DATA PARALLELISM: HOW TO TRAIN DEEP LEARNING MODELS ON MULTIPLE GPUS

LAB 1, PART 1: INTRODUCTION AND MOTIVATION

NVIDIA

DEEP
LEARNING
INSTITUTE

COURSE OVERVIEW

- Lab 1: Gradient Descent vs Stochastic Gradient Descent, and the Effects of Batch Size
- Lab 2: Multi-GPU DL Training Implementation using DistributedDataParallel (DDP)
- Lab 3: Algorithmic Concerns for Training at Scale

COURSE AGENDA

10:00-10:15 Introduction

10:15-11:15 Neural Network Training and Stochastic Gradient Descent

11:15-11:30 ***Coffee Break***

11:30-12:30 Neural Network Training and Intro to Parallel Training

12:30-13:30 ***Lunch Break***

13:30-15:00 Data Parallelism using Pytorch DDP

15:00-15:15 ***Coffee break***

15:15-16:45 Challenges of Data Parallel using Multiple GPUs

16:45-17:00 Q&A, Final Remarks

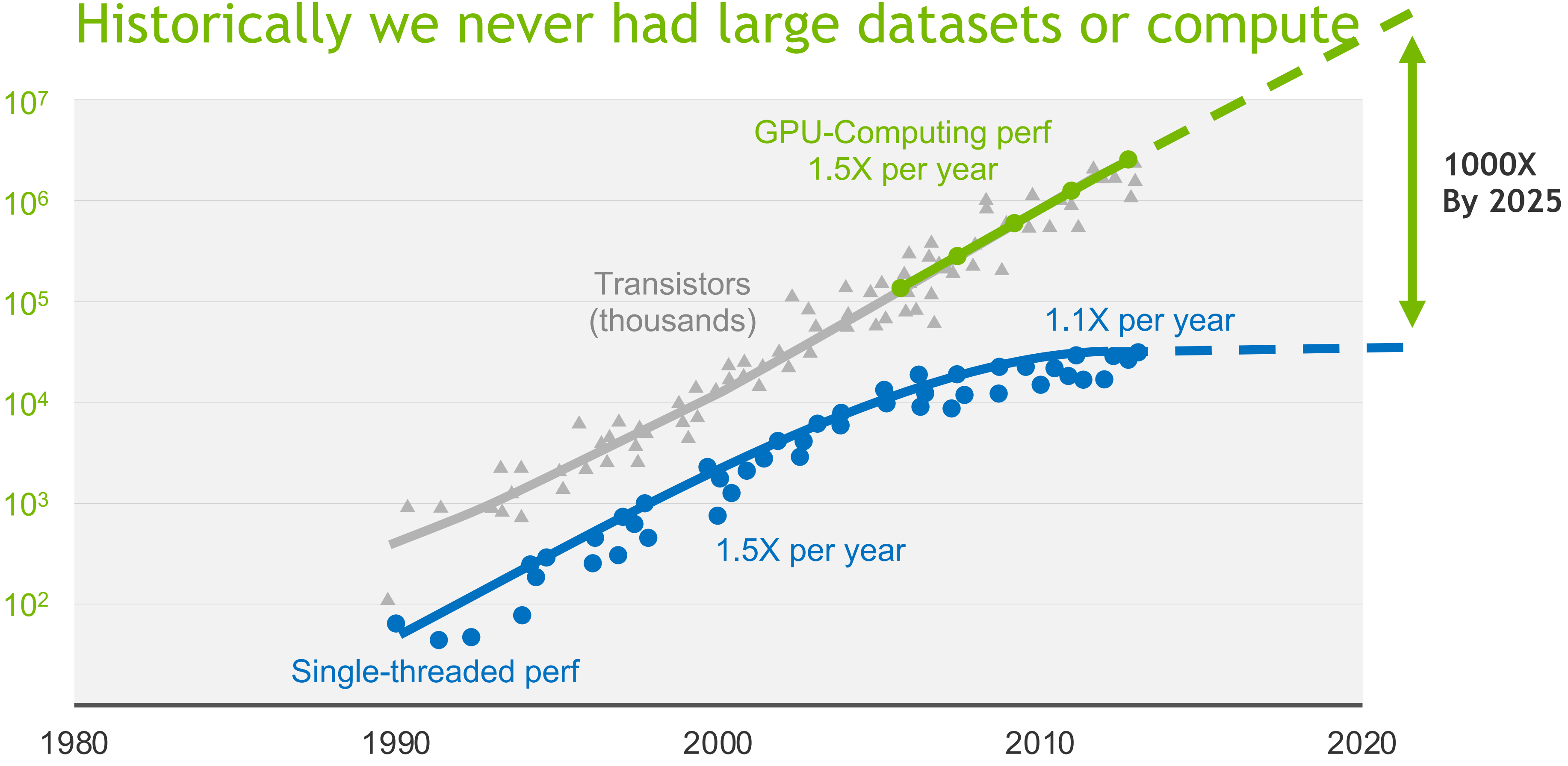
LAB 1 OVERVIEW

- Part 1: Gradient Descent
- Part 2: Stochastic Gradient Descent
- Part 3: Optimizing training with batch size

CONTEXT: WHY USE MULTIPLE GPUS?

