Module 1 Discussion

Jose Bonet
Yusen Wang
Lifei Tang

Measuring the Effects of Data Parallelism on Neural Network Training

Introduction

- **Data parallelism**: Distributing training examples across multiple processors to compute gradient updates
- Parallelism **dichotomy**:
  - **+** Speedup
  - **-** Out-of-sample error
- **Costs** and **benefits** of data parallelism

Scope of the study

1. What is the relationship between batch size and number of training steps to reach a goal out-of-sample error?
2. What governs this relationship?
3. Do large batch sizes incur a cost in out-of-sample error?

Convergence bounds (1)

- Upper bound SGD performance applied to L-Lipschitz convex loss:

$$\underbrace{J(\theta_T) - J^\star}_{\text{Achievable objective}} \leq \underbrace{O\left(\sqrt{\frac{L^2}{T}}\right)}_{\text{Time complexity}}$$

- **No benefit:** Increasing batch size does not change the number of steps to convergence

Convergence bounds (2)

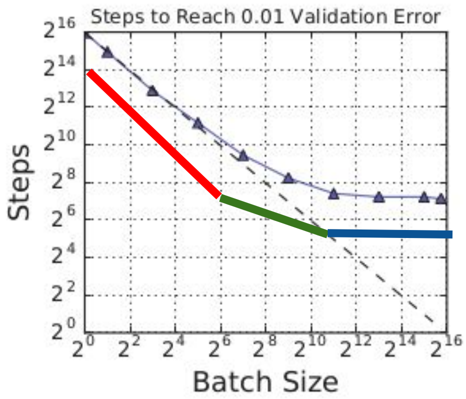
- SGD bound when loss is convex and objective H-smooth:

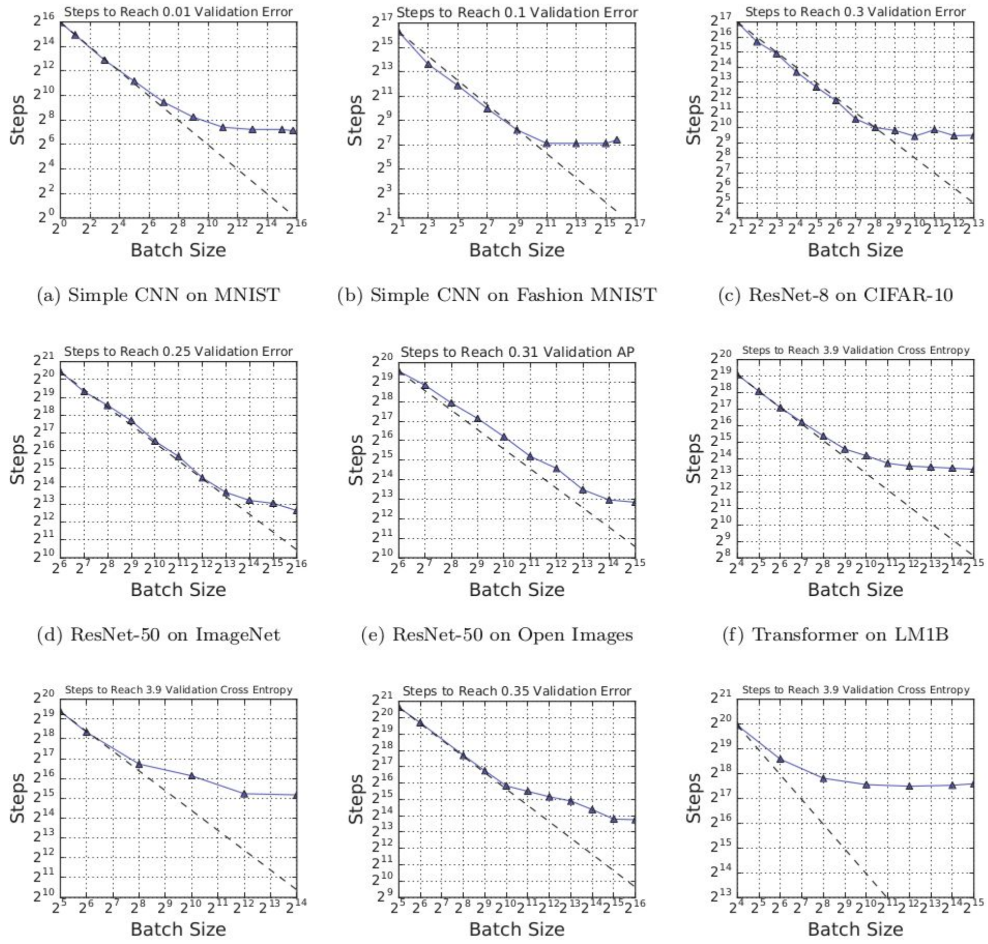
$$\underbrace{J(\theta_T) - J^\star}_{\text{Achievable objective}} \leq O\underbrace{\left(\frac{H}{T^2} + \sqrt{\frac{L^2}{Tb}}\right)}_{\text{Time complexity}}$$

- **A b-fold benefit**: Increasing b decreases T to a given objective value by the same factor

Convergence & study plots

- **Perfect scaling**: b-fold benefit
- **Diminishing returns**
- **Maximal data parallelism**: no benefit



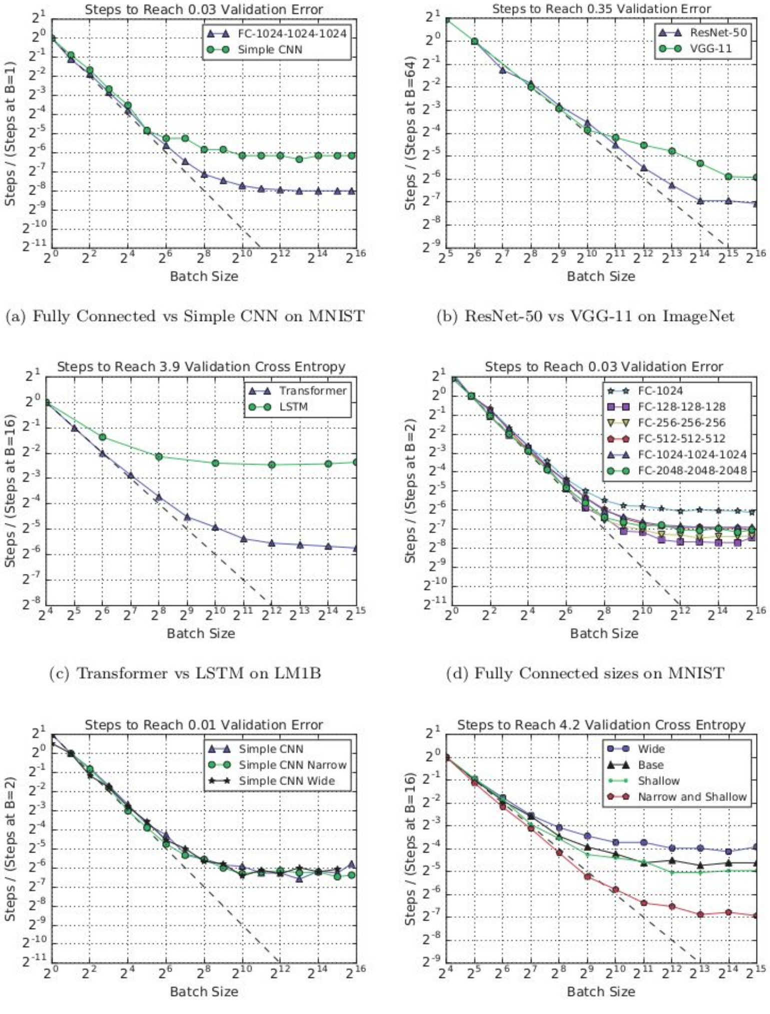


(a) Simple CNN on MNIST (b) Simple CNN on Fashion MNIST (c) ResNet-8 on CIFAR-10

(d) ResNet-50 on ImageNet (e) ResNet-50 on Open Images (f) Transformer on LM1B

(g) Transformer on Common Crawl (h) VGG-11 on ImageNet (i) LSTM on LM1B

Steps to result depends on batch size in a similar way across problems



(a) Fully Connected vs Simple CNN on MNIST

(b) ResNet-50 vs VGG-11 on ImageNet

(c) Transformer vs LSTM on LM1B

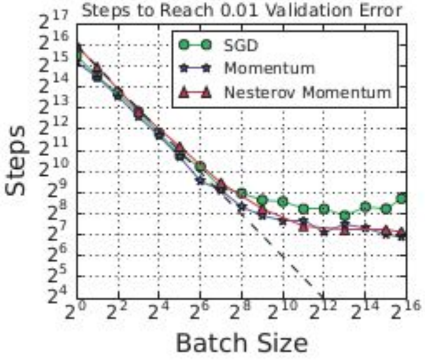
(d) Fully Connected sizes on MNIST

(e) Simple CNN sizes on MNIST

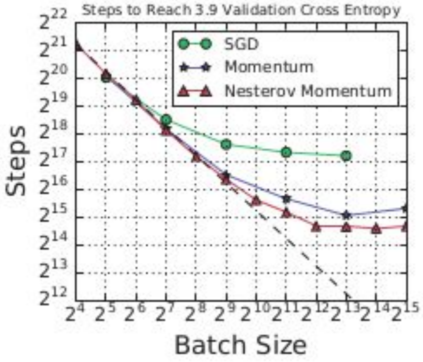
(f) Transformer sizes on LM1B

Not all models scale equally with batch size

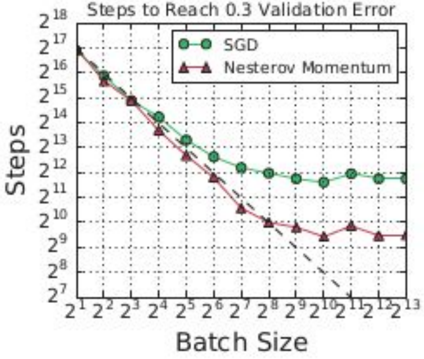
Momentum extends perfect scaling to larger batch sizes



