Benchmarking Neural Network Training Algorithms

ELLIS Workshop

Frank Schneider

July 19, 2023

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

MAX PLANCK INSTITUTE FOR INTELLIGENT SYSTEMS

erc

Federal Ministry of Education and Research

imprs-is

Neural Networks are extremely useful models...

...but!

In practice, neural networks are...

- ...**slow** to train (training can easily take days or weeks)
- ...**tedious** (demanding human intervention and fiddeling)
- ...**expensive** (requiring dozens of trial runs)

Neural Networks are extremely useful models...

...but!

In practice, neural networks are...

- ...**slow** to train (training can easily take days or weeks)
- ...**tedious** (demanding human intervention and fiddeling)
- ...**expensive** (requiring dozens of trial runs)

We all want neural network training to be faster, more automatic, and more efficient!

The Three Pillars of Neural Network Training

Our focus is on the (training) algorithms

Hardware

- Leveraging accelerators (e.g. GPUs, TPUs, etc.)
- Maximizing accelerator utilization throughout training
- Reducing hardware bottlenecks

Software

- Convenient deep learning frameworks (e.g. PyTorch, JAX, etc.)
- Efficient software implementations
- Putting ML models into production

Algorithms

- Efficient **training algorithms** (e.g. Adam, Shampoo, etc.)
- Powerful deep learning models (e.g. transformers, etc.)
- Faster tuning methods (e.g. BayesOpt, etc.)

The State of Deep Learning Training Methods

A confusingly crowded field of methods & hyperparameters

A huge number of training methods...

from "Descending through a Crowded Valley"
(Schmidt, Schneider, Hennig; 2021)

The State of Deep Learning Training Methods

A confusingly crowded field of methods & hyperparameters

A huge number of training methods...

from "Descending through a Crowded Valley" (Schmidt, Schneider, Hennig; 2021)

...and training tricks

- OneCycle scheduler, gradient checkpointing
- Genetic Algorithm for Hyperparameters
- Avoid batches that lead to NaN/inf losses
- One cycle, low fidelity training, SGD with restarts
- Proximal optimization for regularizers
- Line searches for the maximum learning rate
- Normalized updates
- Distributed Shampoo, Normformer, GLU
- Weight averaging
- FreezeOut
- A different epsilon value!
- Check hyperparameter performance over multiple seeds
- Lowering the learning rate!
- Normalizing data works better than batch or layer norm
- Mixed precision training
- Train with a small subset
- Cyclic and one cycle LR
- Label smoothing
- ...

from the NeurIPS "HITY Workshop" (Schneider et al.; 2022)

The Current Benchmarks for Training Algorithms are Insufficient

and this holds back the entire field

- **Unclear SOTA**
  There is no established protocol to train neural networks.
- **No reliable way to detect algorithmic improvements**
  Let alone understand what novel methods are most promising.
- **Training algorithms are assumed to not be useful until widely adopted**
  This chicken-and-egg problem is troubling for practitioners training neural networks and developers of new training algorithms.

The Current Benchmarks for Training Algorithms are Insufficient

and this holds back the entire field

- **Unclear SOTA**
  There is no established protocol to train neural networks.
- **No reliable way to detect algorithmic improvements**
  Let alone understand what novel methods are most promising.
- **Training algorithms are assumed to not be useful until widely adopted**
  This chicken-and-egg problem is troubling for practitioners training neural networks and developers of new training algorithms.

We desperately need new benchmarks for neural network training algorithms!

MLCommons Introduces the AlgoPef: Training Algorithm Benchmark

An unprecedented effort to find faster deep learning training algorithms

ML Commons

Algorithms Working Group

George Dahl Google

Frank Schneider University of Tübingen

AlgoPerf: Training Algorithm Benchmark

A competition to measure neural network training speedups due to algorithmic changes.

- A competitive benchmark with open submissions
- Compete on time-to-result over multiple deep learning workloads
- A huge large-scale effort by 25+ researchers from Google, University of Tübingen, University of Toronto, Meta AI, etc.

Challenge I: What is the target?