TRENDS IN COMPUTATIONAL POWER

Historically we never had large datasets or compute



NVIDIA. DEEP LEARNING INSTITUTE

TRENDS IN COMPUTATIONAL POWER

2 PF/s in November 2009

TRENDS IN COMPUTATIONAL POWER

32 PF/s today

8x NVIDIA H100 GPUs With 640 Gigabytes of Total GPU Memory

- 18x NVIDIA NVLink connections per GPU
- 900 gigabytes per second of bidirectional GPU-to-GPU bandwidth
- 24 TB/s memory bandwidth

4x NVIDIA NVSwitches

- 7.2 terabytes per second of bidirectional GPU-to-GPU bandwidth

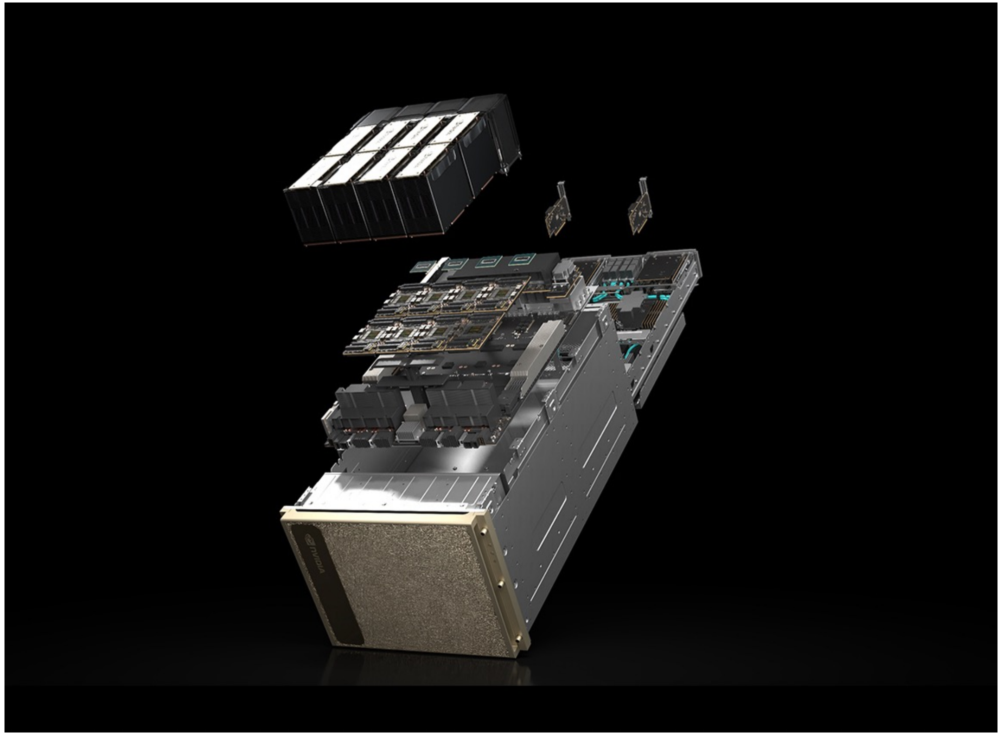
10x NVIDIA ConnectX-7 400 Gigabits-Per-Second Network Interface

