Resource Allocation for Mini-batch SGD via Adaptive Batch Sizes

Scott Sievert

eLUCID8, 2019-08-06, UW–Madison

Agenda

- Motivation
- Results
- Practical implementation, with experiments

Joint work with
Zachary Charles

What do these images have common?

Use a popular image classification model,

tensorflow / **tpu** / models / official / **resnet** /

Star 2,215

... the model should train to above 76% accuracy **in around 17 hours** [on a Google TPU]...

https://github.com/tensorflow/tpu

ImageNet dataset: http://www.image-net.org/

Choice of Plausible Alternatives (COPA)
An evaluation of commonsense causal reasoning

Q: *The man broke his toe. What was the CAUSE of this?*
(a) He got a hole in his sock.
(b) He dropped a hammer on his foot.

Use a popular natural language processing model,

openai / **gpt-2**

★ Star 7,156

Drawbacks

... Our approach requires an expensive *pre-training* step – **1 month on 8 GPUs.** ...

The total compute used to train this model was **0.96 petaflop days** (pfs-days).

More than 2 days for *one training* with 4 high end GPUs!

https://openai.com/blog/language-unsupervised/

COPA dataset: http://people.ict.usc.edu/~gordon/copa.html

Data

Results

Developer

Debug loop

Supercomputer

Deep learning is required to get these models.

Deep learning requires “big data”

During training, dataset is approximated with few data

Context: deep learning

Specifically, approximate with
batch size examples

In deep learning, typically solved with updates of the form

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\gamma}{B} \sum_{j=1}^{B} g_{i_j}$$

...with a *static* batch size *B*

Questions?

How it batch size specified?

batch_size (*int, optional*) – how many samples per batch to load (default: `1`).

```
13   from keras.layers import Conv2D, MaxPooling2D
14   from keras import backend as K
15
16   batch_size = 128
...             ...
63   model.fit(x_train, y_train,
64             batch_size=batch_size,
```

Typically small
(32 ≤ batch_size ≤ 1024)

Keras Documentation

Search docs

PyTorch

TensorFlow

Computer Science > Distributed, Parallel, and Cluster Computing arXiv:1711.04325

Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes

Takuya Akiba, Shuji Suzuki, Keisuke Fukuda

(Submitted on 12 Nov 2017)

> We demonstrate that training ResNet-50 on ImageNet for 90 epochs can be achieved **in 15 minutes with 1024 Tesla P100 GPUs**. This was made possible by using **a large minibatch size of 32k.** …

Computer Science > Computer Vision and Pattern Recognition arXiv:1706.02677

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, Kaiming He

(Submitted on 8 Jun 2017 (v1), last revised 30 Apr 2018 (this version, v2))

> … trains ResNet- 50 with **a minibatch size of 8192 on 256 GPUs in one hour**, …

It’d be nice to train quickly
without having to buy 100’s of GPUs

Claim: the batch size should grow as optimization proceeds

Result: quicker training

Intuition for batch size growth

Why compute gradients for multiple examples
if all examples have an optimal model
in the same direction?

Not much gain to re-computing gradients if they're all pretty similar

Batch size growth

Let’s grow the batch size via

$$B_k = \mathcal{O}\left(\frac{1}{\text{loss}_k}\right)$$

Poor initialization: $B_k = 1$
At optimum: $B_k = \infty$

$$B_k = \left\lceil \frac{c}{F(\boldsymbol{w_k}) - F^\star} \right\rceil$$

Main result: summary

Feature of SGD: no dependence on number of examples in training set

Feature of gradient descent (GD): few model updates

When the batch size is grown as shown, then the same number of...

- model updates as GD are required
- examples as SGD are required

I have formal theorems backing this statement.

Main result

To achieve a model x with training loss $F(x_k) - F^\star \in [\epsilon/2, \epsilon]$,

Function class	SGD	Adaptive batch sizes	Gradient descent
Smooth and convex	$\mathcal{O}\left(1/\epsilon^2\right)$	$\mathcal{O}\left(1/\epsilon\right)$	$\mathcal{O}\left(1/\epsilon\right)$
Strongly convex*	$\mathcal{O}\left(1/\epsilon\right)$	$\mathcal{O}\left(\log(1/\epsilon)\right)$	$\mathcal{O}\left(\log(1/\epsilon)\right)$

model updates need to be completed and

Function class	SGD	Adaptive batch sizes	Gradient descent
Smooth and convex	$\mathcal{O}\left(1/\epsilon^2\right)$	$\mathcal{O}\left(1/\epsilon^2\right)$	$\mathcal{O}\left(n/\epsilon\right)$
Strongly convex*	$\mathcal{O}\left(1/\epsilon\right)$	$\mathcal{O}\left(\log(1/\epsilon)/\epsilon\right)$	$\mathcal{O}\left(n\log(1/\epsilon)\right)$

examples need to be processed

Similar results for reaching a saddle point with non-convex function

*actually a generalization of strongly convex [3]

1. Bubeck et al. *Convex optimization: Algorithms and complexity*. 2015.
2. Hamed Karimi et al. *Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition*. 2016.
3. Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. 2013.