(a) Simple CNN on MNIST



(b) Transformer Shallow on LM1B

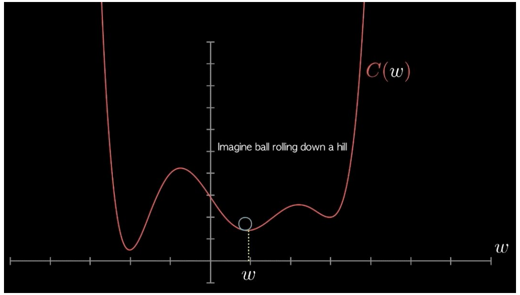


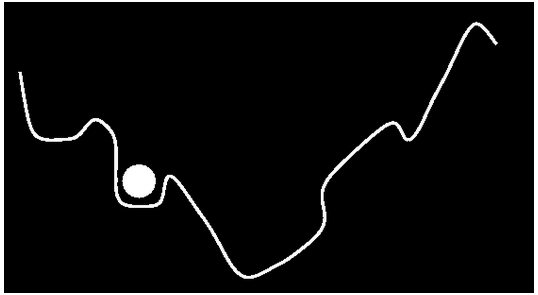
(c) ResNet-8 on CIFAR-10

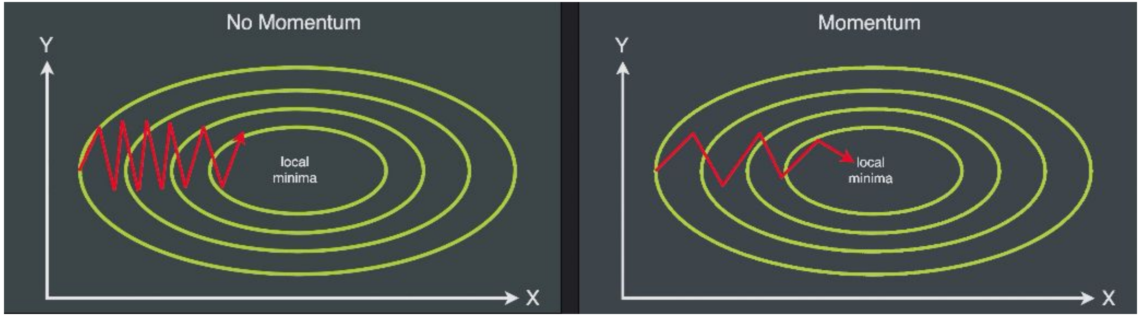
Comment on momentum

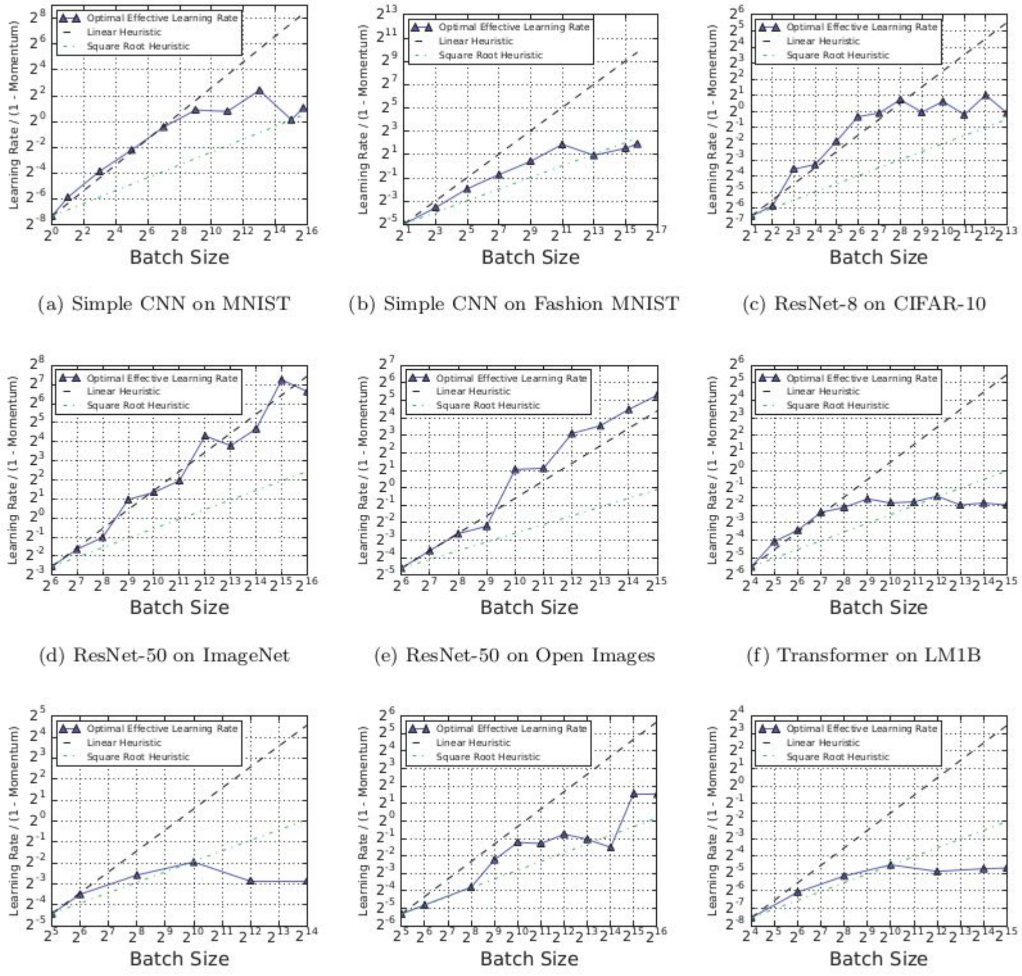
https://mlfromscratch.com/optimizers-explained/#/

https://distill.pub/2017/momentum/







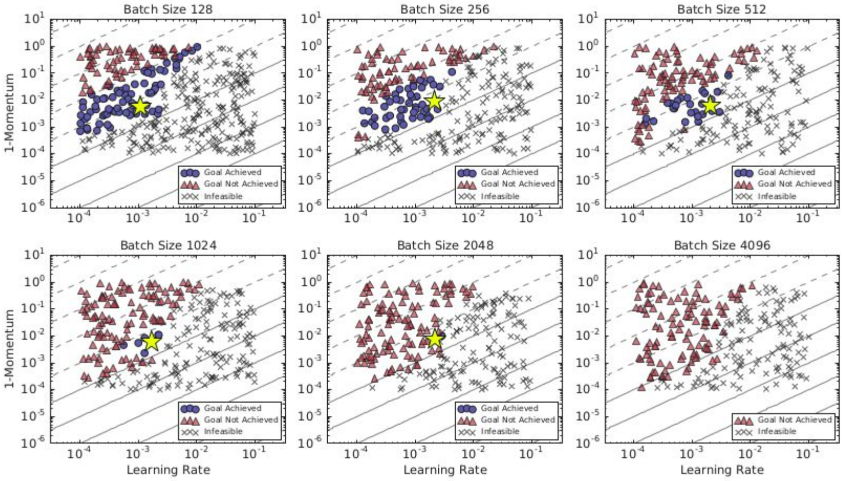


(a) Simple CNN on MNIST
(b) Simple CNN on Fashion MNIST
(c) ResNet-8 on CIFAR-10
(d) ResNet-50 on ImageNet
(e) ResNet-50 on Open Images
(f) Transformer on LM1B
(g) Transformer on Common Crawl
(h) VGG-11 on ImageNet
(i) LSTM on LM1B

Best learning rate and momentum vary with batch size (1)