- 1 terabyte per second of peak bidirectional network bandwidth

Dual x86 CPUs and 2 Terabytes of System Memory

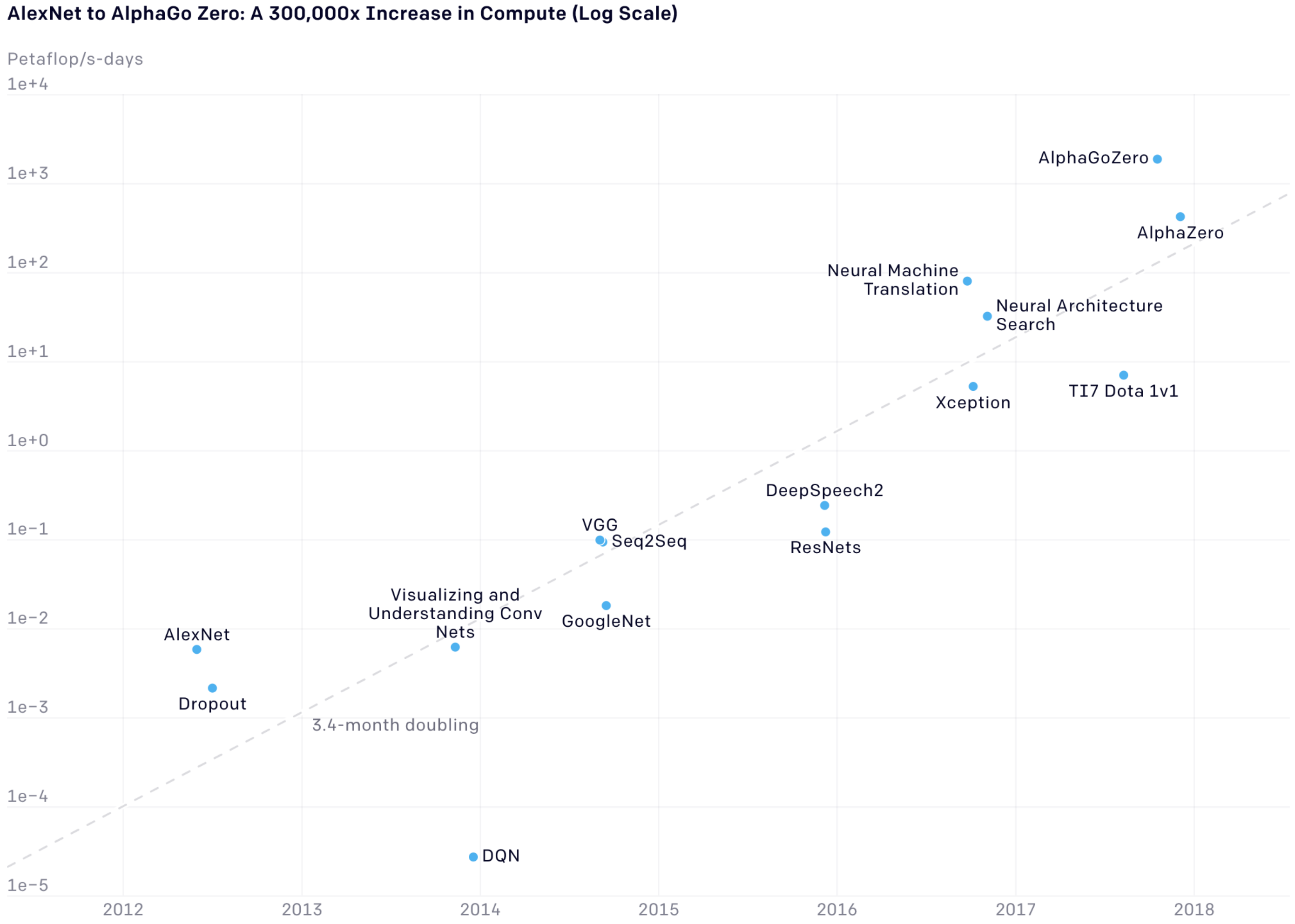
- Powerful CPUs and massive system memory for the most intensive AI jobs

