Mixed Precision Training

Sharan Narang, Gregory Diamos, Erich Elsen, Paulius Micikevicius, Jonah Alben, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, Hao Wu
Background and Motivation

- Training with reduced precision
  - Reduces memory bandwidth pressure
  - Faster arithmetic
  - Reduces memory required for training
- But FP16 has a narrower dynamic range than FP32
  - May cause underflow/overflow and other arithmetic issues
VV Fast Refresher on IEEE FP numbers

- Representation FP16/32
- Denormalized numbers
  - The zero exponent is reserved for denormalized numbers
- FP Addition
  - Loss of precision while adding
  - For FP16, if operand exponents differ by more than 10 we lose all mantissa bits
Idea 1: FP32 Master Copy Of Weights

• If model weights and gradients are in FP16, weight gradients may underflow
• Also, the ratio of weight value and weight update might be very large
  • Loss of precision while adding
Idea 1: FP32 Master Copy Of Weights
Idea 1: FP32 Master Copy Of Weights

- Impact on
  - Performance
    - Using a FP32 master copy fixes training
  - Memory
    - Keeping a master copy of the weights requires more memory
    - For the models they tested the activation memory is the major bottleneck
    - May not be true if using techniques like activation checkpointing
    - May not be true for LLM training.

(a) Training and validation (dev0) curves for Mandarin speech recognition model
Idea 2: Loss Scaling

- Histogram of activation gradient values
- If cast to FP16, most gradient values will become 0!
- Scaling gradients during backpropagation prevents underflow
- The gradient is scaled before backpropagation begins and rescaled before updating weights
Idea 3: Arithmetic Precision

- Neural Net math
  - Vector dot-products
  - Reductions (BatchNorm, Softmax)
  - Point-wise operations (Non-linearities)
- Accumulating FP16 math into an FP16 value doesn’t work
- The paper proposes accumulating outputs in FP32 and saving them in FP16 format
Results

• Configuration
  • Baseline: Weights, activations, gradients, and arithmetic in FP32
  • Mixed Precision Training (MPT)

• Tasks
  • Vision: Classification, Detection
  • Language: Machine Translation, Language modeling
  • Speech recognition
  • Generative Modeling
Table 1: ILSVRC12 classification top-1 accuracy.

<table>
<thead>
<tr>
<th>Model</th>
<th>Baseline</th>
<th>Mixed Precision</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>56.77%</td>
<td>56.93%</td>
<td>(Krizhevsky et al., 2012)</td>
</tr>
<tr>
<td>VGG-D</td>
<td>65.40%</td>
<td>65.43%</td>
<td>(Simonyan and Zisserman, 2014)</td>
</tr>
<tr>
<td>GoogLeNet (Inception v1)</td>
<td>68.33%</td>
<td>68.43%</td>
<td>(Szegedy et al., 2015)</td>
</tr>
<tr>
<td>Inception v2</td>
<td>70.03%</td>
<td>70.02%</td>
<td>(Ioffe and Szegedy, 2015)</td>
</tr>
<tr>
<td>Inception v3</td>
<td>73.85%</td>
<td>74.13%</td>
<td>(Szegedy et al., 2016)</td>
</tr>
<tr>
<td>Resnet50</td>
<td>75.92%</td>
<td>76.04%</td>
<td>(He et al., 2016b)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Baseline</th>
<th>MP without loss-scale</th>
<th>MP with loss-scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN</td>
<td>69.1%</td>
<td>68.6%</td>
<td>69.7%</td>
</tr>
<tr>
<td>Multibox SSD</td>
<td>76.9%</td>
<td>diverges</td>
<td>77.1%</td>
</tr>
</tbody>
</table>
Figure 4: English to French translation network training perplexity, 3x1024 LSTM model with attention. Ref1, ref2, and ref3 represent three different FP32 training runs.
Table 3: Character Error Rate (CER) using mixed precision training for speech recognition. English results are reported on the WSJ '92 test set. Mandarin results are reported on our internal test set.

<table>
<thead>
<tr>
<th>Model/Dataset</th>
<th>Baseline</th>
<th>Mixed Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>2.20</td>
<td>1.99</td>
</tr>
<tr>
<td>Mandarin</td>
<td>15.82</td>
<td>15.01</td>
</tr>
</tbody>
</table>
Closing Comments

• This paper is from 2018.
• For people working in ML ...

Recent work on MPT

- Automatic mixed precision package: torch.amp
  - Automatic casting to FP16/bfloat16
  - Loss scaling
  - Using underlying tensor-core units
- MPT for LLMs
  - FP8 parameter training
  - Adaptive loss scaling to prevent overflow/underflows
  - Lowering precision of some optimizer states \[ \frac{4}{\text{master weights}} + \frac{4}{\text{gradients}} + \frac{4 + 4}{\text{Adam states}} = 16 \text{ bytes}. \]