Which algorithms trains the fastest depends on what it means for training to be complete

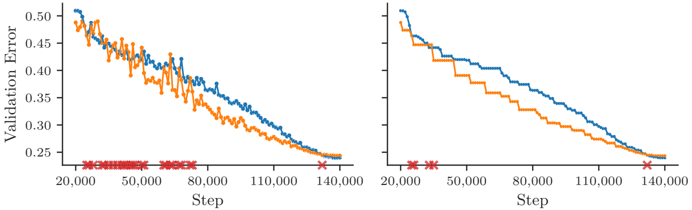

Figure 1: *Left:* Validation error for two different runs (—, —) of Adam on ResNet-50 on ImageNet. *Right:* The *best* validation error obtained so far. The runs intersect multiple times (✖).

Challenge I: What is the target?

Which algorithms trains the fastest depends on what it means for training to be complete

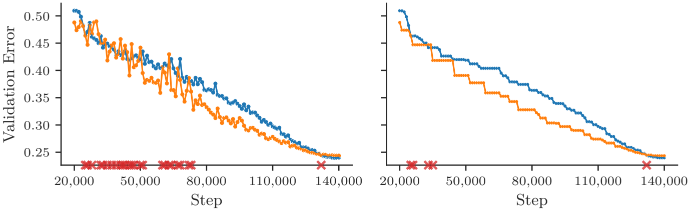

Figure 1: *Left:* Validation error for two different runs (—, —) of ADAM on RESNET-50 on IMAGENET. *Right:* The *best* validation error obtained so far. The runs intersect multiple times (✖).

- Directly comparing training curves is ill-posed
- Without defining the target in advance, we can champion any method
- We are effectively moving the goal post after the experiment

Solution I: Defining the Target and the Ruler

Setting a common goal post

- **Defined Target Performances**
  We define competitive validation and test targets for each workload that can be reliably achieved with currently popular methods.
- **A Time-To-Results Competition**
  Measure the wall-clock runtime until the algorithm first hits the targets.
- **Fixed Hardware**
  Submissions need to innovate on the training algorithm.

Challenge II: Dependence on the Workload

Seemingly minor changes to the model can have a large effect on the performance of different training algorithms

Training Algorithm	**ResNet-200** *(standard)*
Nesterov	**0.2090**
AdamW	0.2626

Table 1: Performance of training methods on different workloads. Shown is the best error rate (lower is better) achieved.

Challenge II: Dependence on the Workload

Seemingly minor changes to the model can have a large effect on the performance of different training algorithms

Training Algorithm	ResNet-200 *(standard)*	ResNet-200 *Extra-BN*
Nesterov	**0.2090**	No Feasible Trials
AdamW	0.2626	0.2722

Table 1: Performance of training methods on different workloads. Shown is the best error rate (lower is better) achieved.

Challenge II: Dependence on the Workload

Seemingly minor changes to the model can have a large effect on the performance of different training algorithms

Training Algorithm	**ResNet-200** *(standard)*	**ResNet-200** *Extra-BN*
Nesterov	**0.2090**	No Feasible Trials
AdamW	0.2626	0.2722

Table 1: Performance of training methods on different workloads. Shown is the best error rate (lower is better) achieved.

- How general is a performance improvement?
- Comparison with published results is dangerous
- Important to choose relevant workloads

Solution II: Evaluate on Multiple Fixed Workloads

Finding general-purpose algorithmic improvements

- **Fixed Workloads**
  We don't allow pipeline changes that are not part of the training algorithm.
- **Aggregate across Multiple Workloads**
  Instead of specialized solutions, we want to find the best general-purpose method.
- **Use Randomized Workload Variants**
  To test the robustness of the methods.

Solution II: Evaluate on Multiple Fixed Workloads

Finding general-purpose algorithmic improvements

Task	Dataset	Model	Loss	Metric	Validation **Target**	Test **Target**	Maximum **Runtime**
Clickthrough rate prediction	**CRITEO 1TB**	**DLRMSMALL**	CE	CE	0.123 649	0.126 060	7703
MRI reconstruction	**FASTMRI**	**U-NET**	L1	SSIM	0.7344	0.741 652	8859
Image classification	**IMAGENET**	**RESNET-50** **VIT**	CE CE	ER ER	0.225 69 0.226 91	0.3440 0.3481	63 008 77 520
Speech recognition	**LIBRISPEECH**	**CONFORMER** **DEEPSPEECH**	CTC CTC	WER WER	0.078 477 0.1162	0.046 973 0.068 093	101 780 92 509
Molecular property prediction	**OGBG**	**GNN**	CE	mAP	0.280 98	0.268 729	18 477
Translation	**WMT 2016**	**TRANSFORMER**	CE	BLEU	30.8491	30.7219	48 151

