DAY 4: DISTRIBUTED TRAINING ON LARGE DATA

Combating Accuracy Loss

2021-02-04 | Jenia Jitsev | Cross Sectional Team Deep Learning, Helmholtz AI @ JSC

JÜLICH Forschungszentrum

DISTRIBUTED TRAINING ON LARGE DATA

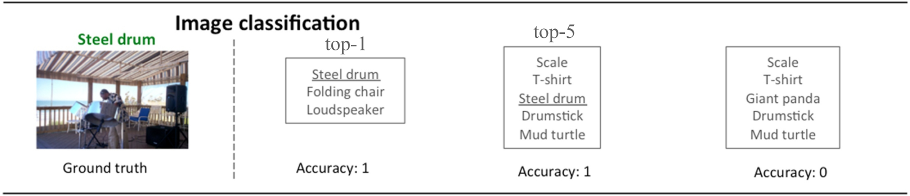
- ImageNet-1k : still gold standard in training large visual recognition models
- Serves as “Hello World” for large dataset training

MNIST, CIFAR-10/100
28x28, 32x32; 60k examples

ImageNet-1k, 21k; OpenImages, FFHQ…
224x224, 1024x1024; 1.2M examples

DISTRIBUTED TRAINING ON LARGE DATA

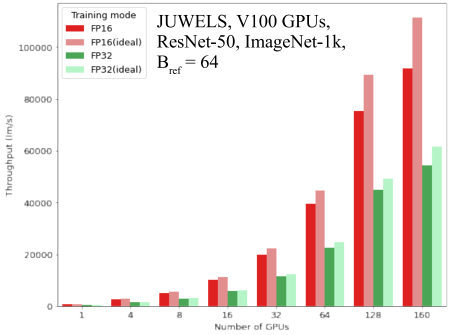
- ImageNet-1k : still gold standard in training large visual recognition models
- ResNet-50 : baseline model network, test accuracies : $\approx 75\%$ top-1, $\approx 94\%$ top-5 (Winner ILSVRC 2015)

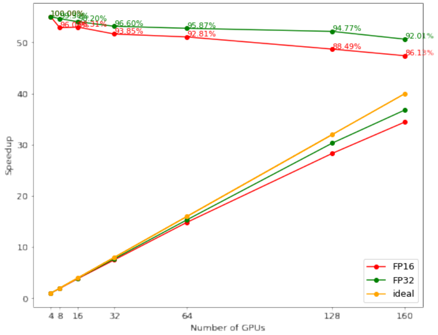


Russakovsky et al, IJCV, 2015

DISTRIBUTED TRAINING ON LARGE DATA

- ResNet-50 : efficient distributed training in data parallel mode possible
  - prerequisite is good scaling of throughput during training
  - image throughput during training ideally increasing as $\tau_K = K \cdot \tau_{ref}$ Images/sec
  - training with a large effective batch size $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$, K workers



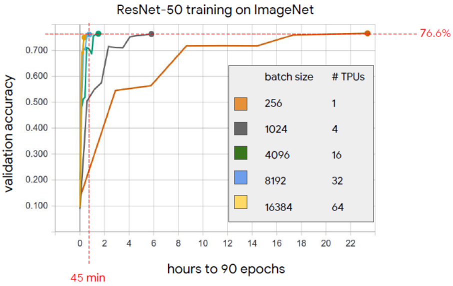


Cherti et al, 2020

DISTRIBUTED TRAINING ON LARGE DATA

- ResNet-50 : efficient distributed training in data parallel mode
  - High test accuracy in the end of the training is the goal

