Large Batch Optimization for Deep Learning: Training BERT in 76 minutes
(You et al.)

CSE 5449
Autumn ‘21

Presented by Nawras Alnaasan
alnaasan.1@osu.edu
Outline

• Problem Statement
• Background
  • Optimizers in DL
  • BERT Model
• LARS and LAMB Optimizers
• BERT Training
• Summary

https://muppet.fandom.com/wiki/Bert
Problem Statement (1)

Training deep learning models is computationally expensive especially with large scale datasets and models.

Wikipedia+BooksCorpus (2.5B+800M words)
BERT takes 3 days on 16 TPUv3
(Devlin et al., 2018)

ImageNet (14.2M images)
ResNet-50 takes 29 hours on 8 Tesla P100
(He et al., 2016)

How can we speed up the training process?
Problem Statement (1)

Training deep learning models is computationally expensive especially with large scale datasets and models.

<table>
<thead>
<tr>
<th>Wikipedia+BooksCorpus (2.5B+800M words)</th>
<th>ImageNet (14.2M images)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT takes 3 days on 16 TPUv3</td>
<td>ResNet-50 takes 29 hours on 8 Tesla P100</td>
</tr>
<tr>
<td>(Devlin et al., 2018)</td>
<td>(He et al., 2016)</td>
</tr>
</tbody>
</table>

How can we speed up the training process?

- Communication optimization.
- Use more resources:
  - Data Parallelism (using larger global training batch size)
  Example: simply increase BS and use 1024 TPUs instead of 16 ... is that possible?
Large batch size causes degradation in accuracy.

It is a sharp minimum generalization problem:

- Small BS converges to a flat minimum (good generalization).
- Large BS converges to sharp minimum (bad generalization).
Problem: how can we train a DL model with large batch size without losing accuracy?
Outline

• Problem Statement
• Background
  • Optimizers in DL
    • BERT Model
  • LARS and LAMB Optimizers
  • BERT Training
• Summary

https://muppet.fandom.com/wiki/Bert
Background: Optimizers in DL (1)

When we’re training a DL model, what’s the problem we’re trying to solve?

$$\min_{params} \ ( \ Error(params) \ )$$
When we’re training a DL model, what’s the problem we’re trying to solve?

$$\min_{\text{params}} ( \text{Error(params)} )$$

We are solving a minimization problem:

Find best set of (params) to minimize the error in your model.

One way to solve this problem is by using the Gradient Descent algorithm (optimizer)

https://builtin.com/data-science/gradient-descent
We have different variants of the Gradient Descent algorithm:

- Stochastic Gradient Descent (SGD): updates parameters more frequently.
- Adagrad: updates learning rate for each parameter.
- Adam: combines multiple optimizers techniques (Adagrad+RMSProb).
- Momentum: reduces SGD high variance and softens the convergence.
- …
Background: Optimizers in DL (3)

Outline

• Problem Statement
• Background
  • Optimizers in DL
  • BERT Model
• LARS and LAMB Optimizers
• BERT Training
• Summary

https://muppet.fandom.com/wiki/Bert

Attention please!
Background: BERT Model (1)

Developed by Google, Bidirectional Encoder Representations from Transformers (BERT), is a transformer model designed to be pre-trained from unlabeled text and can be fine-tuned to address more specific tasks like:

- Question answering
- Sentiment analysis
- Text summarization

Two phases of training:
1. Pre-train BERT to have general understanding of language.
2. Fine tune BERT to learn specific task

https://www.codemotion.com/magazine/dev-hub/machine-learning-dev/bert-how-google-changed-nlp-and-how-to-benefit-from-this/
Background: BERT Model (2)

1. **Pre-train BERT:**
   - Masked Language Modeling (MLM): basically, fill in the blanks.
   - Next Sentence Prediction (NSP): take two sentences A and B. Determine if B follows A.

   ![Diagram](image-url)

   It means no worries.
   For the rest of your [MASK1].
   It's our problem-free [MASK2].

