Neural Network Training through the Lens of Benchmarking and Debugging

SPP 2353 - Summer School

Frank Schneider

April 4, 2023

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

MAX PLANCK INSTITUTE FOR INTELLIGENT SYSTEMS

erc

Federal Ministry of Education and Research

imprs-is

Deep Learning

A highly popular field with headline-grabbing applications

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

LLMs
ChatGPT/GPT-4

Generative AI
Stable Diffusion/DALL-E/Midjourney

AI4Science
AlphaFold

SPP 2353

How do we train neural networks?

The Three Pillars of Neural Network Training

Our focus is on the (training) algorithms

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

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 OPT Training Logbook

A real-world case study of how hard it is to train a neural network

(Zhang et al. 2022)

- Meta AI trained an open-source GPT-3-like large language model (LLM) called OPT.

OPT: Open Pre-trained Transformer Language Models

Susan Zhang[*], **Stephen Roller**[*], **Naman Goyal**[*],
Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li,
Xi Victoria Lin, Todor Mihaylov, Myle Ott[†], **Sam Shleifer**[†], **Kurt Shuster, Daniel Simig,**
Punit Singh Koura, Anjali Sridhar, Tianlu Wang, Luke Zettlemoyer

Meta AI

{susanz,roller,naman}@fb.com

The OPT Training Logbook

A real-world case study of how hard it is to train a neural network

(Zhang et al. 2022)

- Meta AI trained an open-source GPT-3-like large language model (LLM) called OPT.
- Training clearly requires more than just "ADAM with a default learning rate".

We use an AdamW optimizer (Loshchilov and Hutter, 2017) with (β_1, β_2) set to $(0.9, 0.95)$, and weight decay of 0.1. We follow a linear learning rate schedule, warming up from 0 to the maximum learning rate over the first 2000 steps in OPT-175B, or over 375M tokens in our smaller baselines, and decaying down to 10% of the maximum LR over 300B tokens. A number of mid-flight changes to LR were also required (see Section 2.5). Our batch sizes range from 0.5M to 4M depending on the model size (see Table 1) and is kept constant throughout the course of training.

We use a dropout of 0.1 throughout, but we do not apply any dropout to embeddings. We clip gradient norms at 1.0, except for some mid-flight changes that reduce this threshold down from 1.0 to 0.3 (see Section 2.5). We also in-

- Meta AI trained an open-source GPT-3-like large language model (LLM) called OPT.
- Training clearly requires more than just "ADAM with a default learning rate".
- The result is a highly empirical and complicated learning rate schedule.

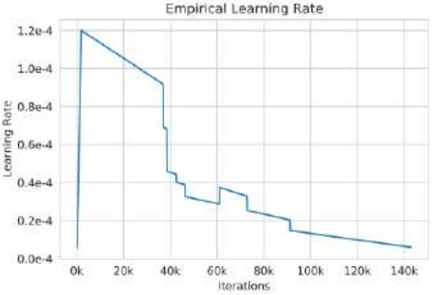


Figure 1: **Empirical LR schedule.** We found that lowering learning rate was helpful for avoiding instabilities.

The OPT Training Logbook

A real-world case study of how hard it is to train a neural network

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Zhang et al. 2022)

- Meta AI trained an open-source GPT-3-like large language model (LLM) called OPT.
- Training clearly requires more than just "ADAM with a default learning rate".
- The result is a highly empirical and complicated learning rate schedule.
- 114-paged logbook detailing the struggles.

Run is stuck in loop of "lower loss scale"

Uh oh. The loss exploded. We don't know why.

Remember to document your actions in the logbook.

Actions:

1. **Don't panic.**
2. Ping the group chat to discuss potential options. If no one responds make a decision by yourself. Letting nodes idle costs $2500/hour so

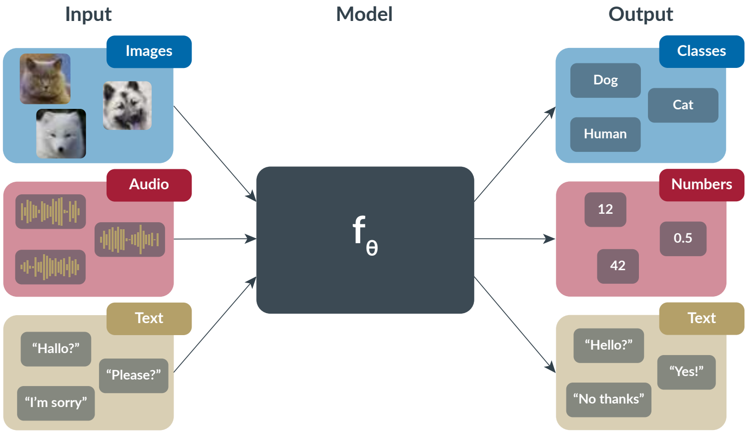
Part I

Fundamentals

Why training a neural network is hard

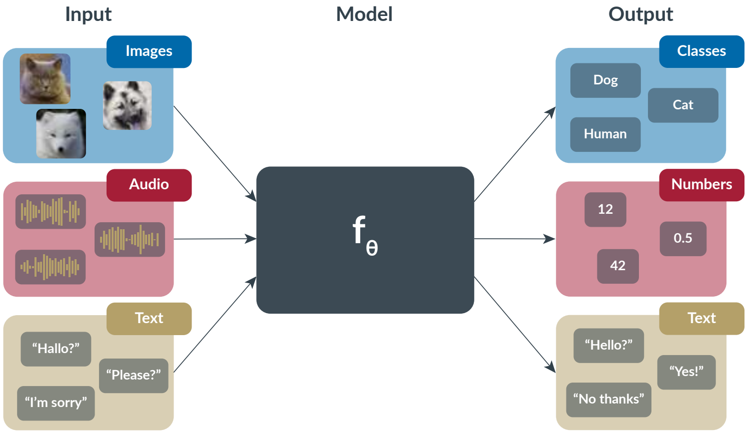
Supervised Learning with Neural Networks