32 petaFLOPS AI performance



NVIDIA DGX H100

NEURAL NETWORK COMPLEXITY IS EXPLODING



Source: OpenAI

1000 PETAFLOP/S-DAYS

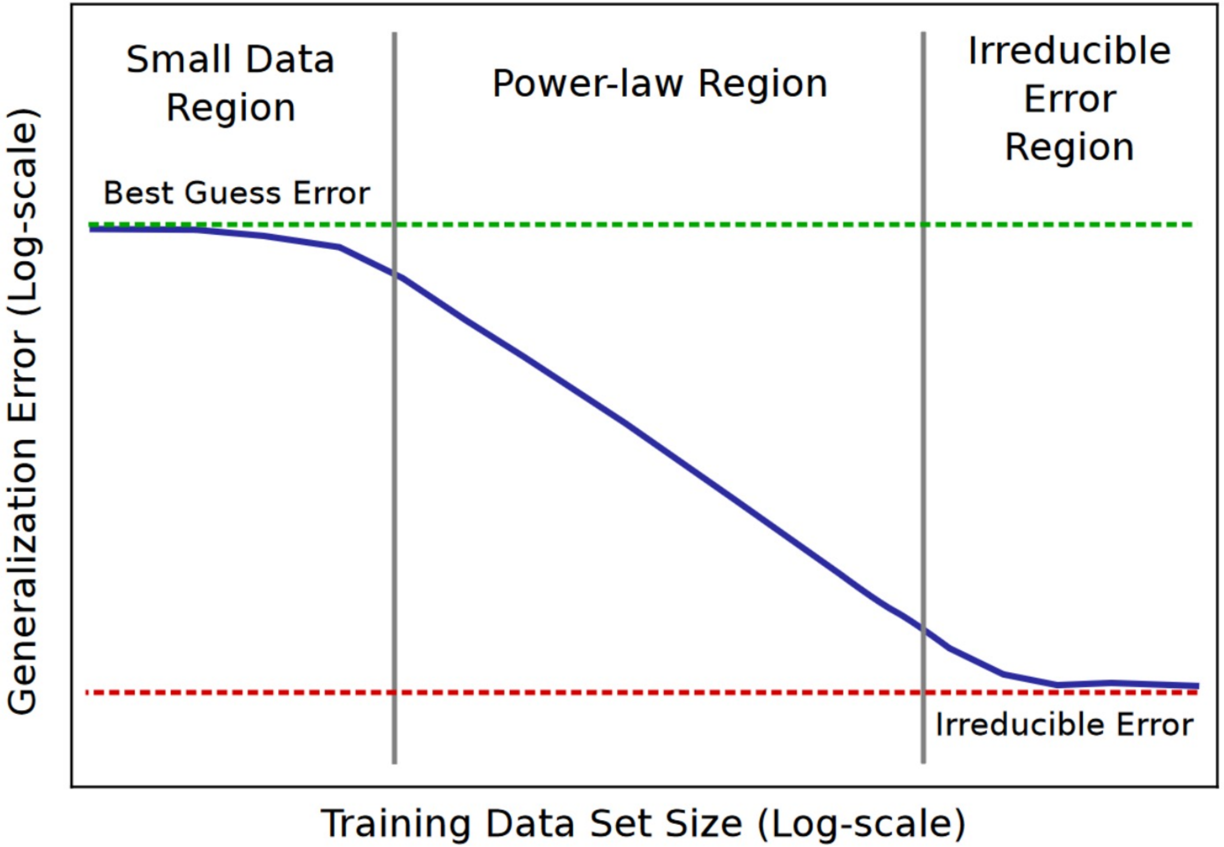
O(100 YEARS) ON A DUAL CPU SERVER

OR

O(30 DAYS) DGX H100

EXPLODING DATASETS

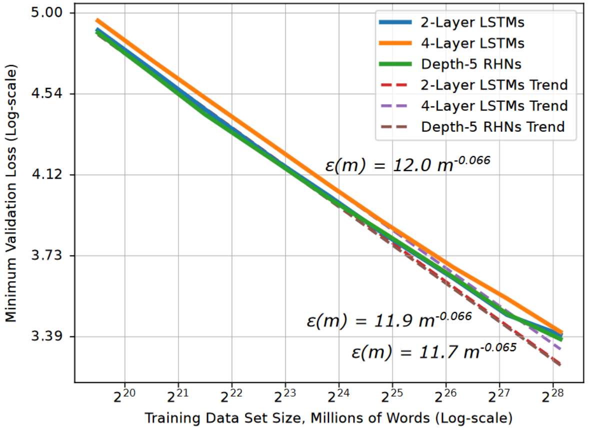
Power-law relationship between dataset size and accuracy



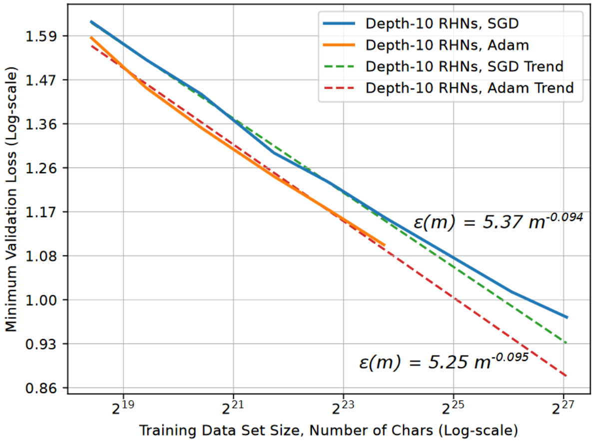
Hestness, J., et al. (2017). Deep Learning Scaling is Predictable, Empirically. arXiv: 1712.00409

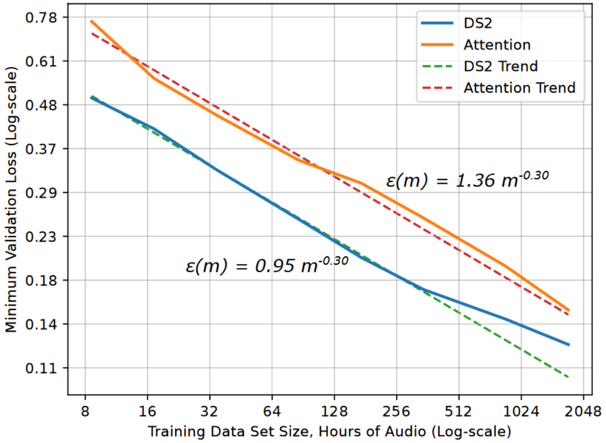
EXPLODING DATASETS

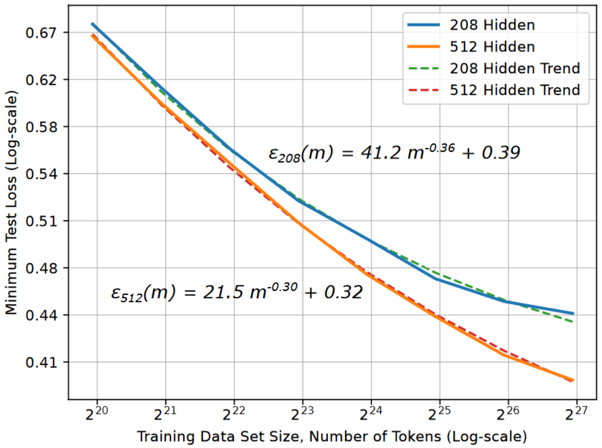
Power-law relationship between dataset size and accuracy

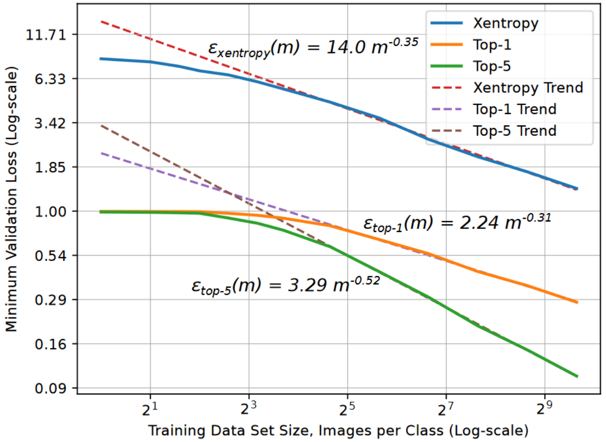


- Translation
- Language Models
- Character Language Models
- Image Classification
- Attention Speech Models