Challenge III: Hyperparameters

Often training algorithms are only templates, not runnable procedures

Search Space	Learning Rate	Weight Decay	$1-\beta_1$	β_2
AdamW Narrow	[2e-4, 5e-3]	[2e-2, 0.5]	0.1	0.999
AdamW Broad	[5e-6, 2e-2]	[5e-6, 2.0]	[1e-3, 1.0]	0.999

Table 2: Hyperparameter search spaces for two algorithms using AdamW.

Challenge III: Hyperparameters

Often training algorithms are only templates, not runnable procedures

Search Space	Learning Rate	Weight Decay	$1-\beta_1$	β_2
AdamW Narrow	[2e-4, 5e-3]	[2e-2, 0.5]	0.1	0.999
AdamW Broad	[5e-6, 2e-2]	[5e-6, 2.0]	[1e-3, 1.0]	0.999

Table 2: Hyperparameter search spaces for two algorithms using AdamW.

Workload		AdamW Narrow	AdamW Broad
Criteo 1TB	DLRMsmall	**0.124 01**	0.124 087
fastMRI	U-Net	**0.734 746**	0.734 311
ImageNet	ResNet-50	**0.232 56**	0.243 34
	ViT	**0.219 92**	0.236 16
LibriSpeech	Conformer	**0.075 989**	0.080 673
	DeepSpeech	**0.112 706**	0.120 674
OGBG	GNN	**0.282 14**	0.276 307
WMT 2016	Transformer	**31.3523**	30.9950

Table 3: Performance across multiple workloads for two AdamW training methods. Bolded number highlights the better performance.

Challenge III: Hyperparameters

Often training algorithms are only templates, not runnable procedures

Search Space	Learning Rate	Weight Decay	$1-\beta_1$	β_2
AdamW Narrow	[2e-4, 5e-3]	[2e-2, 0.5]	0.1	0.999
AdamW Broad	[5e-6, 2e-2]	[5e-6, 2.0]	[1e-3, 1.0]	0.999

Table 2: Hyperparameter search spaces for two algorithms using AdamW.

Workload		AdamW Narrow	AdamW Broad
Criteo 1TB	DLRMsmall	**0.124 01**	0.124 087
fastMRI	U-Net	**0.734 746**	0.734 311
ImageNet	ResNet-50	**0.232 56**	0.243 34
	ViT	**0.219 92**	0.236 16
LibriSpeech	Conformer	**0.075 989**	0.080 673
	DeepSpeech	**0.112 706**	0.120 674
OGBG	GNN	**0.282 14**	0.276 307
WMT 2016	Transformer	**31.3523**	30.9950

Table 3: Performance across multiple workloads for two AdamW training methods. Bolded number highlights the better performance.

- Methods have hyperparameters that need to be set/tuned
- We can't just ignore them as they are workload-dependent
- Same update rule with different hyperparameters is effectively a different algorithm

Solution III: Make Tuning an Explicit Part of the Benchmark

Submitting algorithms not templates

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Hyperparameter (Search Spaces) are Part of the Submission**
  Submitters have to provide search spaces or default values.
- **A Competitive Benchmark**
  Everyone submits their algorithm and how it should be tuned, therefore generating strong baselines.
- **Two Tuning Rulesets**
  - In the **self-tuning** ruleset, everything is done on the clock (e.g. line-searches, freeze-thaw, default hyperparameters, etc.).
  - The **external** tuning ruleset allows parallel resources and only the fastest trial counts.

Summary

- Neural networks are useful but expensive models
- We are currently unable to identify which training algorithms are “best”
- We created the **AlgoPerf: Training Algorithms Benchmark** to find faster deep learning training algorithms
  - A competitive, time-to-result benchmark
  - Running on fixed hardware and workloads
  - Computing a joint score across multiple realistic workloads

Read the Rules	`github.com/mlcommons/algorithmic-efficiency/blob/main/RULES.md`
Read the Paper	`arxiv.org/abs/2306.07179`
Submit!	Call for Submission coming soon