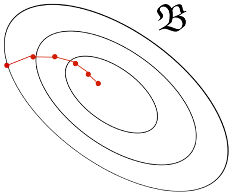
	Batch Size	Processor	DL Library	Time	Accuracy
He et al. [1]	256	Tesla P100 × 8	Caffe	29 hours	75.3 %
Goyal et al. [2]	8,192	Tesla P100 × 256	Caffe2	1 hour	76.3 %
Smith et al. [3]	8,192 → 16,384	full TPU Pod	TensorFlow	30 mins	76.1 %
Akiba et al. [4]	32,768	Tesla P100 × 1,024	Chainer	15 mins	74.9 %
Jia et al. [5]	65,536	Tesla P40 × 2,048	TensorFlow	6.6 mins	75.8 %
Ying et al. [6]	65,536	TPU v3 × 1,024	TensorFlow	1.8 mins	75.2 %
Mikami et al. [7]	55,296	Tesla V100 × 3,456	NNL	2.0 mins	75.29 %
This work	**81,920**	**Tesla V100 × 2,048**	**MXNet**	**1.2 mins**	**75.08%**



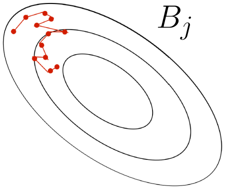
Yamazoto et al, 2019; Ying, 2018

DISTRIBUTED TRAINING ON LARGE DATA

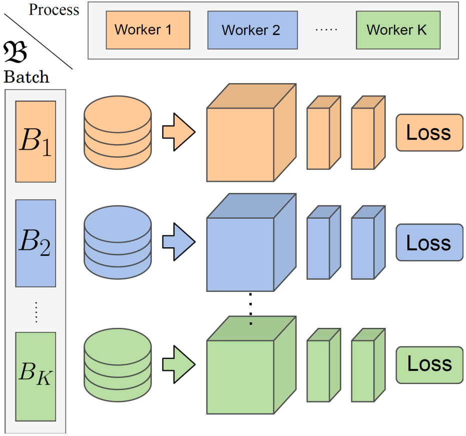
- Data parallel training: working with large effective batch sizes
- Reminder: Training with $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$, K workers
- Large effective batch sizes alter model optimization trajectory



Effective larger batch, over all K workers

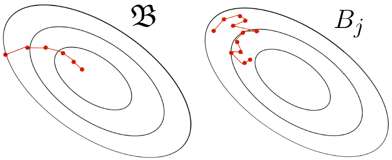


Reference small batch, per single worker



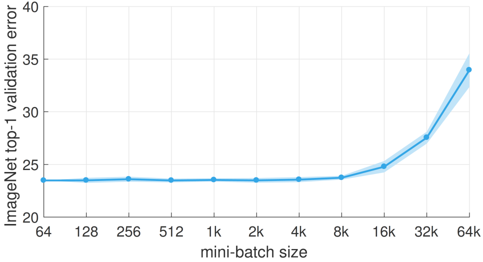
DISTRIBUTED TRAINING ON LARGE DATA

- Data parallel training: working with large effective batch sizes
- Training with $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$, K workers
- Large effective batch sizes alter model optimization trajectory
  - may require hyperparameter re-tuning compared to a working smaller batch (single node) version



Effective larger batch, over all K workes

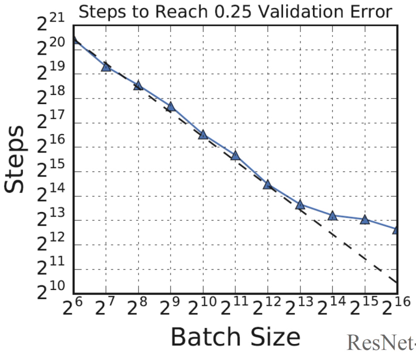
Reference small batch, per single worker

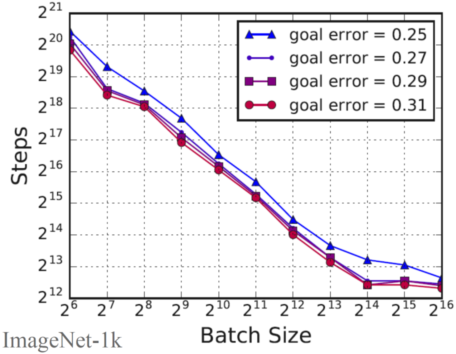


Goyal et al, 2017

DISTRIBUTED TRAINING ON LARGE DATA

- ResNet-50 : efficient distributed training in data parallel mode
  - for very large batch sizes $|\mathfrak{B}|$: diminishing speed-up returns when training towards a given test accuracy