A simplified overview of what sounds like an optimization problem



Supervised Learning with Neural Networks

A simplified overview of what sounds like an optimization problem



1. **Data**
   $\mathbb{D}_{\text{train}} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}$
2. **Model**
   $f_{\boldsymbol{\theta}}$
3. **Loss**
   L
4. **Training Algorithm**
   $\boldsymbol{\theta}_{t+1} = \ldots$

Setting

We want to learn a function which is able to correctly predict the output $y \in \mathbb{Y}$ given some features $\boldsymbol{x} \in \mathbb{X}$. A **loss function** $\ell(\hat{y}, y) : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}^+$ quantifies how different the predictions of a model $\hat{y}$ are from the true targets y.

True Risk/Expected Loss

Given the **true underlying data distribution** $P_{\text{true}}(\boldsymbol{x}, y)$, we want to minimize

$$L_{P_{\text{true}}}(f_{\boldsymbol{\theta}}) = \mathbb{E}_{(\boldsymbol{x},y)\sim P_{\text{true}}(\boldsymbol{x},y)} \left[\ell\left(f(\boldsymbol{x}; \boldsymbol{\theta}), y\right)\right]$$

Empirical Risk/Empirical Loss

However, we only have access to a finite **training set** $\mathbb{D}_{\text{train}} = \left\{(\boldsymbol{x}^{(1)}, y^{(1)}), \dots, (\boldsymbol{x}^{(N)}, y^{(N)})\right\}$ and instead minimize

$$L_{\mathbb{D}_{\text{train}}}(f_{\boldsymbol{\theta}}) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\ell(f(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}), y^{(i)})}_{=:\ell^{(i)}}$$

Stochastic Gradient Descent

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \boldsymbol{g}_{\mathbb{B}^{(t)}}$$

(Heavy Ball) Momentum

$$\begin{aligned} \boldsymbol{v}^{(t)} &= \rho \boldsymbol{v}^{(t-1)} + \eta \boldsymbol{g}_{\mathbb{B}^{(t)}} \\ \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \boldsymbol{v}^{(t)} \end{aligned}$$

Nesterov Accelerated Gradient

$$
\begin{aligned}
\boldsymbol{v}^{(t)} &= \rho \boldsymbol{v}^{(t-1)} + \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_{\boldsymbol{\theta}} \ell(f(\boldsymbol{x}^{(i)}, \boldsymbol{\theta} - \rho \boldsymbol{v}^{(t-1)}), y^{(i)}) \\
\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \boldsymbol{v}^{(t)}
\end{aligned}
$$

RMSProp

$$
\begin{aligned}
\boldsymbol{s}^{(t)} &= \rho \boldsymbol{s}^{(t-1)} + (1-\rho)\boldsymbol{g}_{\mathbb{B}^{(t)}} \odot \boldsymbol{g}_{\mathbb{B}^{(t)}} \\
\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \frac{\eta}{\sqrt{\boldsymbol{s}^{(t)}} + \varepsilon} \odot \boldsymbol{g}_{\mathbb{B}^{(t)}}
\end{aligned}
$$

Adam

$$
\begin{aligned}
\boldsymbol{m}^{(t)} &= \beta_1 \boldsymbol{m}^{(t-1)} + (1-\beta_1)\boldsymbol{g}_{\mathbb{B}^{(t)}} \\
\boldsymbol{v}^{(t)} &= \beta_2 \boldsymbol{v}^{(t-1)} + (1-\beta_2)\boldsymbol{g}_{\mathbb{B}^{(t)}} \odot \boldsymbol{g}_{\mathbb{B}^{(t)}} \\
\hat{\boldsymbol{m}}^{(t)} &= \frac{\boldsymbol{m}^{(t)}}{1-\beta_1^t} \\
\hat{\boldsymbol{v}}^{(t)} &= \frac{\boldsymbol{v}^{(t)}}{1-\beta_2^t} \\
\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}^{(t)}}+\varepsilon} \odot \hat{\boldsymbol{m}}^{(t)}
\end{aligned}
$$

The State of Neural Network Training

A crowded field of methods and hyperparameters

A huge number of optimization methods

AcceleGrad	AMSBound	K-BFGS/K-BFGS(L)	RMSProp
ACClip	AMSGrad	KF-QN-CNN	RMSterov
AdaAlter	AngularGrad	KFAC	S-SGD
AdaBatch	ArmijoLS	KFLR/KFRA	SAdam
AdaBayes	ARSG	L4Adam/L4Momentum	Sadam/SAMSGrad
AdaBelief	ASAM	LAMB	SALR
AdaBlock	AutoLRS	LaProp	SAM
AdaBound	AvaGrad	LARS	SC-Adagrad/-RMSProp
AdaComp	BAdam	LHOPT	SDProp
Adadelta	BGAdam	LookAhead	SGD
Adafactor	BPGrad	M-SVAG	SGD-BB
AdaFix	BRMSProp	MADGRAD	SGD-G2
AdaFom	BSGD	MAS	SGDEM
AdaFTRL	C-ADAM	MEKA	SGDHess
Adagrad	CADA	MTAdam	SGDM
ADAHESSIAN	Cool Momentum	MVRC-1/MVRC-2	SGDR
Adai	CProp	Nadam	SHAdagrad
AdaLoss	Curveball	NAMSB/NAMSG	Shampoo
Adam	Dadam	ND-Adam	SignAdam++
Adam$^+$	DeepMemory	Nero	SignSGD
AdamAL	DGNOpt	Nesterov	SKQN/S4QN
AdaMax	DiffGrad	Noisy Adam/Noisy K-FAC	SM3
AdamBS	EAdam	NosAdam	SMG
AdamNC	EKFAC	Novograd	SNGM
AdaMod	Eve	NT-SGD	SoftAdam
AdamP/SGDP	Expectigrad	Padam	SRSGD
AdamT	FastAdaBelief	PAGE	Step-Tuned SGD
AdamW	FRSGD	PAL	SWATS
AdamX	G-AdaGrad	PolyAdam	SWNTS
ADAS	GADAM	Polyak	TAdam
AdaS	Gadam	PowerSGD/PowerSGDM	TEKFAC
AdaScale	GOALS	Probabilistic Polyak	VAdam
AdaSGD	GOLS-I	ProbLS	VR-SGD
AdaShift	Grad-Avg	PStorm	vSGD-b/vSGD-g/vSGD-l
AdaSqrt	GRAPES	QHAdam/QHM	vSGD-fd
Adathm	Gravilon	RAdam	WNGrad
AdaX/AdaX-W	Gravity	Ranger	YellowFin
AEGD	HAdam	RangerLars	Yogi
ALI-G	HyperAdam		