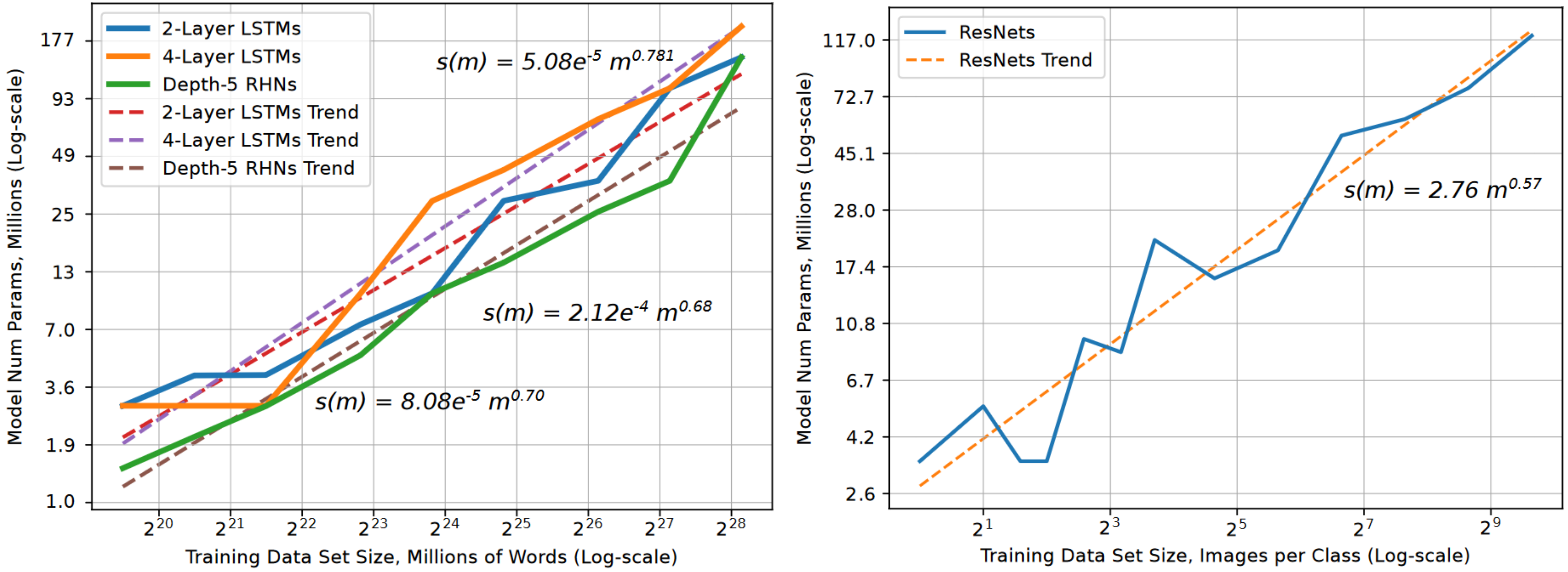




Hestness, J., et al. (2017). Deep Learning Scaling is Predictable, Empirically. arXiv: 1712.00409