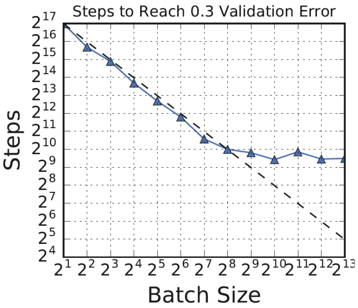


ResNet-50, ImageNet-1k

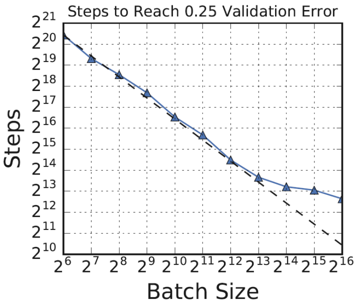
Shallue et al, JMLR, 2019

DISTRIBUTED TRAINING ON LARGE DATA

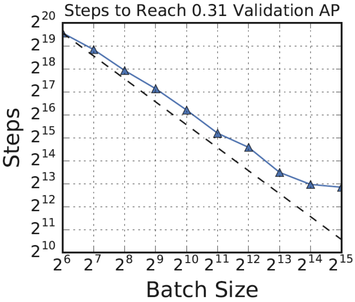
- Critical large batch sizes $|\mathfrak{B}_{\text{crit}}|$: diminishing speed-up when crossing, given target test accuracy



ResNet-8, CIFAR-10



ResNet-50, ImageNet-1k

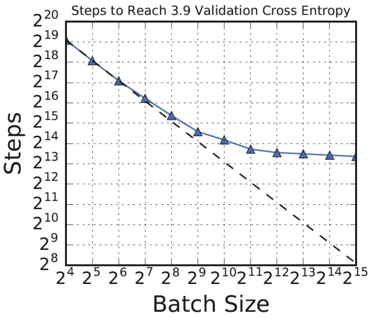


ResNet-50, OpenImages

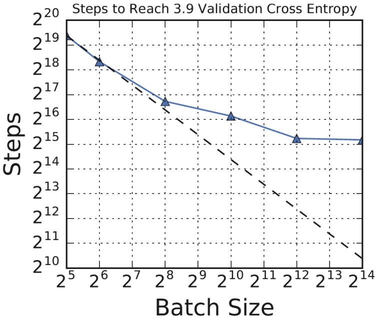
Shallue et al, JMLR, 2019

DISTRIBUTED TRAINING ON LARGE DATA

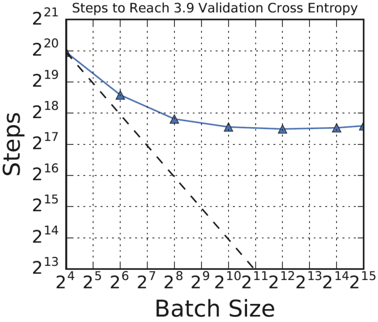
- Critical large batch sizes $|\mathfrak{B}_{\text{crit}}|$: systematic evidence across datasets, tasks and architectures