The State of Neural Network Training

A crowded field of methods and hyperparameters

A huge number of optimization methods

AcceleGrad	AMSBound	K-BFGS/K-BFGS(L)	RMSProp
ACClip	AMSGrad	KF-QN-CNN	RMSterov
AdaAlter	AngularGrad	KFAC	S-SGD
AdaBatch	ArmijoLS	KFLR/KFRA	SAdam
AdaBayes	ARSG	L4Adam/L4Momentum	Sadam/SAMSGrad
AdaBelief	ASAM	LAMB	SALR
AdaBlock	AutoLRS	LaProp	SAM
AdaBound	AvaGrad	LARS	SC-Adagrad/-RMSProp
AdaComp	BAdam	LHOPT	SDProp
Adadelta	BGAdam	LookAhead	SGD
Adafactor	BPGrad	M-SVAG	SGD-BB
AdaFix	BRMSProp	MADGRAD	SGD-G2
AdaFom	BSGD	MAS	SGDEM
AdaFTRL	C-ADAM	MEKA	SGDHess
Adagrad	CADA	MTAdam	SGDM
ADAHESSIAN	Cool Momentum	MVRC-1/MVRC-2	SGDR
Adai	CProp	Nadam	SHAdagrad
AdaLoss	Curveball	NAMSB/NAMSG	Shampoo
Adam	Dadam	ND-Adam	SignAdam++
Adam^{+}	DeepMemory	Nero	SignSGD
AdamAL	DGNOpt	Nesterov	SKQN/S4QN
AdaMax	DiffGrad	Noisy Adam/Noisy K-FAC	SM3
AdamBS	EAdam	NosAdam	SMG
AdamNC	EKFAC	Novograd	SNGM
AdaMod	Eve	NT-SGD	SoftAdam
AdamP/SGDP	Expectigrad	Padam	SRSGD
AdamT	FastAdaBelief	PAGE	Step-Tuned SGD
AdamW	FRSGD	PAL	SWATS
AdamX	G-AdaGrad	PolyAdam	SWNTS
ADAS	GADAM	Polyak	TAdam
AdaS	Gadam	PowerSGD/PowerSGDM	TEKFAC
AdaScale	GOALS	Probabilistic Polyak	VAdam
AdaSGD	GOLS-I	ProbLS	VR-SGD
AdaShift	Grad-Avg	PStorm	vSGD-b/vSGD-g/vSGD-l
AdaSqrt	GRAPES	QHAdam/QHM	vSGD-fd
Adathm	Gravilon	RAdam	WNGrad
AdaX/AdaX-W	Gravity	Ranger	YellowFin
AEGD	HAdam	RangerLars	Yogi
ALI-G	HyperAdam		

Tuning hyperparameters

SGD:

$$\theta = \theta - \boxed{\eta}\, \boldsymbol{g}$$

The State of Neural Network Training

A crowded field of methods and hyperparameters

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

A huge number of optimization methods

AcceleGrad	AMSBound	K-BFGS/K-BFGS(L)	RMSProp
ACClip	AMSGrad	KF-QN-CNN	RMSterov
AdaAlter	AngularGrad	KFAC	S-SGD
AdaBatch	ArmijoLS	KFLR/KFRA	SAdam
AdaBayes	ARSG	L4Adam/L4Momentum	Sadam/SAMSGrad
AdaBelief	ASAM	LAMB	SALR
AdaBlock	AutoLRS	LaProp	SAM
AdaBound	AvaGrad	LARS	SC-Adagrad/-RMSProp
AdaComp	BAdam	LHOPT	SDProp
Adadelta	BGAdam	LookAhead	SGD
Adafactor	BPGrad	M-SVAG	SGD-BB
AdaFix	BRMSProp	MADGRAD	SGD-G2
AdaFom	BSGD	MAS	SGDEM
AdaFTRL	C-ADAM	MEKA	SGDHess
Adagrad	CADA	MTAdam	SGDM
ADAHESSIAN	Cool Momentum	MVRC-1/MVRC-2	SGDR
Adai	CProp	Nadam	SHAdagrad
AdaLoss	Curveball	NAMSB/NAMSG	Shampoo
Adam	Dadam	ND-Adam	SignAdam++
Adam^{+}	DeepMemory	Nero	SignSGD
AdamAL	DGNOpt	Nesterov	SKQN/S4QN
AdaMax	DiffGrad	Noisy Adam/Noisy K-FAC	SM3
AdamBS	EAdam	NosAdam	SMG
AdamNC	EKFAC	Novograd	SNGM
AdaMod	Eve	NT-SGD	SoftAdam
AdamP/SGDP	Expectigrad	Padam	SRSGD
AdamT	FastAdaBelief	PAGE	Step-Tuned SGD
AdamW	FRSGD	PAL	SWATS
AdamX	G-AdaGrad	PolyAdam	SWNTS
ADAS	GADAM	Polyak	TAdam
AdaS	Gadam	PowerSGD/PowerSGDM	TEKFAC
AdaScale	GOALS	Probabilistic Polyak	VAdam
AdaSGD	GOLS-I	ProbLS	VR-SGD
AdaShift	Grad-Avg	PStorm	vSGD-b/vSGD-g/vSGD-l
AdaSqrt	GRAPES	QHAdam/QHM	vSGD-fd
Adathm	Gravilon	RAdam	WNGrad
AdaX/AdaX-W	Gravity	Ranger	YellowFin
AEGD	HAdam	RangerLars	Yogi
ALI-G	HyperAdam		