EXPLODING MODEL COMPLEXITY

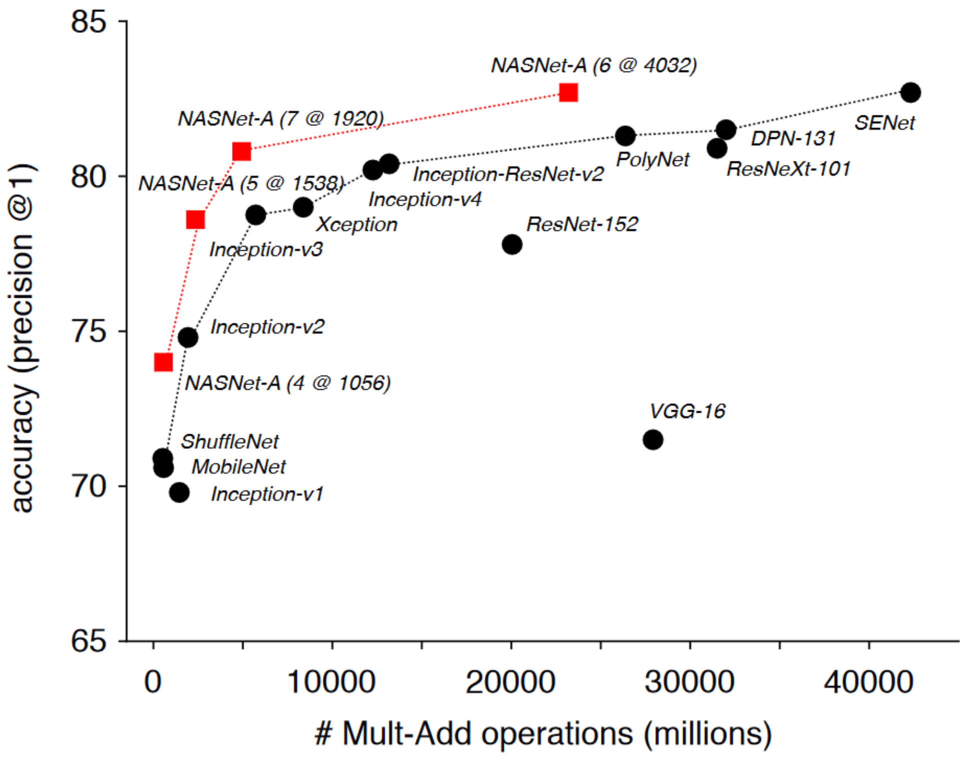
Though model size scales sublinearly

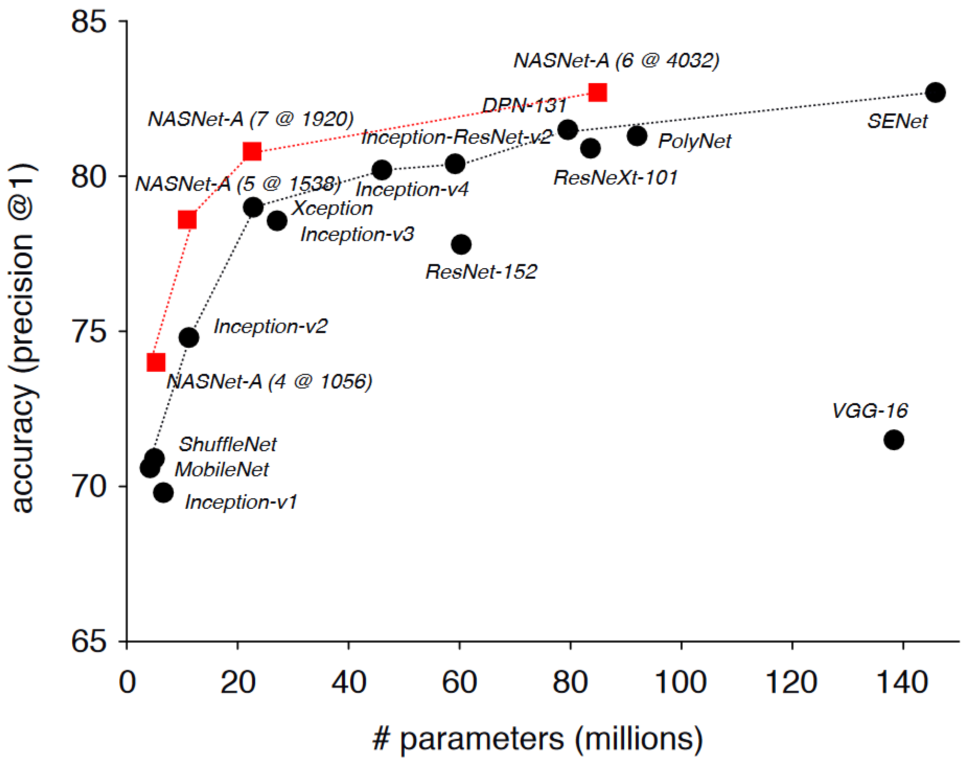


Hestness, J., et al. (2017). Deep Learning Scaling is Predictable, Empirically. [arXiv: 1712.00409]

EXPLODING MODEL COMPLEXITY

Though model size scales sublinearly





Zoph, Barret, et al. (2017). "Learning transferable architectures for scalable image recognition." [arXiv: 1707.07012]