However, it did not always follow a particular scaling

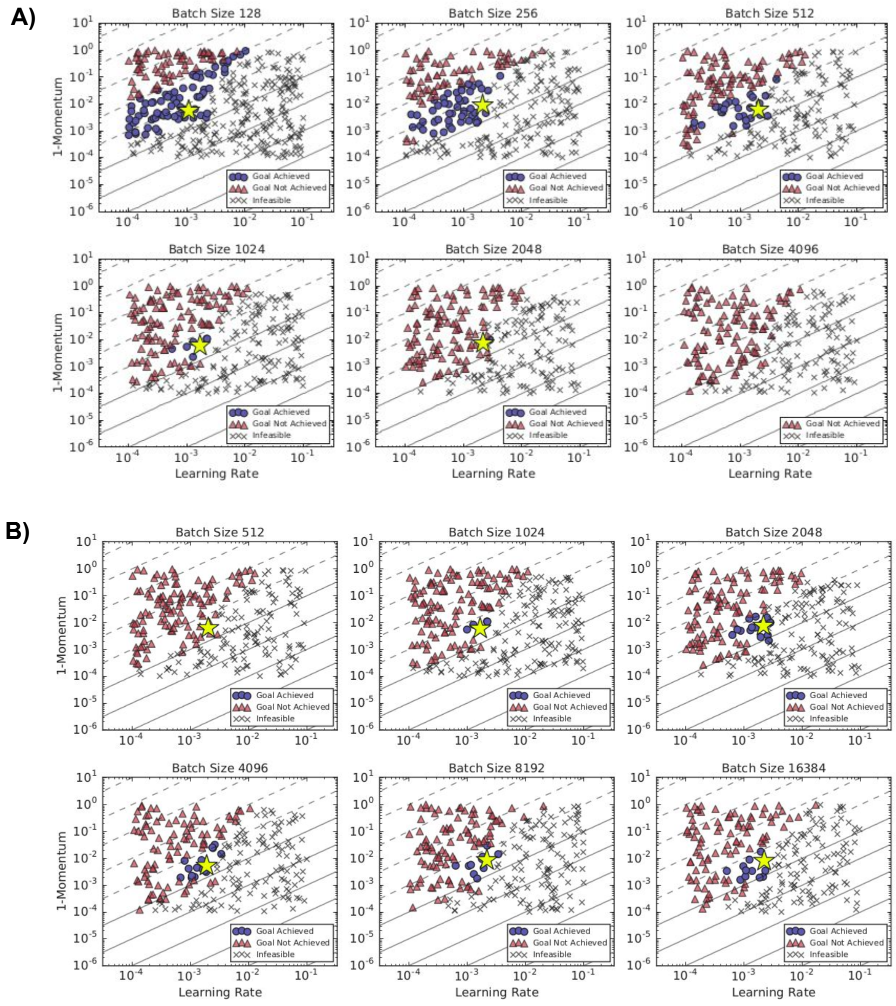
A)



B)

Region in metaparameter space corresponding to rapid training in terms of epochs becomes smaller

Best learning rate and momentum vary with batch size (2)

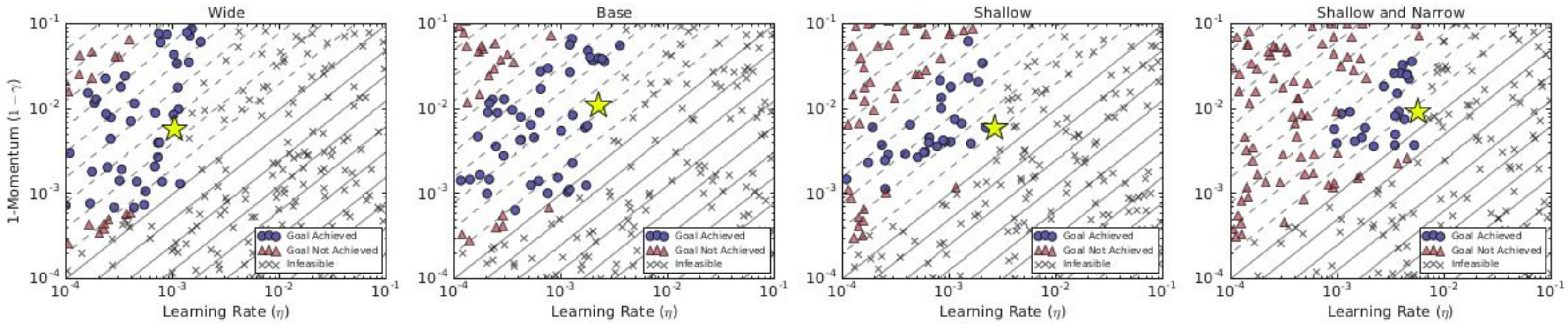


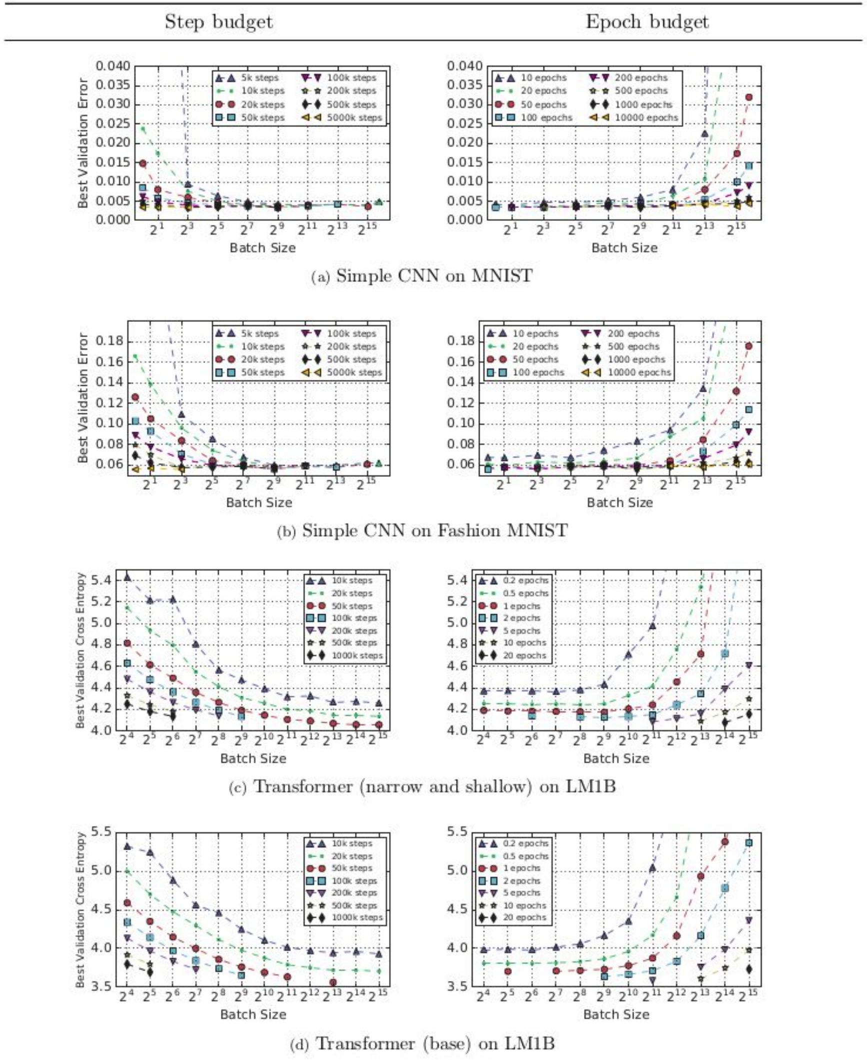
Region in metaparameter space corresponding to rapid training in terms of epochs becomes smaller

Best learning rate and momentum vary with batch size (2)

Region in metaparameter space corresponding to rapid training in terms of step-count grows larger

Best learning rate and momentum vary with batch size (3)





(a) Simple CNN on MNIST

(b) Simple CNN on Fashion MNIST

(c) Transformer (narrow and shallow) on LM1B

(d) Transformer (base) on LM1B

Solution quality depends on compute budget more than batch size

Main contributions

1. Shared relationship between batch size and number of training steps to reach out-of-sample error.
2. Maximum useful batch size varies significantly between workloads and depends on model properties, training algorithm and data set.
3. Optimal values of training metaparameters do not consistently follow any simple relationships with batch size

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

Outline

1. Linear scaling rule
2. Different warmup methods
3. Experiment results and discussion

Large Minibatch SGD

Loss function:

$$L(w) = \frac{1}{|X|} \sum_{x \in X} l(x, w).$$

Minibatch Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{x \in \mathcal{B}} \nabla l(x, w_t).$$

Learning Rates for Large Minibatches

Goal:

Use large minibatches in place of small minibatches while

Maintaining training and generalization accuracy

Linear Scaling Rule:

When the minibatch size is multiplied by k, multiply the learning rate by k

Interpretation of Linear Scaling Rule

After *k* iterations of SGD:

$$w_{t+k} = w_t - \eta \frac{1}{n} \sum_{j<k} \sum_{x \in \mathcal{B}_j} \nabla l(x, w_{t+j})$$

Take a single step of large minibatch:

$$\hat{w}_{t+1} = w_t - \hat{\eta} \frac{1}{kn} \sum_{j<k} \sum_{x \in \mathcal{B}_j} \nabla l(x, w_t).$$

Strong assumption:

If $\nabla l(x, w_t) \approx \nabla l(x, w_{t+j})$ is true,

then setting $\hat{\eta} = k\eta$ would yield $\hat{w}_{t+1} \approx w_{t+k}$

“Strong” Assumption

The approximation might be valid in large-scale, real-world data.