Tuning hyperparameters

SGD:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta\, \boldsymbol{g}$$

ADAM:

$$\boldsymbol{m} = \beta_1 \boldsymbol{m} + (1-\beta_1)\boldsymbol{g}$$

$$\boldsymbol{v} = \beta_2 \boldsymbol{v} + (1-\beta_2)\boldsymbol{g} \odot \boldsymbol{g}$$

$$\hat{\boldsymbol{m}} = \frac{\boldsymbol{m}}{1-\beta_1^t} \qquad \hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{1-\beta_2^t}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}} + \varepsilon} \odot \hat{\boldsymbol{m}}$$

The State of Neural Network Training

A crowded field of methods and hyperparameters

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

A huge number of optimization methods

AcceleGrad	AMSBound	K-BFGS/K-BFGS(L)	RMSProp
ACClip	AMSGrad	KF-QN-CNN	RMSterov
AdaAlter	AngularGrad	KFAC	S-SGD
AdaBatch	ArmijoLS	KFLR/KFRA	SAdam
AdaBayes	ARSG	L4Adam/L4Momentum	Sadam/SAMSGrad
AdaBelief	ASAM	LAMB	SALR
AdaBlock	AutoLRS	LaProp	SAM
AdaBound	AvaGrad	LARS	SC-Adagrad/-RMSProp
AdaComp	BAdam	LHOPT	SDProp
Adadelta	BGAdam	LookAhead	SGD
Adafactor	BPGrad	M-SVAG	SGD-BB
AdaFix	BRMSProp	MADGRAD	SGD-G2
AdaFom	BSGD	MAS	SGDEM
AdaFTRL	C-ADAM	MEKA	SGDHess
Adagrad	CADA	MTAdam	SGDM
ADAHESSIAN	Cool Momentum	MVRC-1/MVRC-2	SGDR
Adai	CProp	Nadam	SHAdagrad
AdaLoss	Curveball	NAMSB/NAMSG	Shampoo
Adam	Dadam	ND-Adam	SignAdam++
Adam$^+$	DeepMemory	Nero	SignSGD
AdamAL	DGNOpt	Nesterov	SKQN/S4QN
AdaMax	DiffGrad	Noisy Adam/Noisy K-FAC	SM3
AdamBS	EAdam	NosAdam	SMG
AdamNC	EKFAC	Novograd	SNGM
AdaMod	Eve	NT-SGD	SoftAdam
AdamP/SGDP	Expectigrad	Padam	SRSGD
AdamT	FastAdaBelief	PAGE	Step-Tuned SGD
AdamW	FRSGD	PAL	SWATS
AdamX	G-AdaGrad	PolyAdam	SWNTS
ADAS	GADAM	Polyak	TAdam
AdaS	Gadam	PowerSGD/PowerSGDM	TEKFAC
AdaScale	GOALS	Probabilistic Polyak	VAdam
AdaSGD	GOLS-I	ProbLS	VR-SGD
AdaShift	Grad-Avg	PStorm	vSGD-b/vSGD-g/vSGD-l
AdaSqrt	GRAPES	QHAdam/QHM	vSGD-fd
Adathm	Gravilon	RAdam	WNGrad
AdaX/AdaX-W	Gravity	Ranger	YellowFin
AEGD	HAdam	RangerLars	Yogi
ALI-G	HyperAdam		

Benchmarks

Tuning hyperparameters

SGD:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boxed{\eta}\, \boldsymbol{g}$$

ADAM:

$$\boldsymbol{m} = \boxed{\beta_1}\, \boldsymbol{m} + (1 - \beta_1)\boldsymbol{g}$$

$$\boldsymbol{v} = \boxed{\beta_2}\, \boldsymbol{v} + (1 - \beta_2)\boldsymbol{g} \odot \boldsymbol{g}$$

$$\hat{\boldsymbol{m}} = \frac{\boldsymbol{m}}{1 - \beta_1^t} \qquad \hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{1 - \beta_2^t}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\boxed{\eta}}{\sqrt{\hat{\boldsymbol{v}}} + \boxed{\varepsilon}} \odot \hat{\boldsymbol{m}}$$