Transformer, LM1B



Transformer, Common Crawl

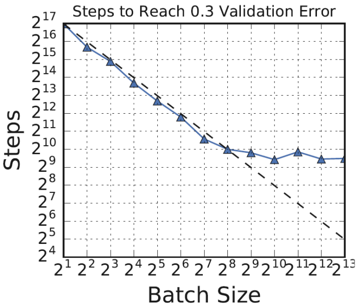


LSTM, LM1B

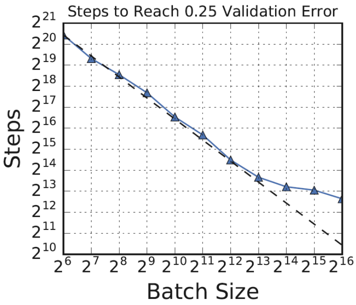
Shallue et al, JMLR, 2019

DISTRIBUTED TRAINING ON LARGE DATA

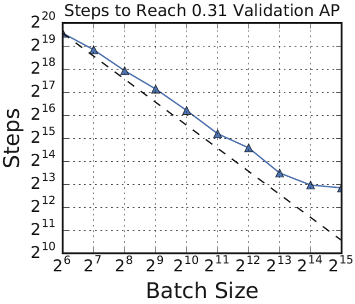
- Critical large batch sizes $|\mathfrak{B}_{\text{crit}}|$: large enough to do efficient distributed training
- Efficient Distributed Training with $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$, for large K
  - providing almost linear training speed up, $t_{\mathfrak{B}} = \frac{1}{K} t_B$



ResNet-8, CIFAR-10



ResNet-50, ImageNet-1k

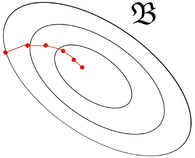


ResNet-50, OpenImages

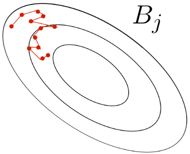
Shallue et al, JMLR, 2019

DISTRIBUTED TRAINING ON LARGE DATA

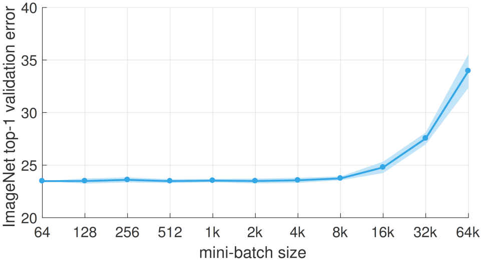
- Efficient Distributed Training with $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$, for large K
- Providing almost linear training speed up, $t_{\mathfrak{B}} = \frac{1}{K} t_B$, **without loss of test accuracy**
- Important: reducing training **time to accuracy** - **time to solution**
  - strong scaling : reducing **time to accuracy**
  - reducing time per update step, per epoch, increasing samples throughput - alone not **sufficient** for speeding-up, reducing **time to accuracy**!
  - doing “bad” update steps during training would require doing a lot of them before reaching target loss/accuracy . . .



Effective larger batch, over all K workes



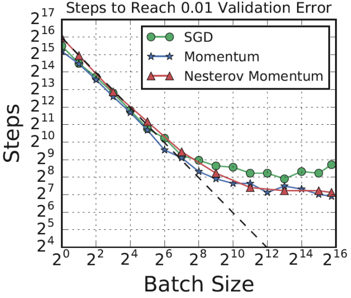
Reference small batch, per single worker



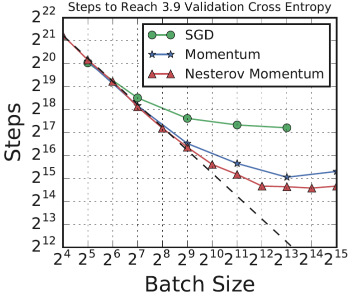
Goyal et al, 2017

DISTRIBUTED TRAINING ON LARGE DATA

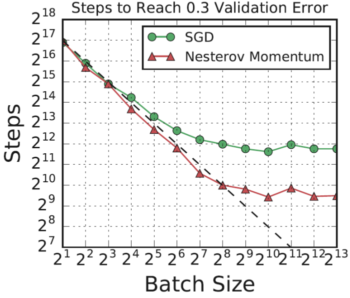
- Efficient Distributed Training with $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$, for large K
- Still debated whether hyperparameters tuning may allow for even larger batch sizes while still reducing time to accuracy