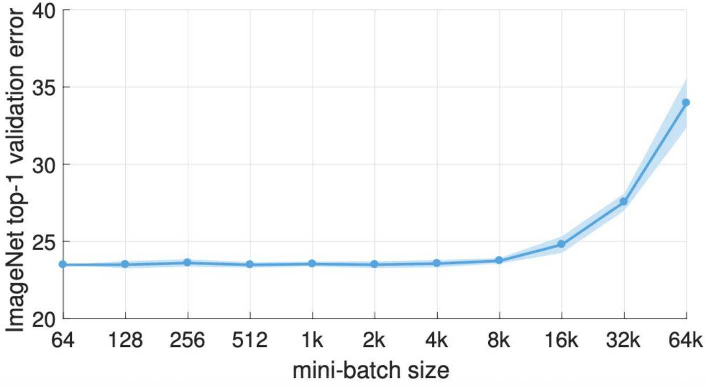
Cases when the condition will not hold:

1. Early stages of training process:

   The network is changing rapidly

2. Minibatch size can not be scaled indefinitely:

   Results are stable for a certain range of sizes



Warmup

Constant warmup vs Gradual warmup

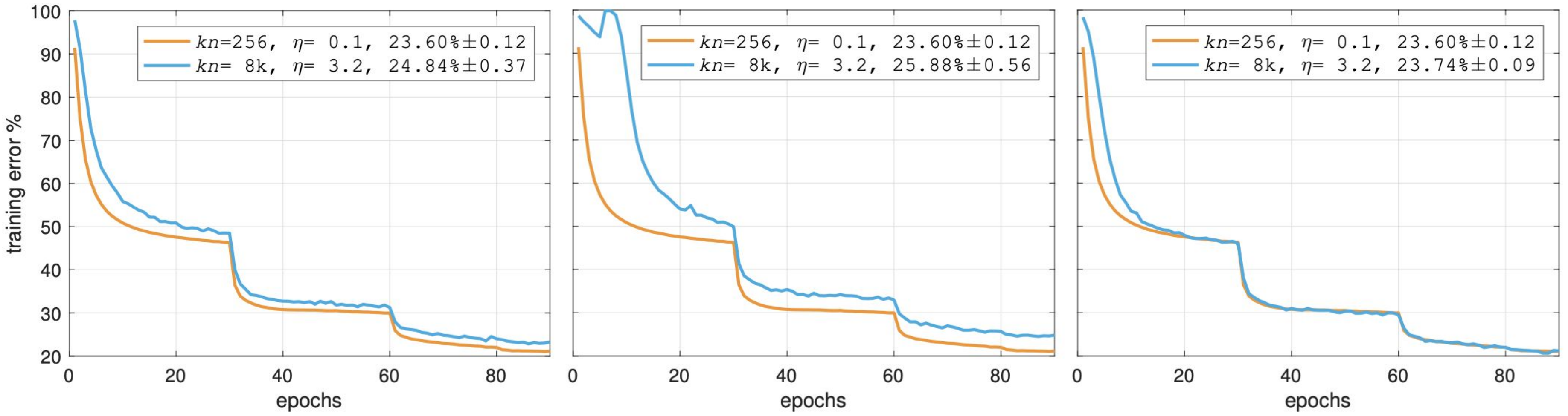
Constant:

Low constant learning rate for first few epochs of training

Gradual:

Increase learning rate by a constant amount at each iteration

Different Warmup Methods



(a) no warmup (b) constant warmup (c) gradual warmup

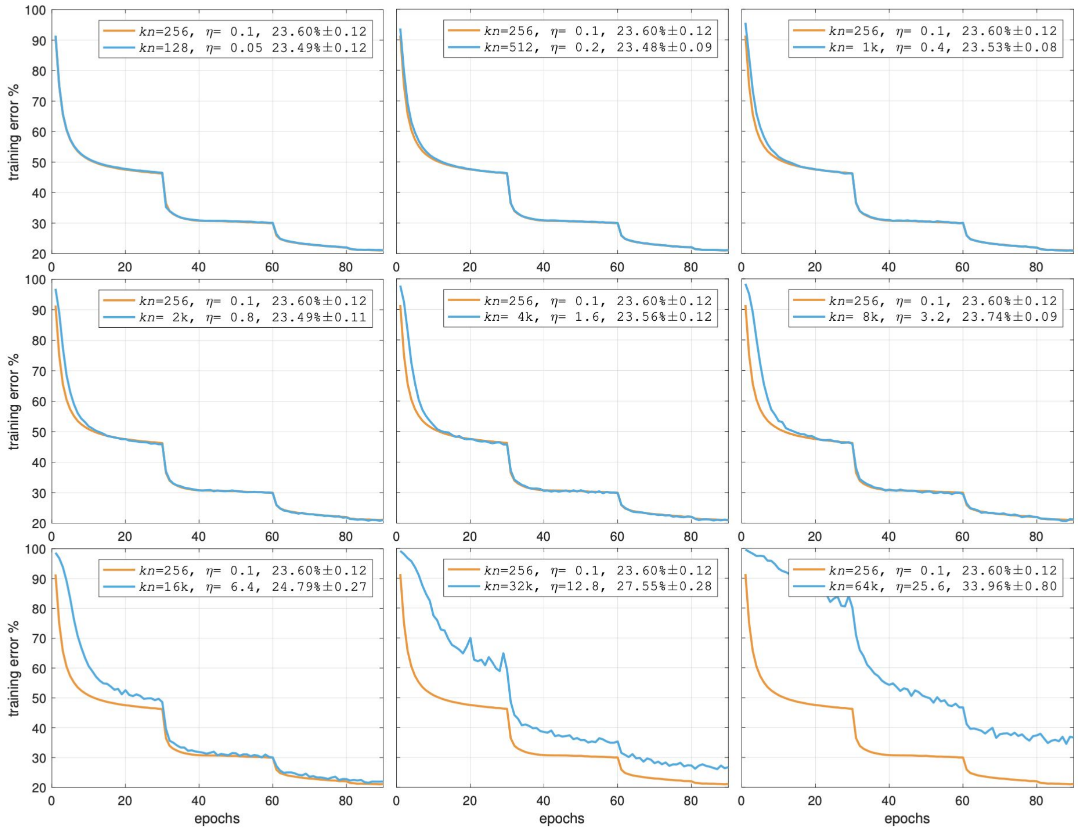
Training curves for different warmup methods

Different Warmup Methods

	k	n	kn	η	top-1 error (%)
baseline (single server)	8	32	256	0.1	23.60 $\pm$0.12
no warmup, Figure 2a	256	32	8k	3.2	24.84 $\pm$0.37
constant warmup, Figure 2b	256	32	8k	3.2	25.88 $\pm$0.56
gradual warmup, Figure 2c	256	32	8k	3.2	23.74 $\pm$0.09

Validation error for different warmup methods

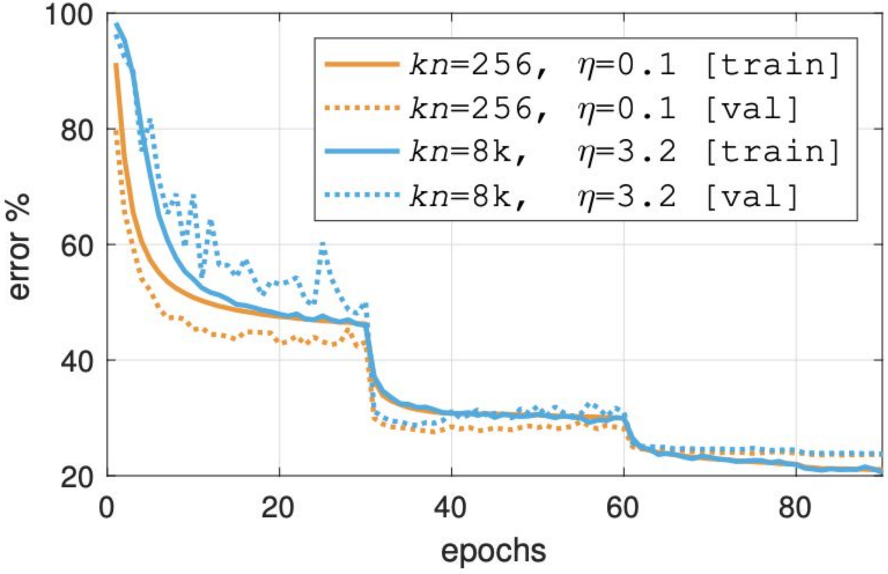
Training error vs Minibatch Size



The stability break down when minibatch size exceeds 8k

(1k = 1024)