   A. Simba is the true king.
   B. He is the son of Mufasa.

   [MASK1] = ?
   [MASK2] = ?

   Yes, B follows A.
Background: BERT Model (2)

1. Pre-train BERT:
   - Masked Language Modeling (MLM): basically, fill in the blanks.
   - Next Sentence Prediction (NSP): take two sentences A and B. Determine if B follows A.

   It means no worries.
   For the rest of your [MASK1].
   It's our problem-free [MASK2].

   A. Simba is the true king.
   B. He is the son of Mufasa.

   [MASK1] = days
   [MASK2] = philosophy

   Yes, B follows A.

   Achieves general understanding of language.
2. Fine tune BERT:

For question answering, use a labeled question/answer dataset (supervised training).

Modify input and output of the model such that:

- Input is a concatenation of a question and a passage that contains the answer.
- Output would be a sequence of words that represent the answer.

Fine tuning is fast..

Modified BERT model called “Stanford Question & Answer Dataset” (SQuAD) takes around only 30 minutes to train.
Background: BERT Model (4)

(Devlin et al.) https://arxiv.org/abs/1810.04805
Background: BERT Model (5)

Adopted from (Devlin et al.) https://arxiv.org/abs/1810.04805
Outline

• Problem Statement
• Background
  • Optimizers in DL
  • BERT Model
• LARS and LAMB Optimizers
• BERT Training
• Summary

https://muppet.fandom.com/wiki/Bert
Problem Statement

**Problem:** how can we train a DL model with large batch size without losing accuracy?
General Strategy

Proposed solution by You et al. is to adapt learning rate to large batch size as follows:

Suppose we update weights iteratively (using SGD or ADAM):

\[ w_{t+1} = w_t + \eta_t u_t \]

Where \( u_t \) is the update made by the optimizer at step \( t \).

Now, for large batch settings, we do the following:

1. For each layer in the model, normalize the update \( u \) to unit \( l_2 \)-norm.
2. For each layer, scale learning rate by \( \phi(\|w_t^{(i)}\|) \)

Resulting equation:

\[ w_{t+1}^{(i)} = w_t^{(i)} - \eta_t \frac{\phi(\|w_t^{(i)}\|)}{\|g_t^{(i)}\|} g_t^{(i)} \]

Where \( (i) \) is the layer number and \( (g) \) is the gradients.
**LARS and LAMB Optimizers (1)**

LARS: Layer-wise Adaptive Rate Scaling by (You et al, 2017) uses the momentum optimizer as a base for large batch size training for the ResNet model.

**LARS algorithm**, for each layer $l$ and each step (iteration) $t$:

- $g_t = \frac{1}{B} \sum_{i=1}^{B} \nabla f (x_t^{(i)}, w_{t-1}^{(l)})$ // Compute gradients
- $r_t^l = 1.0$ // Initial trust ratio
- $r_1 = \phi \left( \| w_{t-1}^{(l)} \| \right)$ // Compute norm of the gradients
- $r_2 = \| g_t^{(l)} \| + \lambda \| w_t^{(l)} \|$ // layer-wise weight decay
- If $r_2 > 0$, then $r_t^l = r1/r2$ // Compute the new trust ratio
- $m_t^{(l)} = \beta_1 m_{t-1}^{(l)} + \eta \times r_t^l \times \left( g_t^{(l)} + \lambda w_{t-1}^{(l)} \right)$ // Update the momentum
- $w_t^{(l)} = w_{t-1}^{(l)} + m_t^{(l)}$ // Update the weights
LARS and LAMB Optimizers (2)

LAMB: Layer-wise Adaptive Moments optimizer for Batch training (You et al, 2020) uses the ADAM optimizer as a base for large batch size training for state-of-the-art models.