IMPLICATIONS

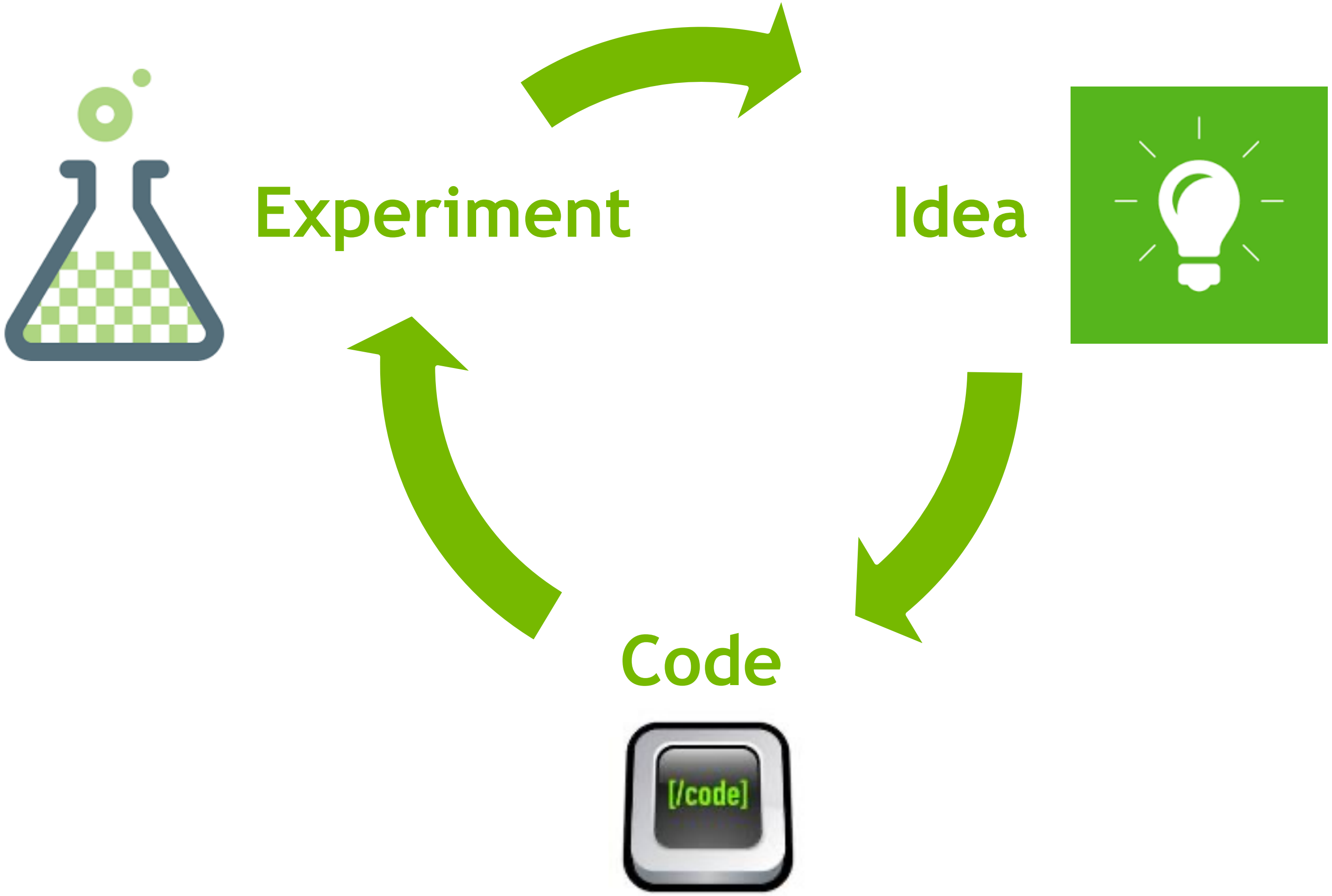
IMPLICATIONS

Good and bad news

- The good news: Requirements are predictable.
  - We can predict how much data we will need.
  - We can predict how much computing power we will need.
- The bad news: The values can be significant.
  - The silver lining is that deep learning has taken impossible problems and made them merely expensive.

IMPLICATIONS

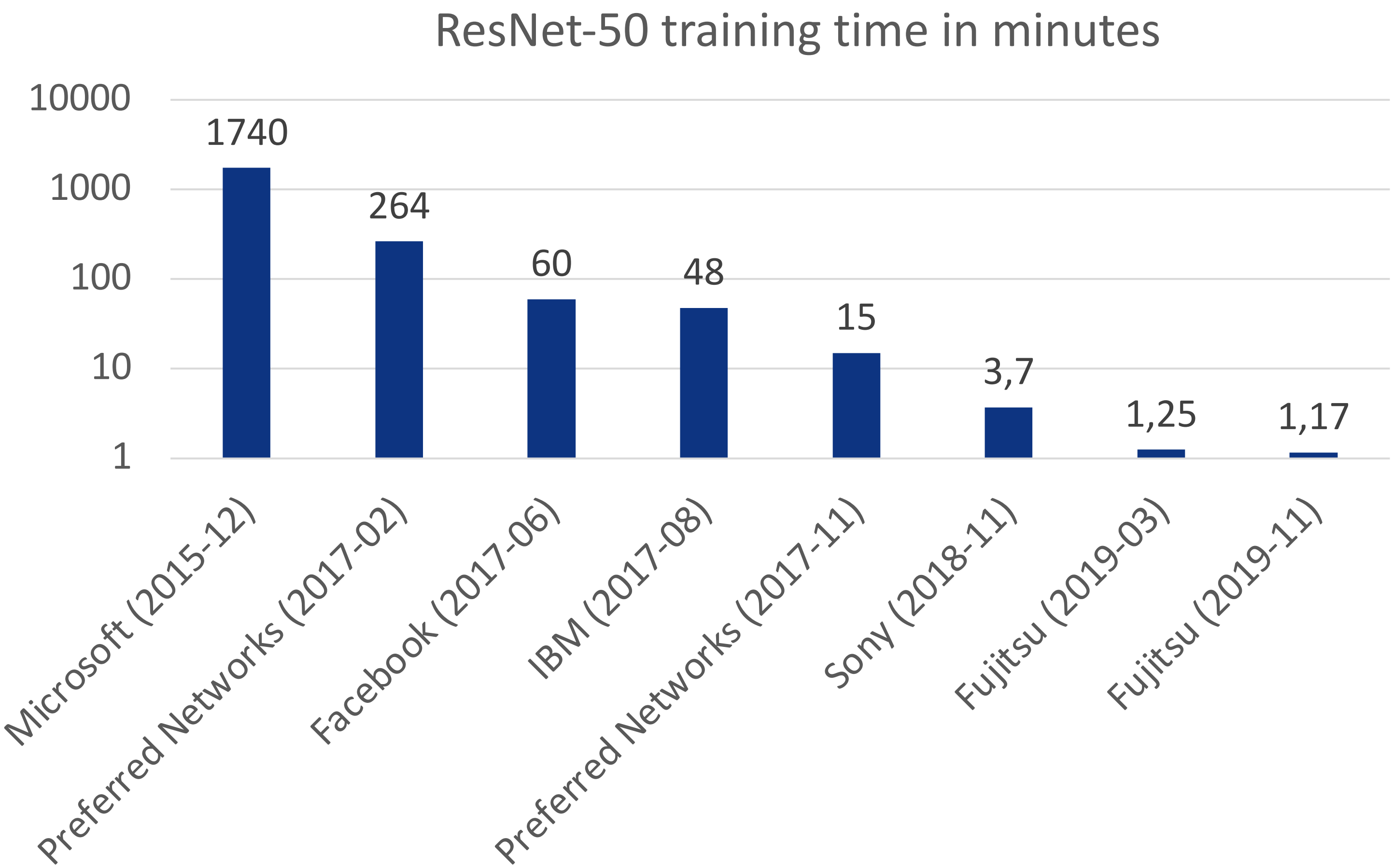
Deep learning is experimental; we need to train quickly to iterate