The State of Neural Network Training

A crowded field of methods and hyperparameters

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

A huge number of optimization methods

AcceleGrad
ACClip
AdaAlter
AdaBatch
AdaBayes
AdaBelief
AdaBlock
AdaBound
AdaComp
Adadelta
Adafactor
AdaFix
AdaFom
AdaFTRL
Adagrad
ADAHESSIAN
Adai
AdaLoss
Adam
Adam^{+}
AdamAL
AdaMax
AdamBS
AdamNC
AdaMod
AdamP/SGDP
AdamT
AdamW
AdamX
ADAS
AdaS
AdaScale
AdaSGD
AdaShift
AdaSqrt
Adathm
AdaX/AdaX-W
AEGD
ALI-G
AMSBound
AMSGrad
AngularGrad
ArmijoLS
ARSG
ASAM
AutoLRS
AvaGrad
BAdam
BGAdam
BPGrad
BRMSProp
BSGD
C-ADAM
CADA
Cool Momentum
CProp
Curveball
Dadam
DeepMemory
DGNOpt
DiffGrad
EAdam
EKFAC
Eve
Expectigrad
FastAdaBelief
FRSGD
G-AdaGrad
GADAM
Gadam
GOALS
GOLS-I
Grad-Avg
GRAPES
Gravilon
Gravity
HAdam
HyperAdam
K-BFGS/K-BFGS(L)
KF-QN-CNN
KFAC
KFLR/KFRA
L4Adam/L4Momentum
LAMB
LaProp
LARS
LHOPT
LookAhead
M-SVAG
MADGRAD
MAS
MEKA
MTAdam
MVRC-1/MVRC-2
Nadam
NAMSB/NAMSG
ND-Adam
Nero
Nesterov
Noisy Adam/Noisy K-FAC
NosAdam
Novograd
NT-SGD
Padam
PAGE
PAL
PolyAdam
Polyak
PowerSGD/PowerSGDM
Probabilistic Polyak
ProbLS
PStorm
QHAdam/QHM
RAdam
Ranger
RangerLars
RMSProp
RMSterov
S-SGD
SAdam
Sadam/SAMSGrad
SALR
SAM
SC-Adagrad/-RMSProp
SDProp
SGD
SGD-BB
SGD-G2
SGDEM
SGDHess
SGDM
SGDR
SHAdagrad
Shampoo
SignAdam++
SignSGD
SKQN/S4QN
SM3
SMG
SNGM
SoftAdam
SRSGD
Step-Tuned SGD
SWATS
SWNTS
TAdam
TEKFAC
VAdam
VR-SGD
vSGD-b/vSGD-g/vSGD-l
vSGD-fd
WNGrad
YellowFin
Yogi

Tuning hyperparameters

SGD:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \boldsymbol{g}$$

ADAM:

$$\boldsymbol{m} = \beta_1 \boldsymbol{m} + (1 - \beta_1)\boldsymbol{g}$$

$$\boldsymbol{v} = \beta_2 \boldsymbol{v} + (1 - \beta_2)\boldsymbol{g} \odot \boldsymbol{g}$$

$$\hat{\boldsymbol{m}} = \frac{\boldsymbol{m}}{1 - \beta_1^t} \qquad \hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{1 - \beta_2^t}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}} + \varepsilon} \odot \hat{\boldsymbol{m}}$$

Benchmarks

Debugging Tools

Key Takeaways I

The fundamentals of neural network training

- **Training $\neq$ optimization**
  A fundamental and crucial difference.
- **Stochasticity is a core property**
  A primary source of the challenge.
- **Training requires more than an optimizer**
  Update rule + hyperparameters + tuning methods + schedules + ...

Part II

Benchmarking Training Algorithms

The quest for finding the state of the art in neural network training

The Benchmarking Process

Many dimensions to explore at once

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Schmidt et al. 2021)

$$\overset{\textbf{Optimizer}}{\begin{Bmatrix} \textsc{Adam} \\ \text{NAG} \\ \cdots \\ \text{SGD} \end{Bmatrix}_{15}} \times \overset{\textbf{Problem}}{\begin{Bmatrix} \text{P1} \\ \text{P2} \\ \cdots \\ \text{P8} \end{Bmatrix}_{8}} \times \overset{\textbf{Tuning}}{\begin{Bmatrix} \text{one-shot} \\ \text{small} \\ \text{medium} \\ \text{large} \end{Bmatrix}_{4}} \times \overset{\textbf{Schedule}}{\begin{Bmatrix} \text{constant} \\ \text{cosine} \\ \text{cosine wr} \\ \text{trapezoidal} \end{Bmatrix}_{4}}$$

The Benchmarking Process

Many dimensions to explore at once

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Schmidt et al. 2021)

Optimizer		Problem		Tuning		Schedule

$$\left\{\begin{array}{c}\text{Adam}\\ \text{NAG}\\ \cdots\\ \text{SGD}\end{array}\right\}_{15} \times \left\{\begin{array}{c}\text{P1}\\ \text{P2}\\ \cdots\\ \text{P8}\end{array}\right\}_{8} \times \left\{\begin{array}{c}\text{one-shot}\\ \text{small}\\ \text{medium}\\ \text{large}\end{array}\right\}_{4} \times \left\{\begin{array}{c}\text{constant}\\ \text{cosine}\\ \text{cosine wr}\\ \text{trapezoidal}\end{array}\right\}_{4}$$

- AMSBound
- AMSGrad
- AdaBelief
- AdaBound
- Adadelta
- AdaGrad
- Adam
- LA(Mom.)
- LA(RAdam)
- Momentum
- NAG
- Nadam
- RAdam
- RMSProp
- SGD

The Benchmarking Process

Many dimensions to explore at once

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Schmidt et al. 2021)

Optimizer		Problem		Tuning		Schedule
ADAM		P1		one-shot		constant
NAG		P2		small		cosine
...	×	...	×	medium	×	cosine wr
SGD		P8		large		trapezoidal
15		8		4		4

	Data set	Model	Task	Metric	Batch size	Budget *in epochs*	Approx. run time
P1	Artificial	**Noisy quadratic**	Minimization	Loss	128	100	$<$ 1 min
P2	MNIST	**VAE**	Generative	Loss	64	50	10 min
P3	Fashion-MNIST	**Simple CNN: *2c2d***	Classification	Accuracy	128	100	20 min
P4	CIFAR-10	**Simple CNN: *3c3d***	Classification	Accuracy	128	100	35 min
P5	Fashion-MNIST	**VAE**	Generative	Loss	64	100	20 min
P6	CIFAR-100	***All-CNN-C***	Classification	Accuracy	256	350	4 h 00 min
P7	SVHN	***Wide ResNet 16-4***	Classification	Accuracy	128	160	3 h 30 min
P8	War and Peace	**RNN**	Character Prediction	Accuracy	50	200	5 h 30 min