These theorems are confirmed with simulations.

Practical effect

- More amenable to distributed computing
- Quicker time to solution if resources allocated with batch size

Limitations

1. Using the *entire* train dataset *every* model update takes a long time
2. How does the model perform on unseen data?
3. GPU memory is finite

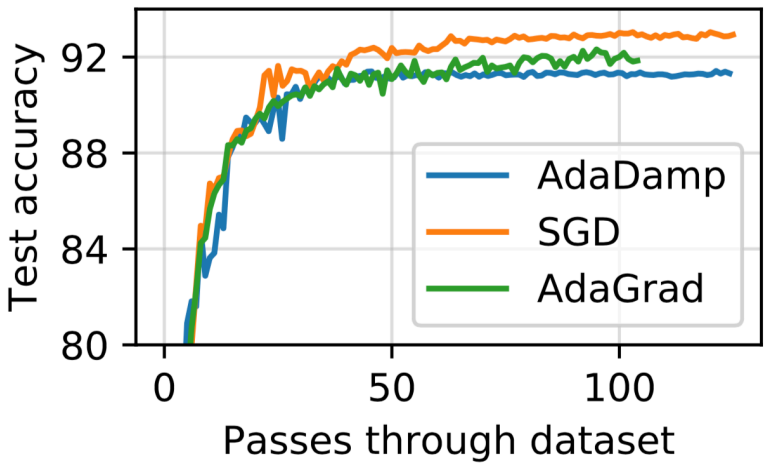
Algorithm 1 Mini-batch SGD with dampening of noise in gradient approximation

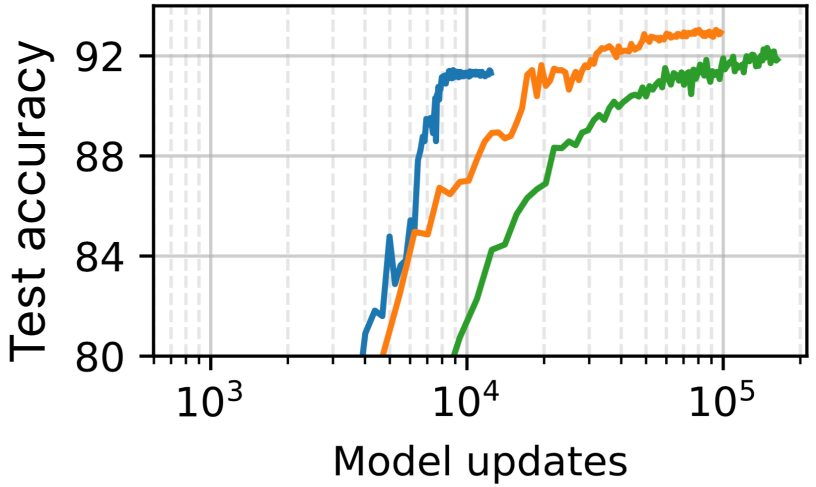
procedure ADADAMP(initial batch size B_0, initial model x_0, step size γ, maximum batch size $B_{\max}$)
  for $k \in [0, 1, 2, \ldots]$ **do**
    if $k = 0$ **then**
      $c \leftarrow B_0(F(x_0) - F^\star)$
    $B_k \leftarrow \lceil c/(F(x_k) - F^\star) \rceil$
    if $B_k > B_{\max}$ **then**
      $\gamma' \leftarrow \gamma B_{\max}/B_k$
      $B_k = B_{\max}$
    $x_{k+1} \leftarrow \text{TRAIN}(x_k, \gamma', B_k)$ ▷ TRAIN computes n gradients
  return x_k

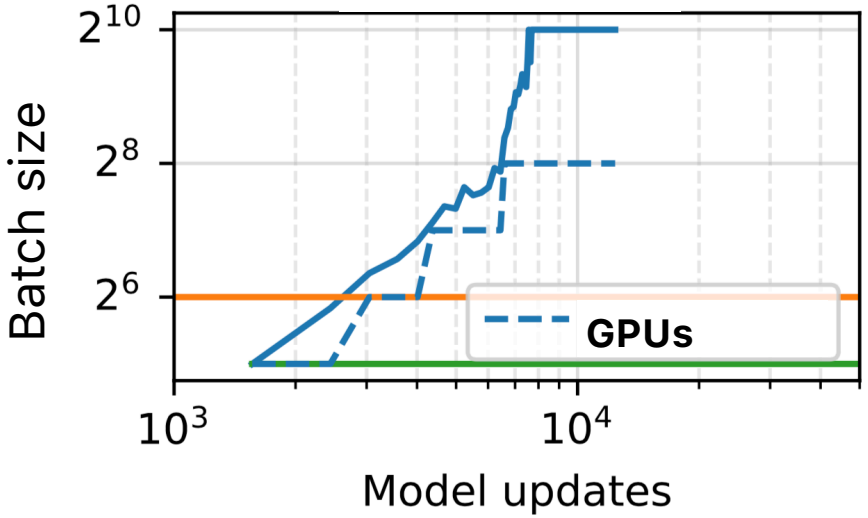
Smith et. al. *A bayesian perspective on generalization and stochastic gradient decent.* 2017