**LAMB algorithm**, for each layer \( l \) and each step (iteration) \( t \):

- \( g_t = \frac{1}{B} \sum_{i=1}^{B} \nabla f(x^{(i)}_t, w^{(l)}_{t-1}) \)  
  // Compute gradients
- \( m^{(l)}_t = \beta_1 m^{(l)}_{t-1} + (1 - \beta_1) g^{(l)}_t \)  
  // Compute the first moment
- \( v^{(l)}_t = \beta_2 v^{(l)}_{t-1} + (1 - \beta_2) g^{(l)}_{t} \odot g^{(l)}_{t} \)  
  // Compute the second moment
- \( \hat{m}^{(l)}_t = m^{(l)}_t / (1 - \beta_{1}^t) \)  
  // Bias correction for first moment
- \( \hat{v}^{(l)}_t = v^{(l)}_t / (1 - \beta_{2}^t) \)  
  // Bias correction for second moment
- \( r^l_t = 1.0 \)  
  // Initial trust ratio
- \( r_1 = \phi \left( \| w^{(l)}_{t-1} \| \right) \)  
  // Compute norm of the gradients
- \( r_2 = \left\| \frac{\hat{m}^{(l)}_t}{\sqrt{\hat{v}^{(l)}_t + \epsilon}} + \lambda w^{(l)}_t \right\| \)  
  // Element-wise weight decay
- If \( r_2 > 0 \), then \( r^l_t = r_1 / r_2 \)  
  // Compute the new trust ratio
- \( w^{(l)}_t = w^{(l)}_{t-1} - \eta \times r^l_t \times \left( \frac{\hat{m}^{(l)}_t}{\sqrt{\hat{v}^{(l)}_t + \epsilon}} + \lambda w^{(l)}_t \right) \)  
  // Update the weights
LARS and LAMB Optimizers (3)

How can LARS/LAMB solve the sharp minimum problem?

- With small batch size, noisy gradients are used in the step computation.
  - Noise is important because it can push the solution out of the sharp minimum attraction.

- However, noise is not sufficient in large batch sizes to do the same.

- Researchers tried adding artificial noise (Gaussian noise) to different parameters like activations, weights, gradients, outputs, etc., but that didn’t help.

- With large batch size, the dynamics of LARS/LAMB act as that needed noise.
LARS and LAMB Optimizers (4)

How can LARS/LAMB solve the sharp minimum problem?

Outline

- Problem Statement
- Background
  - Optimizers in DL
  - BERT Model
- LARS and LAMB Optimizers
- BERT Training
- Summary

https://muppet.fandom.com/wiki/Bert
BERT Training with LAMB (1)

Experiment

Dataset: Wikipedia+BooksCorpus (2.5B+800M words)
Model: BERT with the SQuAD task (question answering)
Metric: F1 score on SQuAD-v1

<table>
<thead>
<tr>
<th>Solver</th>
<th>batch size</th>
<th>steps</th>
<th>F1 score on dev set</th>
<th>TPs</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>512</td>
<td>1000k</td>
<td>90.395</td>
<td>16</td>
<td>81.4h</td>
</tr>
<tr>
<td>LAMB</td>
<td>512</td>
<td>1000k</td>
<td>91.752</td>
<td>16</td>
<td>82.8h</td>
</tr>
<tr>
<td>LAMB</td>
<td>1k</td>
<td>500k</td>
<td>91.761</td>
<td>32</td>
<td>43.2h</td>
</tr>
<tr>
<td>LAMB</td>
<td>2k</td>
<td>250k</td>
<td>91.946</td>
<td>64</td>
<td>21.4h</td>
</tr>
<tr>
<td>LAMB</td>
<td>4k</td>
<td>125k</td>
<td>91.137</td>
<td>128</td>
<td>693.6m</td>
</tr>
<tr>
<td>LAMB</td>
<td>8k</td>
<td>62500</td>
<td>91.263</td>
<td>256</td>
<td>390.5m</td>
</tr>
<tr>
<td>LAMB</td>
<td>16k</td>
<td>31250</td>
<td>91.345</td>
<td>512</td>
<td>200.0m</td>
</tr>
<tr>
<td>LAMB</td>
<td>32k</td>
<td>15625</td>
<td>91.475</td>
<td>1024</td>
<td>101.2m</td>
</tr>
<tr>
<td>LAMB</td>
<td>64k/32k</td>
<td>8599</td>
<td>90.584</td>
<td>1024</td>
<td>76.19m</td>
</tr>
</tbody>
</table>

BERT pre-training has two stages:

1. 9/10 epochs use sequence length (number of tokens) of 128.
2. 1/10 epochs use sequence length of 512.