(a) Simple CNN on MNIST



(b) Transformer on LM1B

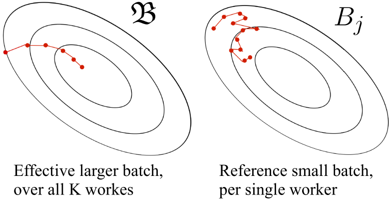


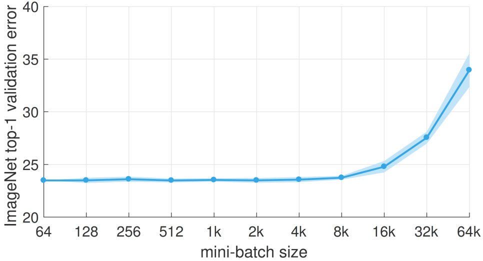
(c) ResNet-8 on CIFAR-10

Shallue et al, JMLR, 2019

DISTRIBUTED TRAINING ON IMAGENET

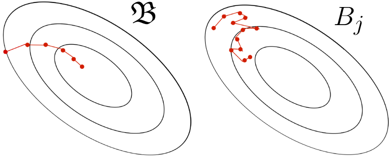
- Efficient Distributed Training with $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$, for large K
- Combating accuracy loss when using larger batch sizes: hyperparameter tuning
- Reducing **time to accuracy** with **target accuracy** equal to a working smaller batch (single node) reference





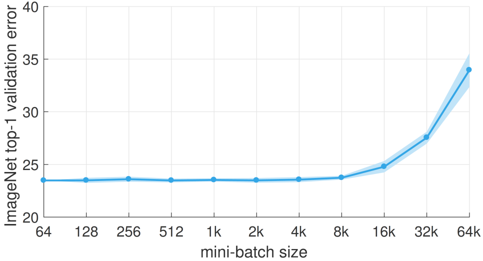
DISTRIBUTED TRAINING ON IMAGENET

- Combating accuracy loss when using larger batch sizes: hyperparameter tuning
- Learning rate rescaling with respect to $|\mathfrak{B}|$ and $|B_{\text{ref}}|$



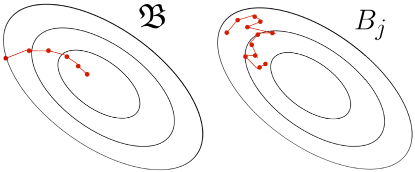
Effective larger batch, over all K workes

Reference small batch, per single worker



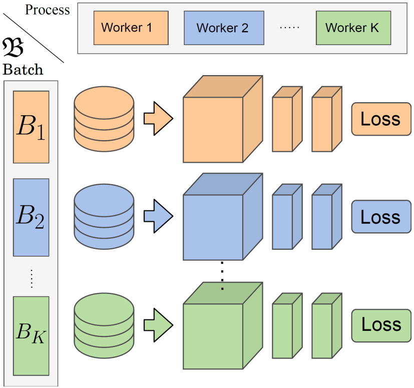
COMBATING ACCURACY LOSS ON IMAGENET

- Learning rate rescaling: motivation to match weight updates for different batch sizes $|\mathfrak{B}|$, $|B_{\text{ref}}|$, $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$



Effective larger batch, over all K workers

Reference small batch, per single worker



COMBATING ACCURACY LOSS ON IMAGENET

- Learning rate rescaling: motivation to match weight updates for different batch sizes, $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$
  - increase the weight update step size to accommodate for the fewer number of update steps when having a larger batch size

K update steps of SGD with learning rate η and $|B_{\text{ref}}| = n$:

$$\mathbf{W}_{t+K} = \mathbf{W}_t - \underbrace{\eta \frac{1}{n}} \sum_{j<K} \sum_{X \in B_j} \nabla \underbrace{\mathcal{L}(X, \mathbf{W}_{t+j})}$$

Single update step with $|\mathfrak{B}| = Kn$, learning rate $\hat{\eta}$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \underbrace{\hat{\eta} \frac{1}{Kn}} \sum_{j<K} \sum_{X \in B_j} \nabla \underbrace{\mathcal{L}(X, \mathbf{W}_t)}$$

Goyal et al, 2017

COMBATING ACCURACY LOSS ON IMAGENET

- Learning rate: linear rescaling, $\hat{\eta} = K\eta$, for $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$