Training and Validation Curves



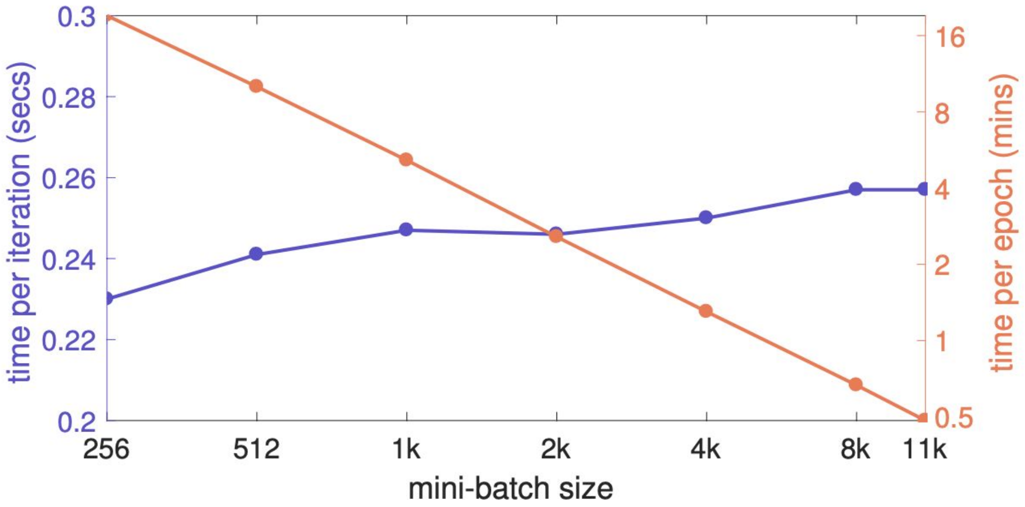
Training and validation curves for

Large minibatch SGD with gradual warmup

vs

Small minibatch SGD

Distributed synchronous SGD Timing



Time per iteration is relatively flat

Time per epoch decreases from over 16 minutes to 30 seconds

Main Contributions

1. Proposed the linear scaling rule for learning rate.

2. Proposed the gradual warmup method for early stages of training process, performance is compared with no warmup and constant warmup.

3. Dramatically decrease the training time of ImageNet while maintaining training and validation accuracy

CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers

Interesting Quotes

“In practice, batch sizes of 64,000 are now not uncommon”

- CROSSBOW, 2018

“Training with large mini-batches is bad for your health. More importantly, it’s bad for your test error. Friends don’t let friends use mini-batches larger than 32.”

- Yann LeCun (@ylecun), April 2018

Motivation:

- To scale deep learning training on Multiple GPUs
- Large batch size, better hardware efficiency, but poorer statistical efficiency.

CROSSBOW: a single-server multi-GPU system for training deep learning models that enables users to freely choose their preferred batch size.

Research questions:

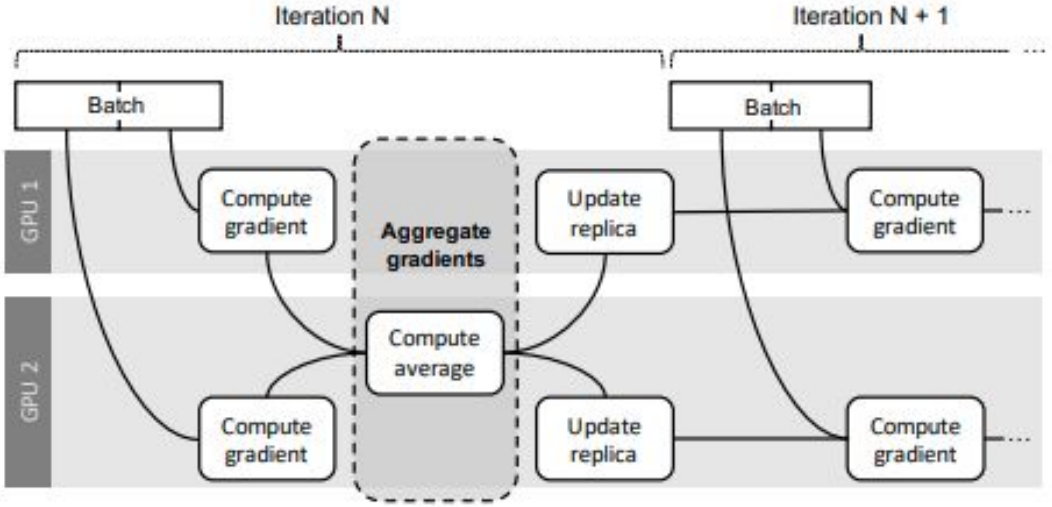
- How to synchronize model replicas without adversely affecting statistical efficiency
- How to ensure high hardware (GPU) efficiency?

Methods:

- **SMA**: **Synchronous Model Averaging**
- **Training multi learner per GPU**
  - Auto-tuning the number of learners
  - Concurrent task engine

SMA vs SSGD:

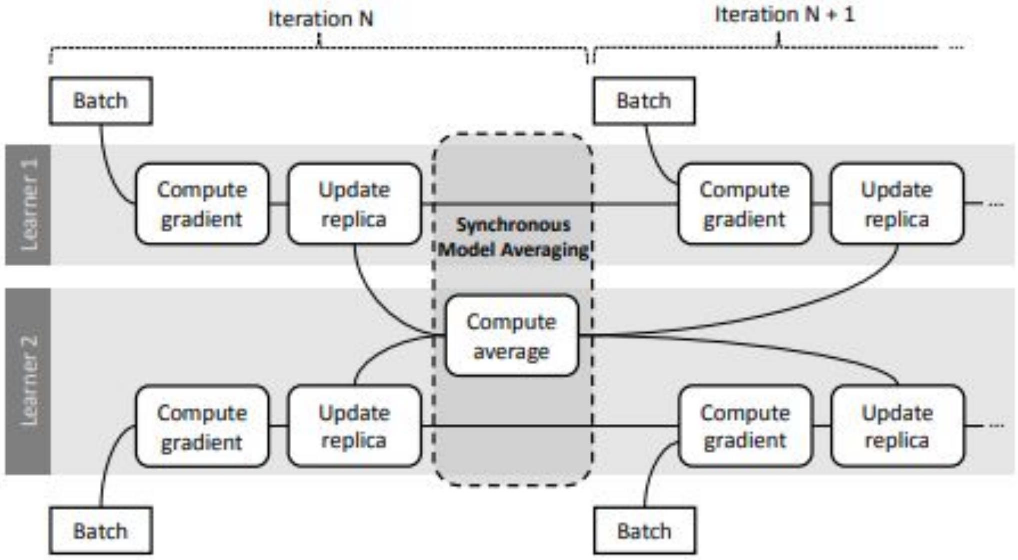
In Synchronous SGD (S-SGD), learner is dependent



(CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers)

SMA vs SSGD:

in SMA, learner is independent



(CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers)

SMA Replica updates:

$learner: \; j \in 1, ..., k, \; B_j: \; batch \; for \; j \; th \; learner:$

$$g_j \leftarrow \gamma \nabla \ell_{B_j}(\mathsf{w}_j) \;; \qquad \text{// Gradient for replica } j$$
$$c_j \leftarrow \alpha(\mathsf{w}_j - \mathsf{z}) \;; \qquad \text{// Correction for replica } j$$
$$\mathsf{w}_j \leftarrow \mathsf{w}_j - g_j - c_j \;; \qquad \text{// Update replica } j$$