You et. Al used batch size 64K for the first stage but 32K for the second stage.

**Question:** Why did they reduce the batch size to 32K in the second stage?
Comparison of LAMB with ADAMW and LARS:

Table 8: ADAMW stops scaling at the batch size of 16K. The target F1 score is 90.5. LAMB achieves a F1 score of 91.345. The table shows the tuning information of ADAMW. In this table, we report the best F1 score we observed from our experiments.

<table>
<thead>
<tr>
<th>Solver</th>
<th>batch size</th>
<th>warmup steps</th>
<th>LR</th>
<th>last step information</th>
<th>F1 score on dev set</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.05×31250</td>
<td>0.0001</td>
<td>loss=8.04471, step=28126</td>
<td>diverged</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.05×31250</td>
<td>0.0002</td>
<td>loss=7.89673, step=28126</td>
<td>diverged</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.05×31250</td>
<td>0.0003</td>
<td>loss=8.35102, step=28126</td>
<td>diverged</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.10×31250</td>
<td>0.0001</td>
<td>loss=2.01419, step=31250</td>
<td>86.034</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.10×31250</td>
<td>0.0002</td>
<td>loss=1.04689, step=31250</td>
<td>88.540</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.10×31250</td>
<td>0.0003</td>
<td>loss=8.05845, step=20000</td>
<td>diverged</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.20×31250</td>
<td>0.0001</td>
<td>loss=1.53706, step=31250</td>
<td>85.231</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.20×31250</td>
<td>0.0002</td>
<td>loss=1.15500, step=31250</td>
<td>88.110</td>
</tr>
<tr>
<td>ADAMW</td>
<td>16K</td>
<td>0.20×31250</td>
<td>0.0003</td>
<td>loss=1.48798, step=31250</td>
<td>85.653</td>
</tr>
</tbody>
</table>

Table 2: LAMB achieves a higher performance (F1 score) than LARS for all the batch sizes. The baseline achieves a F1 score of 90.390. Thus, LARS stops scaling at the batch size of 16K.

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>512</th>
<th>1K</th>
<th>2K</th>
<th>4K</th>
<th>8K</th>
<th>16K</th>
<th>32K</th>
</tr>
</thead>
<tbody>
<tr>
<td>LARS</td>
<td>90.717</td>
<td>90.369</td>
<td>90.748</td>
<td>90.537</td>
<td>90.548</td>
<td>89.589</td>
<td>diverge</td>
</tr>
<tr>
<td>LAMB</td>
<td>91.752</td>
<td>91.761</td>
<td>91.946</td>
<td>91.137</td>
<td>91.263</td>
<td>91.345</td>
<td>91.475</td>
</tr>
</tbody>
</table>

Outline

- Problem Statement
- Background
  - Optimizers in DL
  - BERT Model
  - LARS and LAMB Optimizers
  - BERT Training
- Summary

Finally...

https://muppet.fandom.com/wiki/Bert
Summary

- Large batch technique is critical for speeding up DNN training.
- We might lose accuracy if we use large batch sizes.
- LAMB optimizer is a solution that works for both small and large batch sizes.
- LAMB can scale BS of BERT up to 64K without losing accuracy.
- Larger batch size allows us to use more resources with data parallelism.
- With 1024 TPUs, BERT training was reduced from 3 days to 76 minutes using LAMB.
Questions?

https://muppet.fandom.com/wiki/Bert