Lecture: Commercial Efforts, Training

• Topics: training algorithm requirements, NVIDIA Volta (inference and training), Intel NNP-T/I, Graphcore (training)
BackProp with Stochastic Gradient Descent

\[
\begin{bmatrix}
1 & 2 & 3 & 4
\end{bmatrix}
\begin{bmatrix}
1 & 5 \\
2 & 6 \\
3 & 7 \\
4 & 8
\end{bmatrix}
= \begin{bmatrix}
30 \\
70
\end{bmatrix}
\]
BackProp with Stochastic Gradient Descent

\[
\begin{bmatrix}
30 & 70
\end{bmatrix}
\times
\begin{bmatrix}
1 & 3 & 5 \\
2 & 4 & 6
\end{bmatrix}
= \begin{bmatrix}
170 \\
370 \\
570
\end{bmatrix}
\]
BackProp with Stochastic Gradient Descent

$$\begin{bmatrix} 30 & 70 \end{bmatrix} \times \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 170 & 370 & 570 \end{bmatrix}$$

Expected output $t = \begin{bmatrix} 170 & 570 & 560 \end{bmatrix}$

Error $C = \frac{1}{2} (0 + 200^2 + 100)$

Error $= \frac{dC}{dy} = y-t$

$= 0$

$= -200$

$= 10$
BackProp with Stochastic Gradient Descent

\[ d_L = d_{L+1} W_{L+1}^T \]

Error = \( \frac{dC}{dy} = y - t \)
\[
\begin{align*}
\text{Error} &= 0 \\
\text{Error} &= -200 \\
\text{Error} &= 10
\end{align*}
\]

\[
\begin{bmatrix}
30 & 70
\end{bmatrix}
\times
\begin{bmatrix}
1 & 3 & 5 \\
2 & 4 & 6
\end{bmatrix}
=
\begin{bmatrix}
170 & 370 & 570
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & -200 & 10
\end{bmatrix}
\times
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
5 & 6
\end{bmatrix}
=
\begin{bmatrix}
-550 & -740
\end{bmatrix}
\]
BackProp with Stochastic Gradient Descent

\[ d_L = d_{L+1} W_{L+1}^T \]

Error = \( dC/dy = y - t \)

\[ = 0 \]
\[ = -200 \]
\[ = 10 \]

\[
\begin{bmatrix}
30 \\
70
\end{bmatrix} \times
\begin{bmatrix}
1 & 3 & 5 \\
2 & 4 & 6
\end{bmatrix}
= \begin{bmatrix}
170 \\
370 \\
570
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & -200 & 10
\end{bmatrix} \times
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
5 & 6
\end{bmatrix}
= \begin{bmatrix}
-550 \\
-740
\end{bmatrix}
\]

dC/dw_L = dC/dy \times dy/dw = a_{L-1} d_L
Take-Homes

• Create a mini-batch which is a random set of training inputs; apply the fwd pass; compute error; backprop the errors (while using a transpose of weights); compute the deltas for all the weights (using the Xs of previous layer); aggregate the deltas and update the weights at the end of the mini-batch; keep creating mini-batches until you run out of training samples; this is one epoch.

• On the fwd pass, we use inputs X and weights W to compute output Y
• On the bwd pass, we use dY and weights W to compute dX; we also use dX and X to compute dW
Graphcore

- Targets graph-connected workloads, DNNs being the prime example
- Primary philosophy is to reduce data movement and dark silicon
- Already deploying racks of Graphcores with high throughput and no DRAM
- Core design principles:
  - A memory-centric die that achieves high efficiency by keeping memory local
  - Communication and compute are provisioned for peak power and the two are serialized for highest efficiency
  - Re-compute data instead of storing it (no DRAM!)
Graphcore

- With 200 W budget and an 800 mm$^2$ chip, only 1/3 of the chip can run ALUs at 1.5GHz
- DDR4: 320pJ/B, 256GB @ 64GB/s costs 20W
- HBM2 on interposer: 64pJ/B, 16GB @ 900GB/s costs 60W
- Monolithic SRAM on chip: 256MB: 10pJ/B, 6TB/s @ 60W
- Distributed SRAM: 1000 256KB banks is 1pJ/B, 60 TB/s @ 60W
- This SRAM has a power density that is 25% of logic power.

Image Source: Graphcore, NIPS’17
Graphcore vs. GPU

**DRAM on interposer**
180W GPU + 60W HBM2

- 16GB @ 64pJ/B
- 900GB/s

**Distributed SRAM on chip**
2x IPU (75W logic + 45W ram)

- 600MB @ 1pJ/B
- 90,000GB/s

Image Source: Graphcore, NIPS'17
Graphcore

- 4 Colossus chips in one 1U IPU-Machine (see pic)
- 16nm chip, 1000 independent processors per chip
- No attached DRAM
- Mixed-precision fp stochastic arithmetic (16b mult, 32b accum)
- Only about 4 TOPs per chip, but exceeds TPU2 and Volta without large batching
- A rack with 32 1U machines and 4 chips = 500 TOPS
NVIDIA Volta GPU

- 640 tensor cores
- Each tensor core performs a MAC on 4x4 tensors
- Throughput: 128 FLOPs x 640 x 1.5 GHz = 125 Tflops
- FP16 multiply operations
- 12x better than Pascal on training and 6x better on inference
- Basic matrix multiply unit – 32 inputs being fed to 64 parallel multipliers; 64 parallel add operations

\[
D = \begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\
A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\
A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\
A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3}
\end{pmatrix}_{\text{FP16 or FP32}} + \begin{pmatrix}
B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\
B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\
B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\
B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3}
\end{pmatrix}_{\text{FP16}} + \begin{pmatrix}
C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\
C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\
C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\
C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3}
\end{pmatrix}_{\text{FP16 or FP32}}
\]

Intel NNP-T

Spring Crest (NNP-T) SoC

Image Source: Intel presentation HotChips’19
Intel NNP-T

- Each Tensor Processing Cluster (TPC) has two 32x32 MAC grids supporting bfloat16 (appears similar to TPU)
- Conv engine to marshall data before each step
- 60MB on-chip SRAM (2.5MB scratchpad per TPC)
- Communication is key: grid network among TPCs, 64 SerDes lanes (3.6 Tb/s) for inter-chip communication, 4 HBM
- Relatively low utilization for GeMM (< 60%) and Conv (59-87%)
Intel NNP-I

- 12 Inference Compute Engines (ICE) that can work together or independently; 24MB central cache and 4MB per ICE
- Each ICE has 4K 8b MAC unit
- 10W, 48 TOPs, 3600 inf/s

Image Source: Intel presentation HotChips’19
References

• “Neural Networks and Deep Learning,” (Chapter 2) Michael Nielsen

• Graphcore: https://www.graphcore.ai/posts/introducing-the-graphcore-rackscale-ipu-pod

• VOLTA: http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

• Intel NNP: https://www.intel.ai/nervana-nnp/#gs.489oye