Note : $\alpha \approx 1/k$, *z is the average model*

Model Aggregation:

learner : $j \in 1, \ldots, k$, *z is the average model*

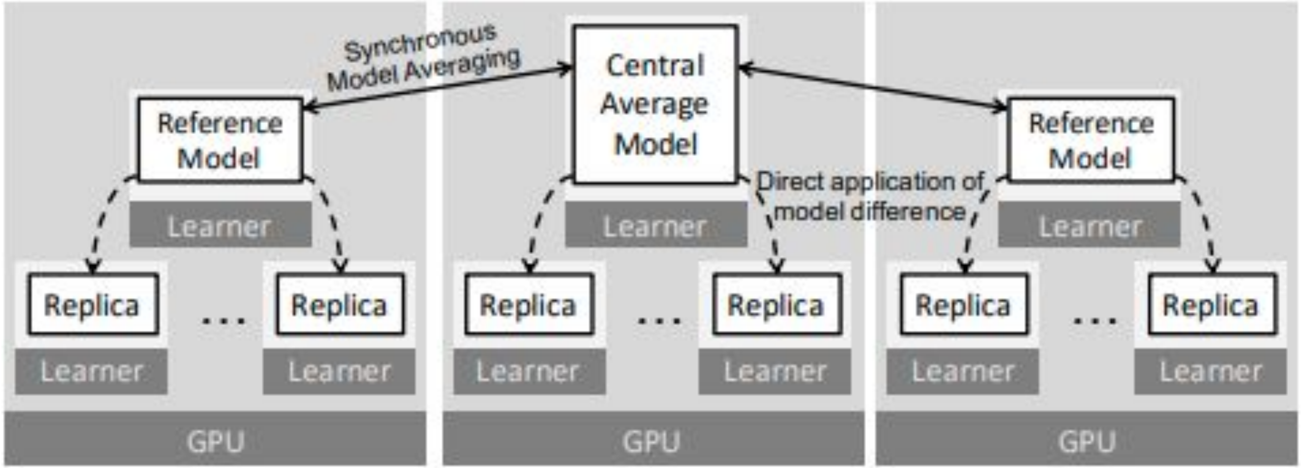
// Update central average model

$$\mathrm{z}' \leftarrow \mathrm{z} \,;$$

$$\mathrm{z} \leftarrow \mathrm{z} + \sum_{j=1}^{k} c_j + \mu(\mathrm{z} - \mathrm{z}_{prev}) \;\;;$$

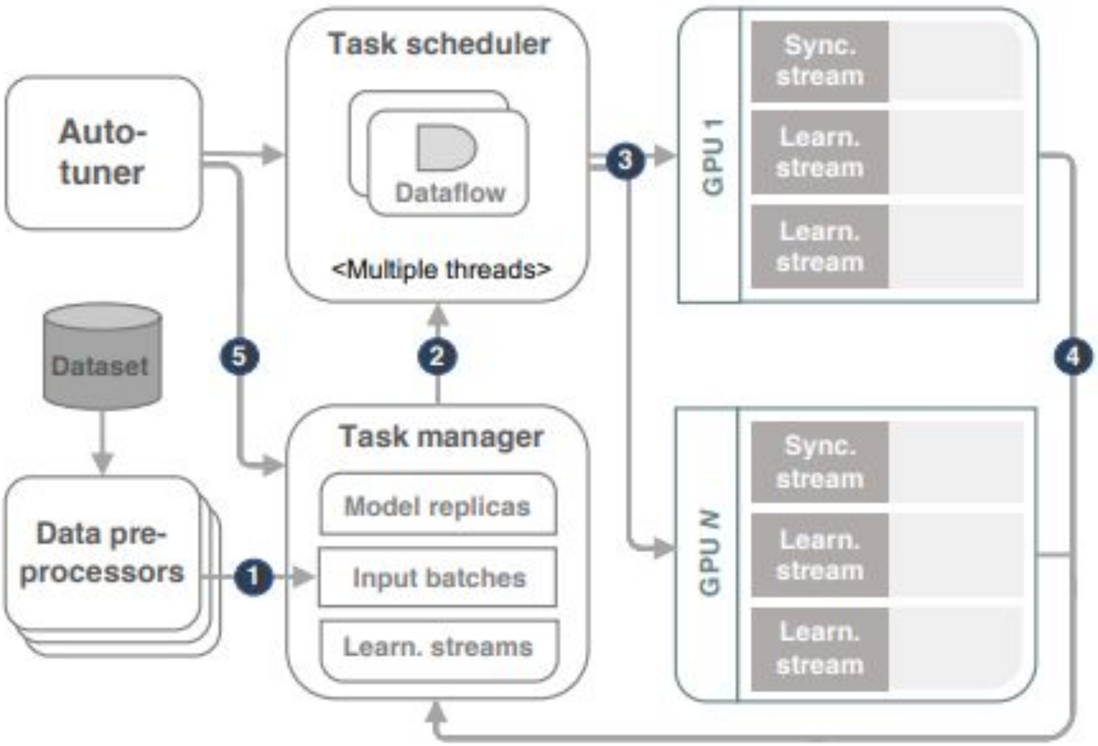
$$\mathrm{z}_{prev} \leftarrow \mathrm{z}';$$

Synchronizing multiple learners per GPU



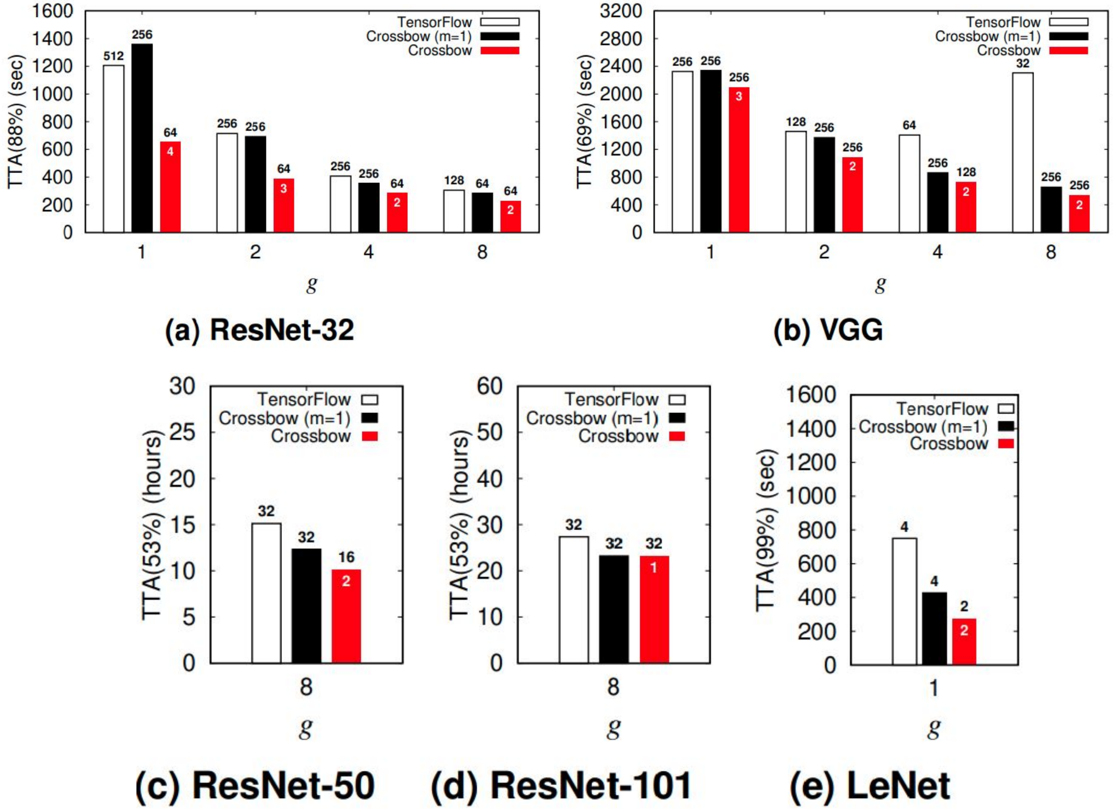
(CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers)

Architecture to maximize HW efficiency



(CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers)

