>
<td>250 W</td>
<td>300 W</td>
<td>1.2x</td>
<td>70 W</td>
</tr>
</tbody>
</table>
Turing follows Volta (Tesla T4, Titan RTX)
Includes lower precision tensor cores
(Not shown: 1 bit @ 128X throughput)
New solvers, new layers, new scaling techniques, new applications for old techniques, and much more…
Computation dominated by linear operations

But the research happens elsewhere:

- New loss functions
- New non-linearities
- New normalizations
- New inputs & outputs

CUDA is fast and flexible parallel C++
REFINING CUDA: CUDA GRAPHS
Latency & Overhead Reductions

Launch latencies:

- CUDA 10.0 takes at least 2.2us CPU time to launch each CUDA kernel on Linux
- Pre-defined graph allows launch of any number of kernels in one single operation

Useful for small models
Works with JIT graph compilers
CUDA LIBRARIES

Optimized Kernels

CUBLAS: Linear algebra
  Many flavors of GEMM

CUDNN: Neural network kernels
  Convolutions (direct, Winograd, FFT)
  Can achieve > Speed of Light!
  Recurrent Neural Networks

Lowering Convolutions to GEMM
IMPROVED HEURISTICS FOR CONVOLUTIONS

cuDNN 7.4.1 (Nov 2018) vs. cuDNN 7.0.5 (Dec 2017)

Speedup of unique cuDNN convolutions calls for the SSD detector model

Speedup: 100.0x, 10.0x, 1.0x, 0.1x

Batch=32, Batch=128, Batch=256

Unique cuDNN convolution API calls
PERSISTENT RNN SPEEDUP ON V100

cuDNN 7.4.1 (Nov 2018) vs. cuDNN 7.0.5 (Dec 2017)

Speedup of unique cuDNN Persistent RNN calls for GNMT @ batch=32
TENSORCORES WITH FP32 MODELS

cuDNN 7.4.1 (Nov 2018) vs. cuDNN 7.0.5 (Dec 2017)

- Enabled as an experimental feature in the TensorFlow NGC Container via an environment variable (same for cuBLAS)
- Should use in conjunction with Loss Scaling

Average speedup of unique cuDNN convolution calls during training

- Batch=32 Resnet-50 v1.5
- Batch=128 SSD
- Batch=32 Mask-RCNN
- Batch=128

Speedup

0.0x 0.5x 1.0x 1.5x 2.0x 2.5x 3.0x 3.5x
NVIDIA DGX-2

1. NVIDIA Tesla V100 32GB
2. Two GPU Boards
   - 8 V100 32GB GPUs per board
   - 6 NVSwitches per board
   - 512GB Total HBM2 Memory interconnected by Plane Card
3. Twelve NVSwitches
   - 2.4 TB/sec bi-section bandwidth
4. Eight EDR Infiniband/100 GigE
   - 1600 Gb/sec Total Bi-directional Bandwidth
5. PCIe Switch Complex
6. Two Intel Xeon Platinum CPUs
7. 1.5 TB System Memory
8. 30 TB NVME SSDs
   - Internal Storage
9. Dual 10/25 GigE
NVSWITCH: NETWORK FABRIC FOR AI

- 2.4 TB/s bisection bandwidth
- Equivalent to a PCIe bus with 1,200 lanes
- Inspired by leading edge research that demands unrestricted model parallelism
  - Each GPU can make random reads, writes and atomics to each other GPU’s memory
- 18 NVLink ports per switch
DGX-2: ALL-TO-ALL CONNECTIVITY

Each switch connects to 8 GPUs
Each GPU connects to 6 switches
Each switch connects to the other half of the system with 8 links
2 links on each switch reserved
FRAMEWORKS

Several AI frameworks
Let researchers prototype rapidly
Different perspectives on APIs
All are GPU accelerated
AUTOMATIC MIXED PRECISION
Four Lines of Code => 2.3x Training Speedup in PyTorch (RN-50)

- Mixed precision training uses half-precision floating point (FP16) to accelerate training
- You can start using mixed precision today with four lines of code
- This example uses AMP: Automatic Mixed Precision, a PyTorch library
  - No hyperparameters changed

```python
+ amp_handle = amp.init()
# ... Define model and optimizer for x, y in dataset:
    prediction = model(x)
    loss = criterion(prediction, y)
- loss.backward()
+ with amp_handle.scale_loss(loss, optimizer) as scaled_loss:
+   scaled_loss.backward()
optimizer.step()
```
AUTOMATIC MIXED PRECISION
Four Lines of Code => 2.3x Training Speedup (RN-50)

- Real-world single-GPU runs using default PyTorch ImageNet example
  - NVIDIA PyTorch 18.08-py3 container
  - AMP for mixed precision
- Minibatch=256
- Single GPU RN-50 speedup for FP32 -> M.P. (with 2x batch size):  
  - MxNet: 2.9x
  - TensorFlow: 2.2x
  - TensorFlow + XLA: ~3x
  - PyTorch: 2.3x
- Work ongoing to bring to 3x everywhere
DATA LOADERS

Fast training means greater demands on the rest of the system

Data transfer from storage (network)

CPU bottlenecks happen fast

GPU accelerated, user defined data loaders

Move decompression & augmentation to GPU

Both for still images and videos

Move all this to the GPU with DALI: [https://github.com/NVIDIA/DALI](https://github.com/NVIDIA/DALI)

Research video data loader using HW decoding: NVVL: [https://github.com/NVIDIA/NVVL](https://github.com/NVIDIA/NVVL)
Many important AI tasks involve agents interacting with the real world

For this, you need simulators

- Physics
- Appearance

Simulation has a big role to play in AI progress

RL needs good simulators - NVIDIA PhysX is now open source:

https://github.com/NVIDIAGameWorks/PhysX-3.4
MAKE INGENUITY THE LIMITING FACTOR

Accelerated Computing for AI

High computational intensity +
Programmability & flexibility fundamental for AI systems
Need a systems approach
Chips are not enough
And lots of software to make it all useful

Bryan Catanzaro
@ctnzr