To get $\mathbf{W}_{t+1} \approx \mathbf{W}_{t+K}$,

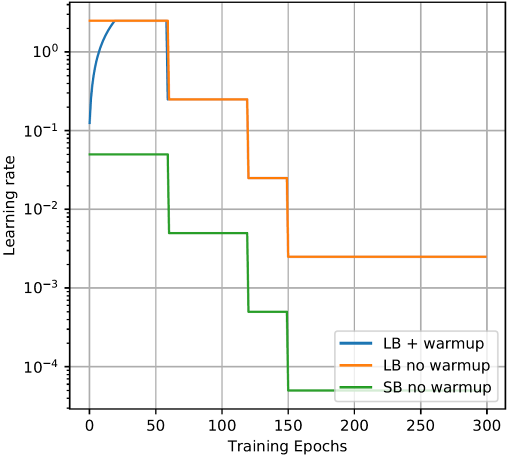
we assume $\nabla\mathcal{L}(X, \mathbf{W}_t) \approx \nabla\mathcal{L}(X, \mathbf{W}_{t+j})$ for $j < K$
and obtain

$$\hat{\eta}\frac{1}{kn} = \eta\frac{1}{n} \Leftrightarrow \hat{\eta} = \frac{kn}{n}\eta \Leftrightarrow \hat{\eta} = K\eta$$

Goyal et al, 2017

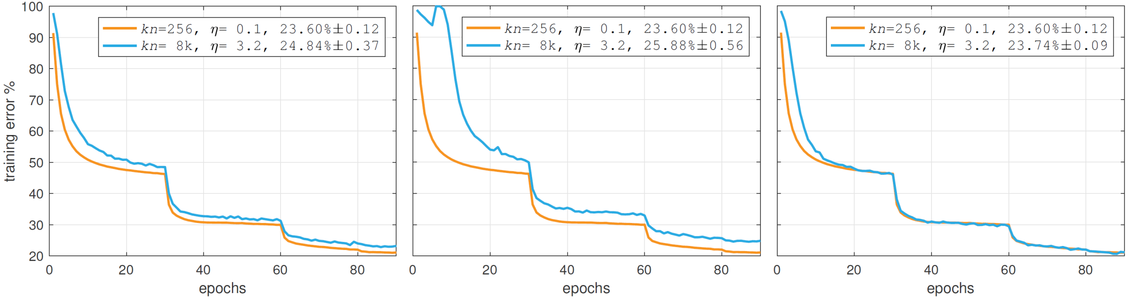
COMBATING ACCURACY LOSS ON IMAGENET

- Learning rate: linear rescaling, $\hat{\eta} = K\eta$, for $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$
  - used in combination with usual learning rate schedules
- $\nabla\mathcal{L}(X, \mathbf{W}_t) \approx \nabla\mathcal{L}(X, \mathbf{W}_{t+j})$ for $j < K$ does not hold in general
  - especially wrong for initial learning phase where gradients vary a lot from step to step
  - A possible remedy: initial warm-up phase



COMBATING ACCURACY LOSS ON IMAGENET

- Learning rate: linear rescaling, $\hat{\eta} = K\eta$, for $|\mathfrak{B}| = K \cdot |B_{\text{ref}}|$
  - used in combination with usual learning rate schedules
- $\nabla \mathcal{L}(X, \mathbf{W}_t) \approx \nabla \mathcal{L}(X, \mathbf{W}_{t+j})$ for $j < K$ is bad assumption for early learning
- Warm-up phase: start with η, increase towards scaled $\hat{\eta} = K\eta$ within few epochs