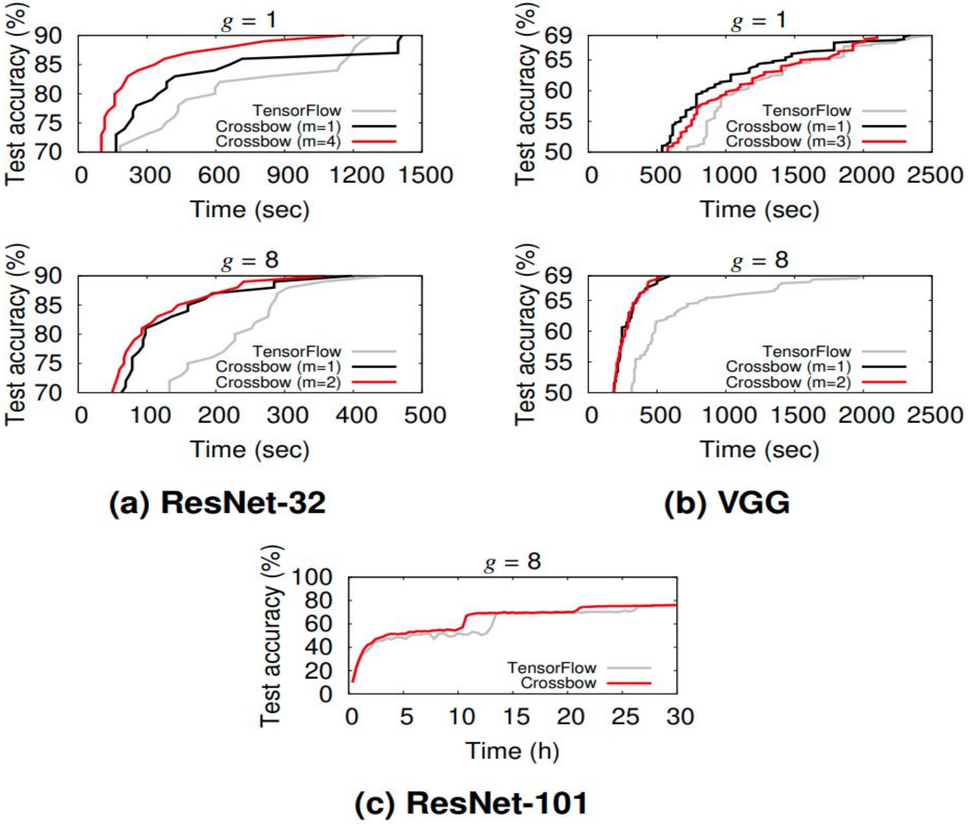
Result: Time to Accuracy



g: number of GPU
m: number of learner per GPU

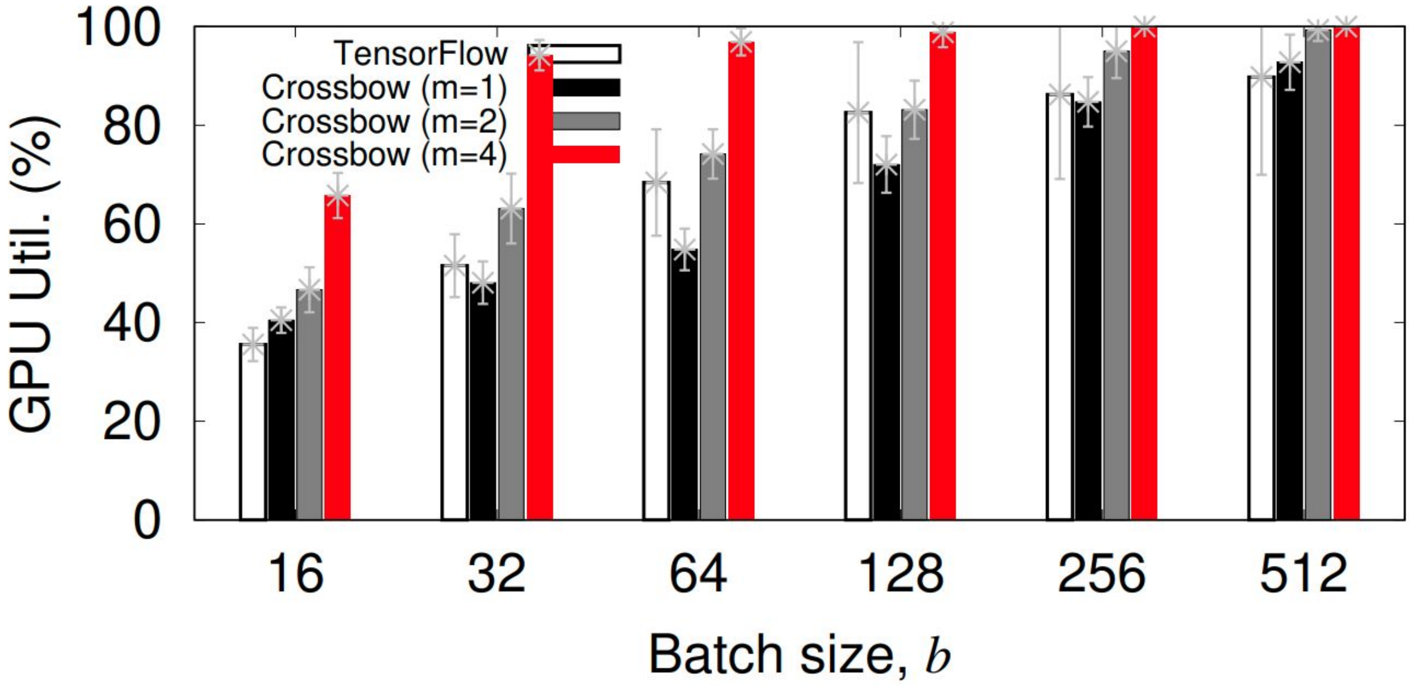
(CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers)

Result: Convergence Over Time



(CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers)

Result: GPU utilisation for various batch sizes



(CROSSBOW: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers)

Don’t Use Large Mini-Batches, Use Local SGD

What is local SGD?

- Mini-batch SGD

$$\boldsymbol{w}_{(t+1)} := \boldsymbol{w}_{(t)} - \gamma_{(t)} \Big[\frac{1}{K} \sum_{k=1}^{K} \ \frac{1}{B} \sum_{i \in \mathcal{I}_{(t)}^k} \nabla f_i \big(\boldsymbol{w}_{(t)} \big) \Big]$$

- **Local SGD**: each worker *k* evolves a **local model** by performing **H** SGD **updates** with mini-batch size B_{loc}, **before communication** among the workers.

$$\boldsymbol{w}_{(t)+h+1}^k := \boldsymbol{w}_{(t)+h}^k - \gamma_{(t)} \Big[\frac{1}{B_{\text{loc}}} \sum_{i \in \mathcal{I}_{(t)+h}^k} \nabla f_i \big(\boldsymbol{w}_{(t)+h}^k \big) \Big]$$

Data parallelisation dichotomy

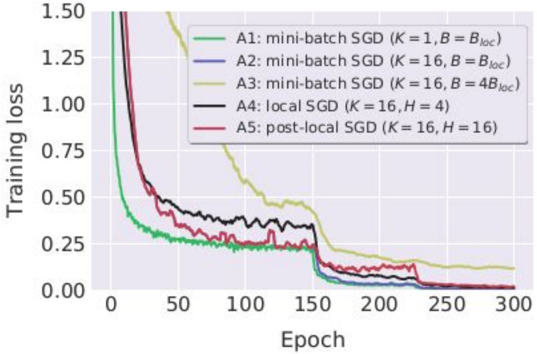
- **Scenario 1**. The communication restricted setting (Communication efficiency)
- **Scenario 2**. The regime of poor generalization of large-batch SGD (out-of-sample error)

Post-local SGD

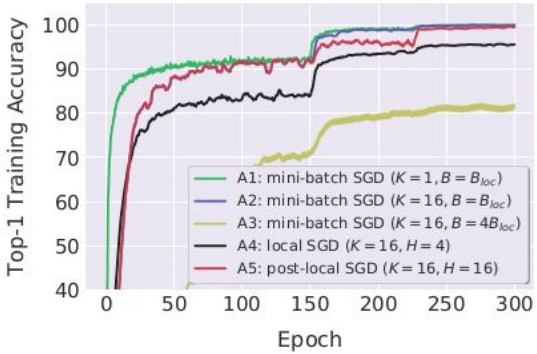
- Local SGD is **only started** in the second phase of training **after *t'* initial steps** (1st learning rate decay) with standard mini-batch SGD.

$$H_{(t)} = \begin{cases} 1, & \text{if } t \leqslant t', \qquad \textit{(mini-batch SGD)} \\ H, & \text{if } t > t'. \qquad \textit{(local SGD)} \end{cases}$$

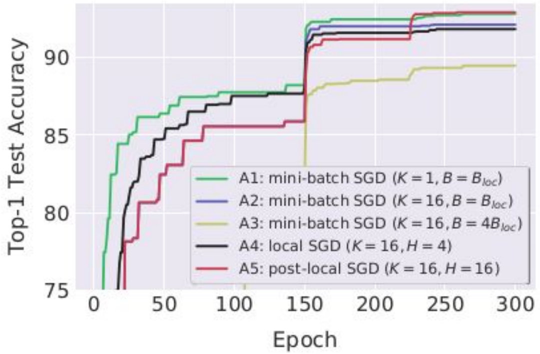
Main results



(a) Training loss.



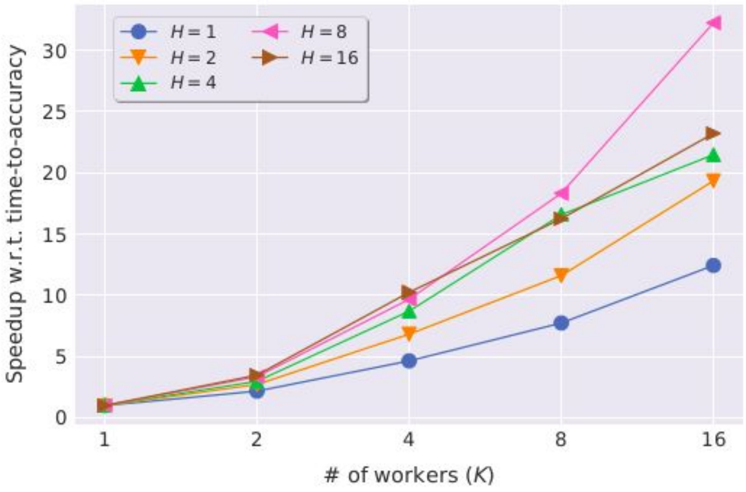
(b) Training top-1 accuracy.