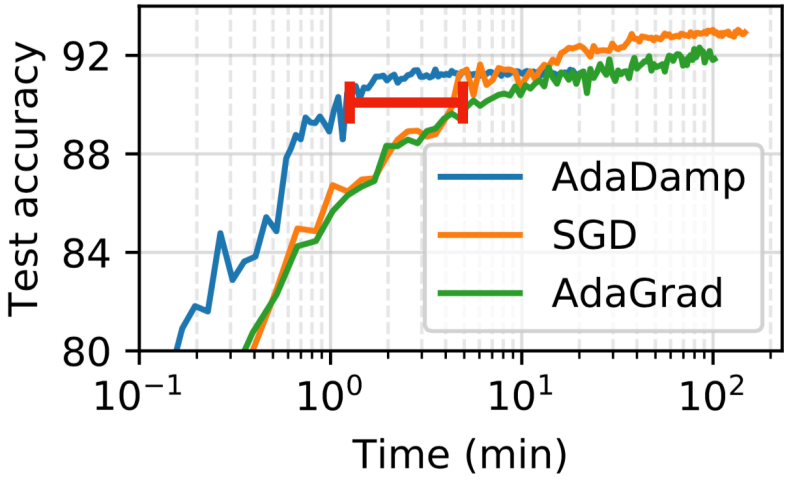
Experiment

ResNet-34 + CIFAR-10
Brief hyperparameter tuning









All plots assume oracle provides $F(\boldsymbol{x}_k) - F^\star$

How can the batch size be estimated?

Our upper bounds suggest $B_{\text{epoch}} \propto r \cdot \text{epoch}$

Similar method uses $B_{\text{epoch}} \propto r^{\text{epoch}}$

Smith et. al. *Don't Decay the Learning Rate, Increase the Batch Size*. 2017. https://arxiv.org/abs/1711.00489

Thanks!

Questions?

Future work

after this work is finished

How does this method generalize?

Bounds

Convex and smooth	$F(\boldsymbol{x}_k) - F^\star \leq \mathcal{O}\left(r^k\right)$
α -PL (generalization of strongly convex)	$F(\boldsymbol{x}_k) - F^\star \leq \mathcal{O}\left(\frac{1}{k}\right)$
Non-convex	$\min_{k=0,\ldots,T-1} \|\nabla F(\boldsymbol{x}_k)\|_2^2 \leq \mathcal{O}\left(\frac{1}{k}\right)$

Max batch size?

Computer Science > Machine Learning

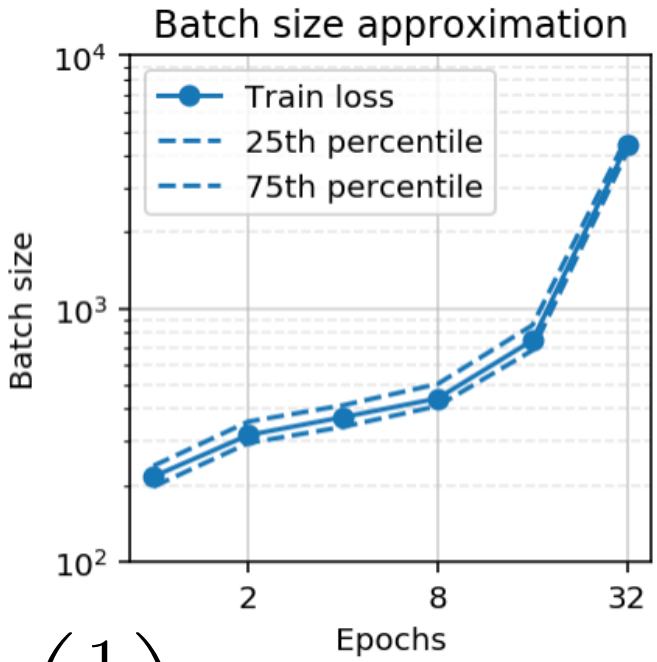
A Bayesian Perspective on Generalization and Stochastic Gradient Descent

Samuel L. Smith, Quoc V. Le

(Submitted on 17 Oct 2017 (v1), last revised 14 Feb 2018 (this version, v3))

> … We also demonstrate that, when one holds the learning rate fixed, **there is an optimum batch size which maximizes the test set accuracy.** …

Entire train dataset?



$$F(\boldsymbol{x}_k) - F^\star \leq \mathcal{O}\left(\frac{1}{k}\right)$$

*For convex and non-convex functions. For convex, relies on

Rule of thumb can likely be developed