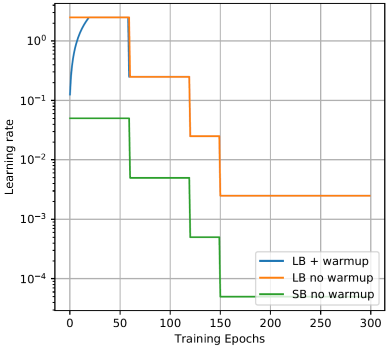


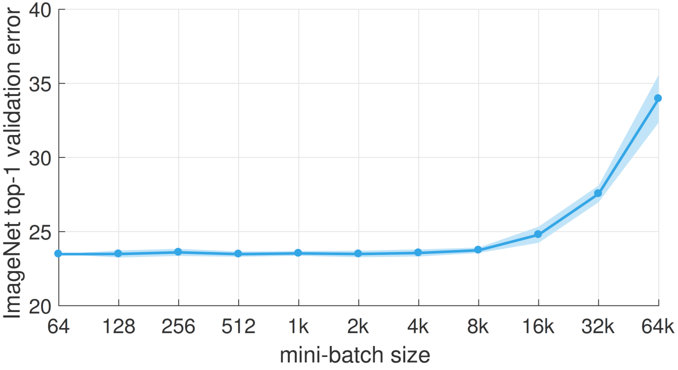
(a) no warmup (b) constant warmup (c) gradual warmup

Goyal et al, 2017

COMBATING ACCURACY LOSS ON IMAGENET

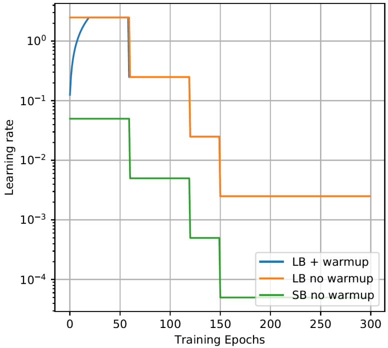
- Learning rate tuning: package of mechanisms
  - linear rescaling
  - Warm-up for initial epochs
  - Schedules

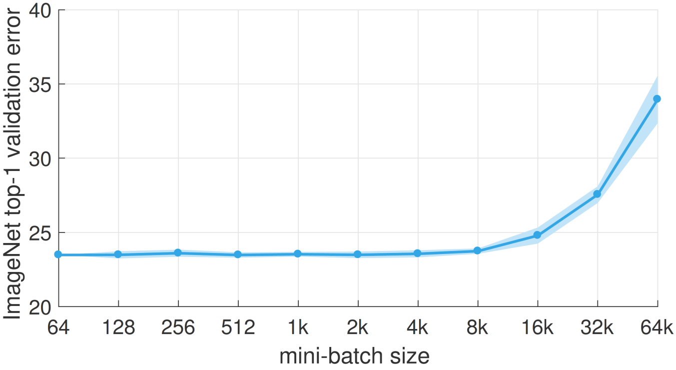




COMBATING ACCURACY LOSS ON IMAGENET

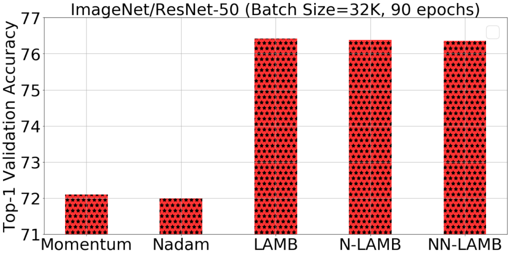
- Learning rate tuning: package of mechanisms
- Often, still not enough for very large batch sizes $|\mathfrak{B}| > 8192$
- Advanced Optimizers that provide further adaptive hyperparamer tuning during training





COMBATING ACCURACY LOSS ON IMAGENET

- Advanced optimizers that provide further adaptive hyperparamer tuning during training: very large batch sizes $|\mathfrak{B}| > 8192$
- LARS : Layer-wise Adaptive Rate Scaling, extension of SGD with momentum
  - tuning learning rates layerwise depending on gradient and weight amplitudes and norms
- LAMB : Layer Adaptive Moment Batch, extension of LARS (use AdamW as base)
  - tuning learning rate layerwise, also per weight parameter using gradient mean and variance