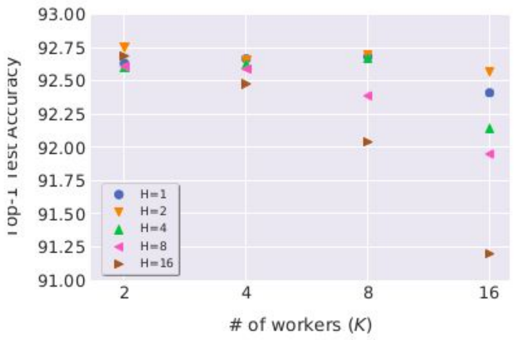
(c) Test top-1 accuracy.

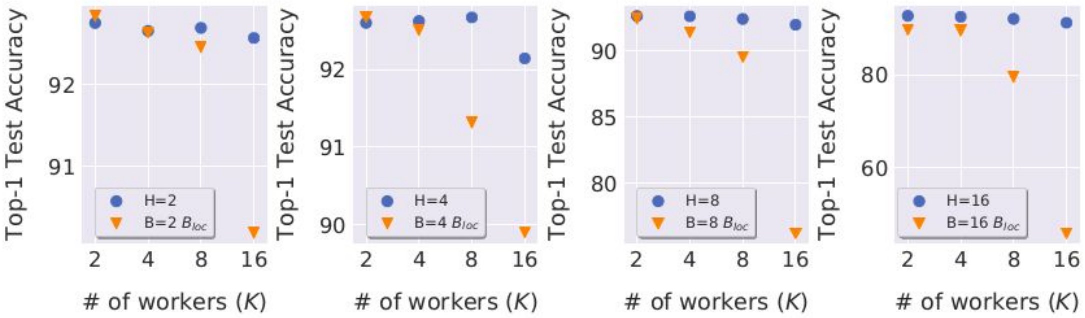
	accuracy on training set			accuracy on test set		system performance		
Algorithm	loss value	top-1 acc.		top-1 acc.		parallelism	communication	
A1: small mini-batch SGD ($K=1$, $B=B_{\text{loc}}$)	0.01	100%	*excellent*	93%	*excellent*	×1	-	*poor*
A2: large mini-batch SGD ($K=16$, $B=B_{\text{loc}}$)	0.01	100%	*excellent*	92%	*good*	×16	÷1	*ok*
A3: huge mini-batch SGD ($K=16$, $B=4B_{\text{loc}}$)	0.10	81%	*poor*	89%	*poor*	×16	÷4	*good*
A4: local SGD ($K=16$, $H=4$)	0.01	95%	*ok*	92%	*good*	×16	÷4	*good*
A5: post-local SGD ($K=16$, $H=16$)	0.01	99%	*excellent*	93%	*excellent*	×16	÷1 (phase 1), ÷16 (phase 2)	*good*

Superior scalability of Local SGD over mini-batch SGD

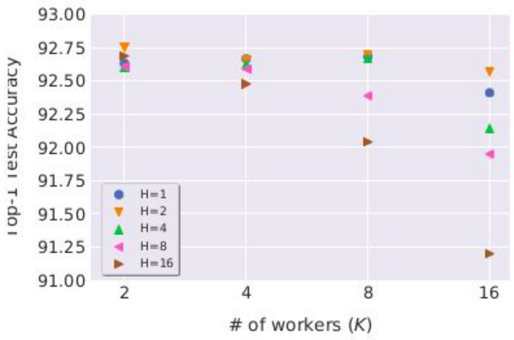


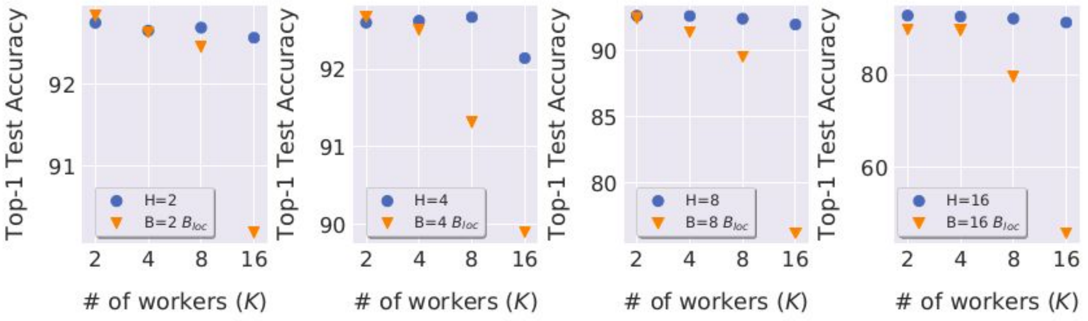
Local SGD outperforms mini-batch SGD at the same effective batch size and communication ratio





Local SGD outperforms mini-batch SGD at the same effective batch size and communication ratio

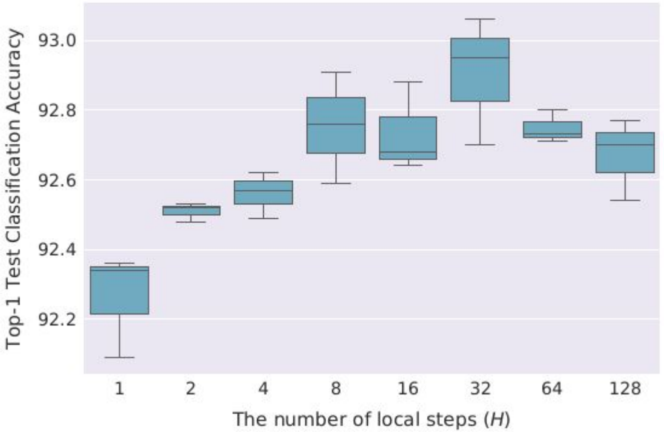


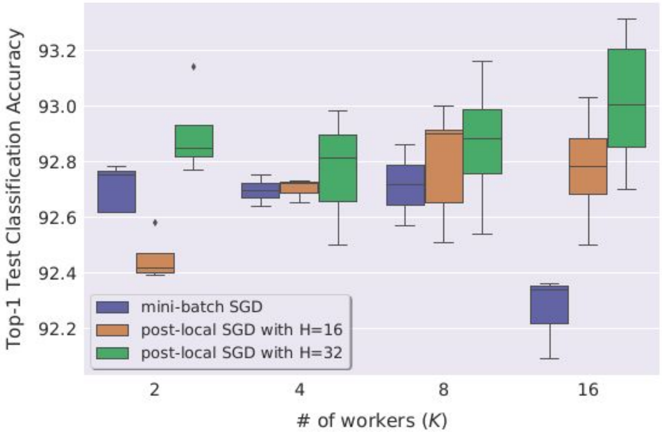


Local SGD still encounters difficulties when scaling to very large mini-batches

Post-local SGD closes the generalization gap of large-batch training

Constant mini-batch size of 2048 or 4096





	CIFAR-10				CIFAR-100			
	Small batch baseline ⋆ $K=2, B=128$	Large batch baseline ⋆ $K=16, B=128$	Post-local SGD (H=16) $K=16, B_{loc}=128$	Post-local SGD (H=32) $K=16, B_{loc}=128$	Small batch baseline ⋆ $K=2, B=128$	Large batch baseline ⋆ $K=16, B=128$	Post-local SGD (H=16) $K=16, B_{loc}=128$	Post-local SGD (H=32) $K=16, B_{loc}=128$
ResNet-20	92.63 ±0.26	92.48 ±0.17	92.80 ±0.16	**93.02** ±0.24	68.84 ±0.06	68.17 ±0.18	69.24 ±0.26	**69.38** ±0.20
DenseNet-40-12	94.41 ±0.14	94.36 ±0.20	94.43 ±0.12	**94.58** ±0.11	74.85 ±0.14	74.08 ±0.46	74.45 ±0.30	**75.03** ±0.05
WideResNet-28-10	95.89 ±0.10	95.43 ±0.37	**95.94** ±0.06	95.76 ±0.25	79.78 ±0.16	79.31 ±0.23	80.28 ±0.13	**80.65** ±0.16

Main contributions

- **Local SGD** can serve as a **communication-efficient** alternative to mini-batch SGD
- **Post-local SGD** provides a state-of-the-art **remedy for the generalization issue of large-batch trainings**