The Benchmarking Process

Many dimensions to explore at once

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Schmidt et al. 2021)

$$\underset{\textbf{Optimizer}}{\begin{Bmatrix} \text{Adam} \\ \text{NAG} \\ \cdots \\ \text{SGD} \end{Bmatrix}_{15}} \times \underset{\textbf{Problem}}{\begin{Bmatrix} \text{P1} \\ \text{P2} \\ \cdots \\ \text{P8} \end{Bmatrix}_{8}} \times \underset{\textbf{Tuning}}{\begin{Bmatrix} \text{one-shot} \\ \text{small} \\ \text{medium} \\ \text{large} \end{Bmatrix}_{4}} \times \underset{\textbf{Schedule}}{\begin{Bmatrix} \text{constant} \\ \text{cosine} \\ \text{cosine wr} \\ \text{trapezoidal} \end{Bmatrix}_{4}}$$

- **One-Shot** - 1 Run
  No tuning, uses default hyperparameters
- **Small** - 25 Runs
  Tuned via random search
- **Medium** - 50 Runs
  Tuned via random search, superset of *small budget*
- **Large** - 75 Runs
  Tuned via random search, refined search spaces

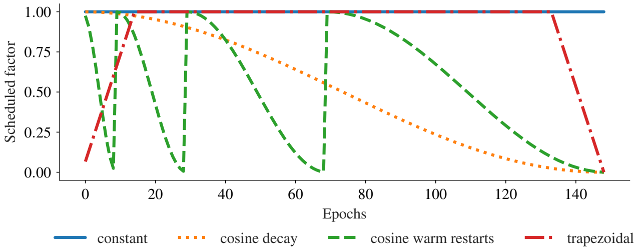
The Benchmarking Process

Many dimensions to explore at once

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Schmidt et al. 2021)

Optimizer		Problem		Tuning		Schedule
{ADAM, NAG, ..., SGD}$_{15}$	×	{P1, P2, ..., P8}$_{8}$	×	{one-shot, small, medium, large}$_{4}$	×	{constant, cosine, cosine wr, trapezoidal}$_{4}$



The Benchmarking Process

Many dimensions to explore at once

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Schmidt et al. 2021)

$$\underset{\textbf{Optimizer}}{\left\{\begin{matrix}\text{Adam}\\\text{NAG}\\\vdots\\\text{SGD}\end{matrix}\right\}_{15}} \times \underset{\textbf{Problem}}{\left\{\begin{matrix}\text{P1}\\\text{P2}\\\vdots\\\text{P8}\end{matrix}\right\}_{8}} \times \underset{\textbf{Tuning}}{\left\{\begin{matrix}\text{one-shot}\\\text{small}\\\text{medium}\\\text{large}\end{matrix}\right\}_{4}} \times \underset{\textbf{Schedule}}{\left\{\begin{matrix}\text{constant}\\\text{cosine}\\\text{cosine wr}\\\text{trapezoidal}\end{matrix}\right\}_{4}}$$

1,920 **Configurations**
fifteen optimizers, eight problems, four budgets, & four schedules

>50,000 **Individual Runs**
includes tuning runs & runs stochastic fidelity

128 **Settings** each optimizer is tested in
eight problems, four budgets, & four schedules

The Benchmarking Process

Many dimensions to explore at once

(Schmidt et al. 2021)

$$\underbrace{\begin{Bmatrix} \textsc{Adam} \\ \text{NAG} \\ \cdots \\ \text{SGD} \end{Bmatrix}_{15}}_{\textbf{Optimizer}} \times \underbrace{\begin{Bmatrix} \text{P1} \\ \text{P2} \\ \cdots \\ \text{P8} \end{Bmatrix}_{8}}_{\textbf{Problem}} \times \underbrace{\begin{Bmatrix} \text{one-shot} \\ \text{small} \\ \text{medium} \\ \text{large} \end{Bmatrix}_{4}}_{\textbf{Tuning}} \times \underbrace{\begin{Bmatrix} \text{constant} \\ \text{cosine} \\ \text{cosine wr} \\ \text{trapezoidal} \end{Bmatrix}_{4}}_{\textbf{Schedule}}$$

1,920 **Configurations**
fifteen optimizers, eight problems, four budgets, & four schedules

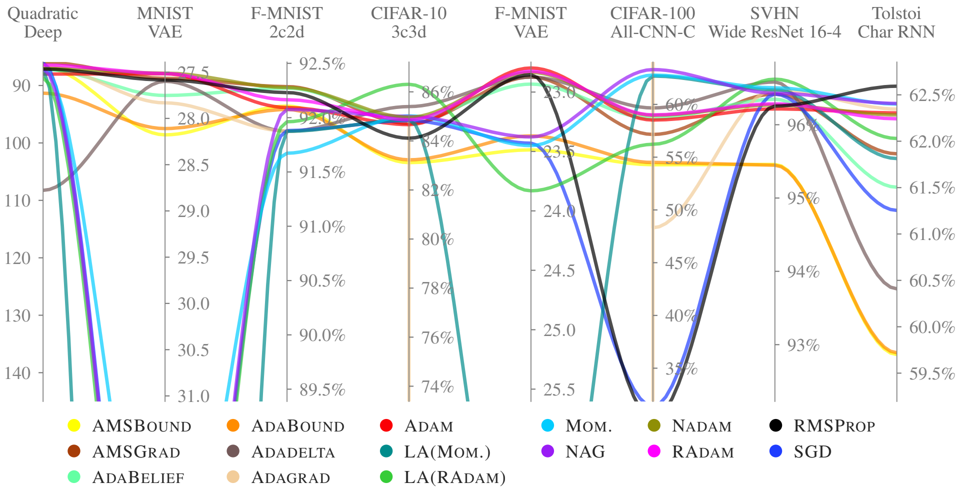
>50,000 **Individual Runs**
includes tuning runs & runs stochastic fidelity