You et al, ICLR, 2020

COMBATING ACCURACY LOSS ON IMAGENET

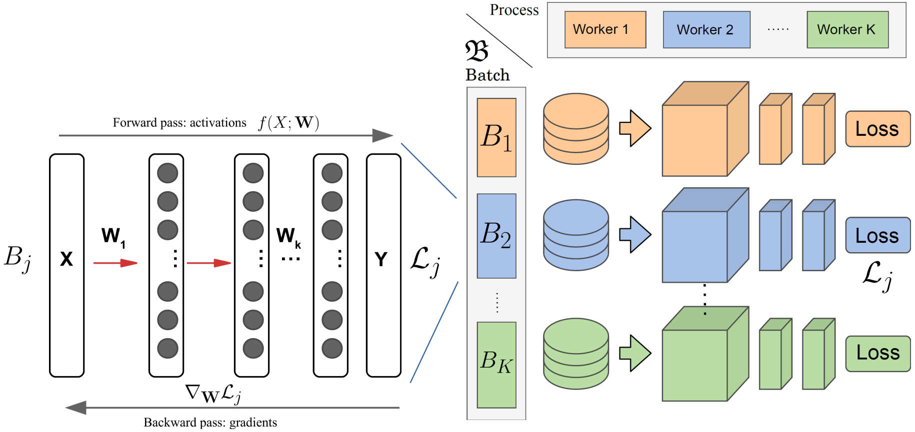
- Learning rate rescaling, schedules and Warm up : works well for $|\mathfrak{B}| \leq 8192$)
- Advanced optimizers (LAMB) : works for $|\mathfrak{B}| \leq 80k$)
- Almost linear speed-up in training time without accuracy loss: reducing **time to accuracy**

	Hardware	Software	Batch size	Optimizer	# Steps	Time/step	Time	Accuracy
Goyal *et al.* [6]	Tesla P100 × 256	Caffe2	8,192	SGD	14,076	0.255 s	1 hr	**76.3** %
You *et al.* [8]	KNL × 2048	Intel Caffe	32,768	SGD	3,519	0.341 s	20 min	75.4 %
Akiba *et al.* [7]	Tesla P100 × 1024	Chainer	32,768	RMSprop/SGD	3,519	0.255 s	15 min	74.9 %
You *et al.* [8]	KNL × 2048	Intel Caffe	32,768	SGD	2,503	0.335 s	14 min	74.9 %
Jia *et al.* [9]	Tesla P40 × 2048	TensorFlow	65,536	SGD	1,800	0.220 s	6.6 min	75.8 %
Ying *et al.* [13]	TPU v3 × 1024	TensorFlow	32,768	SGD	3,519	**0.037 s**	2.2 min	**76.3** %
Mikami *et al.* [10]	Tesla V100 × 3456	NNL	55,296	SGD	2,086	0.057 s	2.0 min	75.3 %
Yamazaki *et al.* [11]	Tesla V100 × 2048	MXNet	81,920	SGD	1,440	0.050 s	**1.2 min**	75.1 %

Osawa et al, 2020

DISTRIBUTED TRAINING WITH LARGE BATCHES

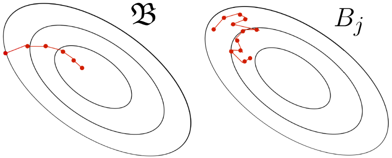
- More advanced techniques may allow efficient distributed training beyond batch size issues
- Local SGD: giving up consistency between model parameters across different workers after each update
- Post Local SGD: combining coupled global SGD and decoupled local SGD
- Natural SGD: attempt to use second derivatives and curvature information



DISTRIBUTED TRAINING WITH LARGE BATCHES

Summary

- Efficient data parallel training on large datasets like ImageNet-1k
- Measures to stabilize training with large batches necessary
- Learning rate scaling, schedules, warm-up phase, specialized optimizers
- Advanced methods required for very large $|\mathfrak{B}| \geq 32k$
- Aim: reduce **time to accuracy** without accuracy loss



Effective larger batch, over all K workes

Reference small batch, per single worker