128 **Settings** each optimizer is tested in
eight problems, four budgets, & four schedules

The Benchmarking Results

Why ADAM is still a good choice | *large budget with a trapezoidal schedule*

(Schmidt et al. 2021)

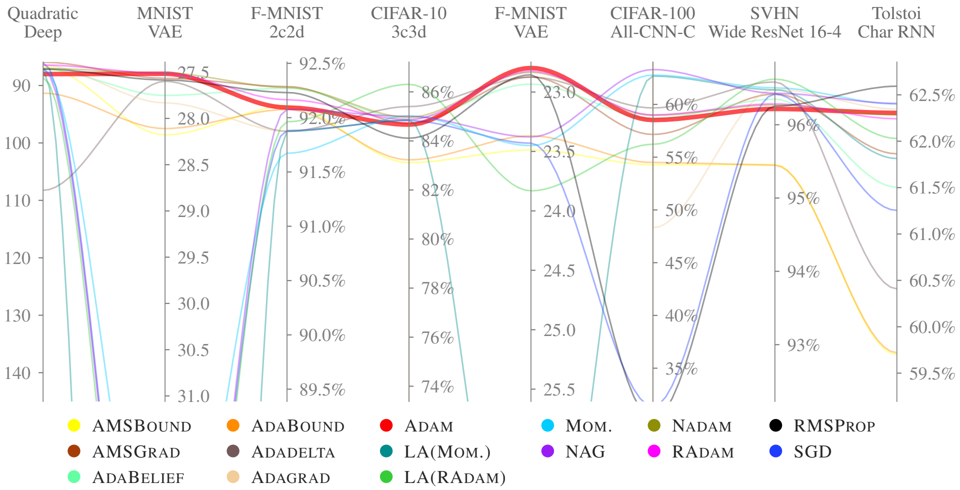


The Benchmarking Results

Why ADAM is still a good choice | *large budget with a trapezoidal schedule*

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

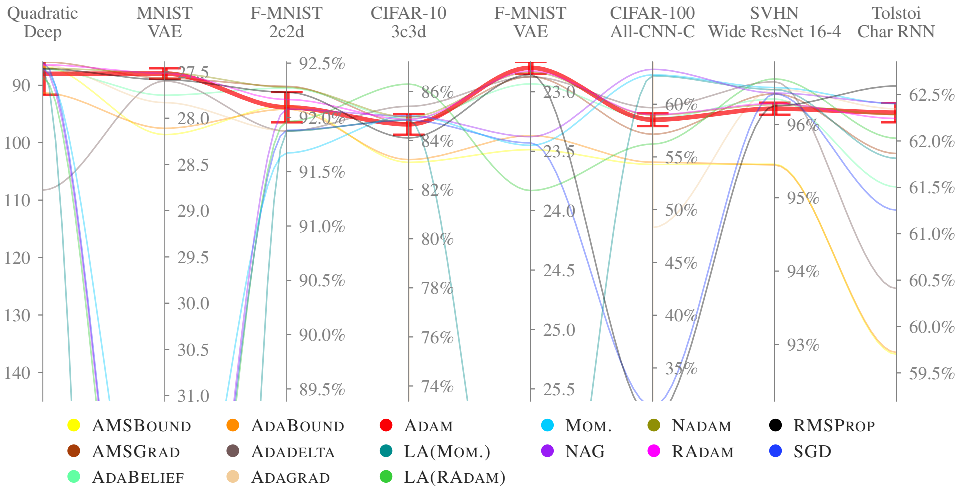
(Schmidt et al. 2021)



The Benchmarking Results

Why ADAM is still a good choice | *large budget with a trapezoidal schedule*

(Schmidt et al. 2021)



MLCommons

A natural extension of our benchmarking efforts

ML Commons

Algorithms Working Group

George Dahl Google

Frank Schneider University of Tübingen

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 workloads
- A huge large-scale effort by 25+ researchers from Google, University of Tübingen, University of Toronto, Meta AI, etc.

Key Takeaways II

Benchmarking neural network training methods

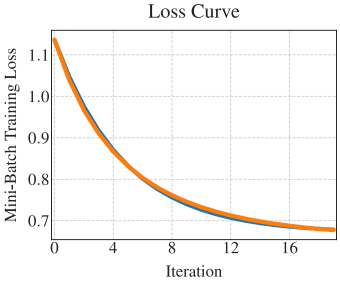
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Unclear SOTA**
  There is no established protocol to train neural networks.
- **Literature only provides families of algorithms**
  The many confounding factors make it hard to benchmark training methods.
- **Babysitting and tuning**
  The existing methods are clearly unsatisfying.

Part III

Novel Debugging Tools

Leveraging all available information for better deep learning debuggers

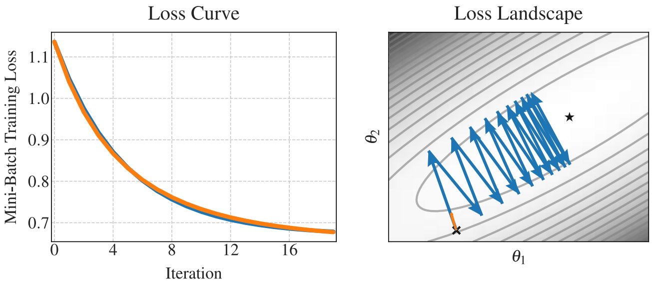


Loss Curve
Mini-Batch Training Loss
1.1
1.0
0.9
0.8
0.7
0
4
8
12
16
Iteration

Loss Curves Do Not Tell the Full Story

Why we need better observables in neural network training

(Schneider et al. 2021)

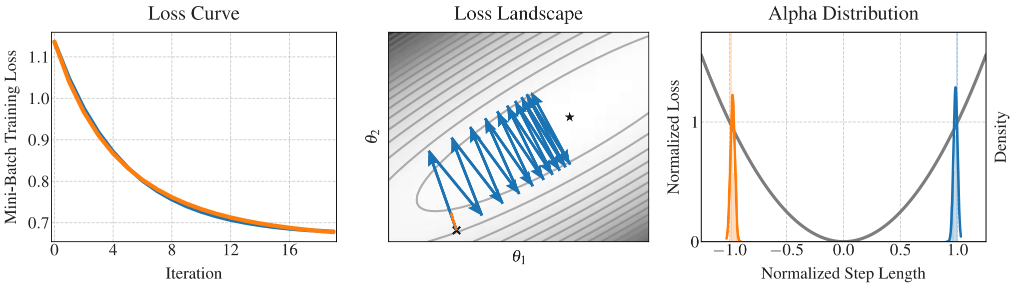


Loss Curves Do Not Tell the Full Story

Why we need better observables in neural network training

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

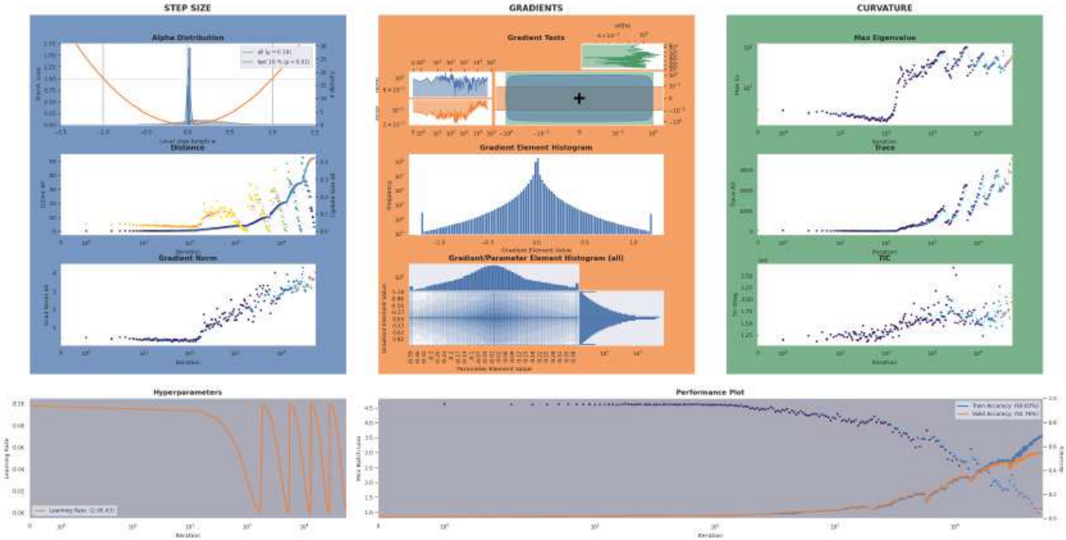
(Schneider et al. 2021)



Cockpit in Action!

Training the All-CNN-C network through the lens of Cockpit

(Schneider et al. 2021)

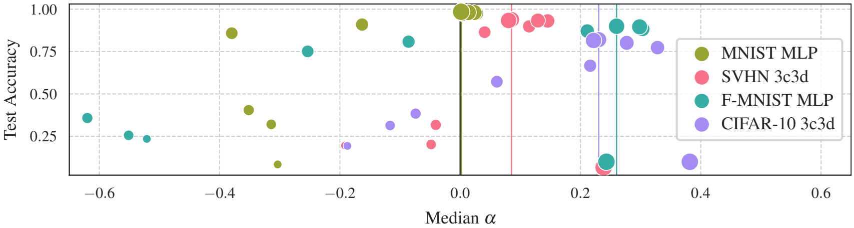


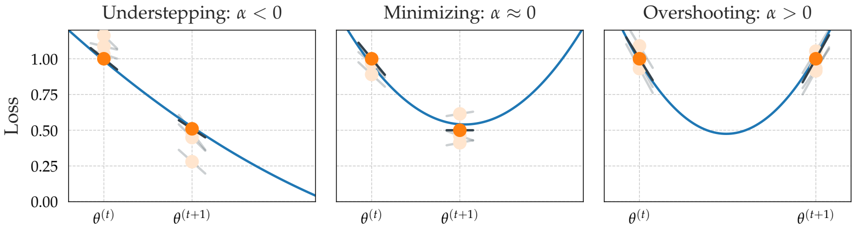
COCKPIT as a Tool for Finding Bugs and Inspiring Research

Neural network training requires systematically overstepping the local minima

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(Schneider et al. 2021)





Key Takeaways III

Debugging tools for deep learning

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Additional available information**
  Inexpensive but often hidden by existing software frameworks.
- **We can leverage it for better debugging tools**
  Or as a path to better autonomous training methods.
- **Practical tips & lessons learned**
  - Start with something easy that was shown to work (or ADAM).
  - Tune your hyperparameters (learning rate + schedule, weight decay, $1 - \beta_1$, etc.).
  - Great resource: "Deep Learning Tuning Playbook".

Summary

- **The current training methods are unsatisfying**: There is no established protocol to train neural networks and all contenders require babysitting and tuning.
- **Benchmarking deep learning algorithms is crucial** but it is challenging and we need a better methodology to find the best training algorithms.
- **Using distributions and confidences** we can built better debugging tools and training methods for neural networks.

with the help of many more

Benchmark	`github.com/SirRob1997/Crowded-Valley---Results`	
MLCommons	`github.com/mlcommons/algorithmic-efficiency`	ML Commons
Cockpit	`pip install cockpit-for-pytorch`	Cockpit