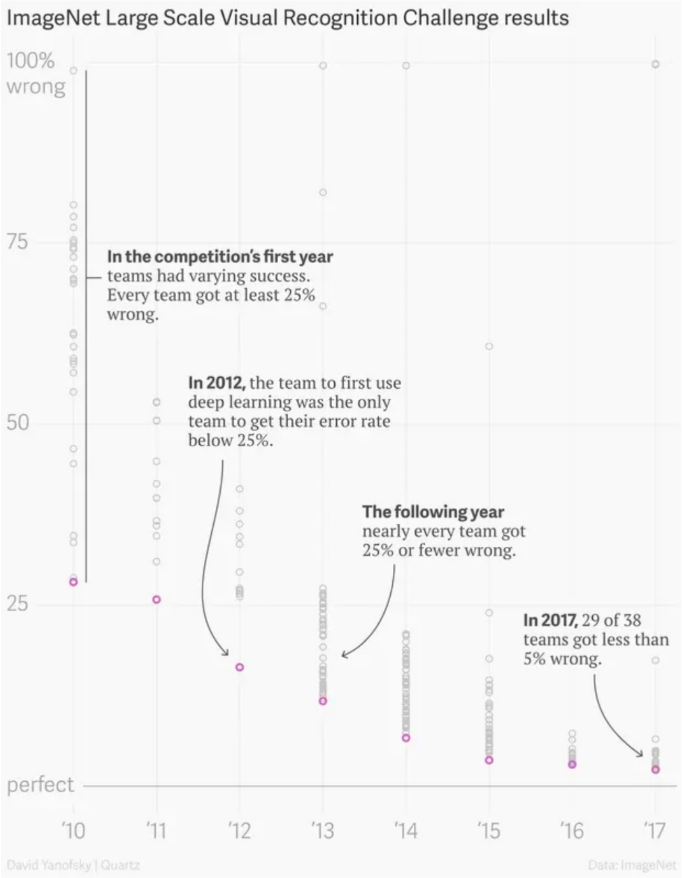


NVIDIA DEEP LEARNING INSTITUTE

ITERATION TIME

Short iteration time is fundamental for success

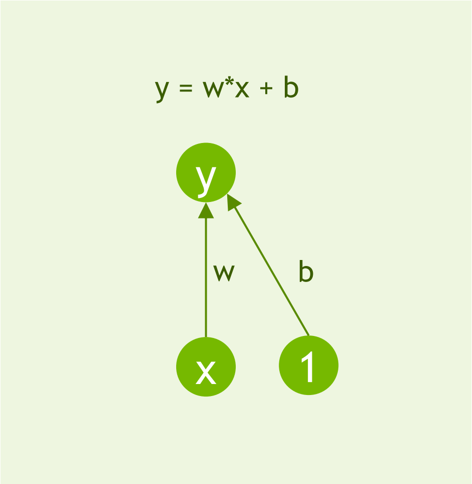


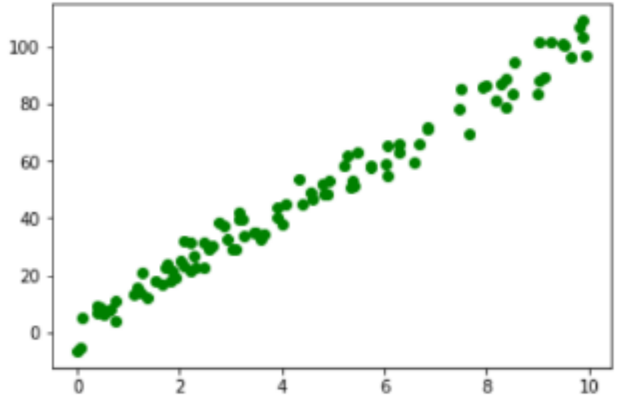


INTRO TO THE LAB

STARTING WITH A LINEAR MODEL

Our goal is to find best model parameters (combination of w and b) to fit the data





DATA PARALLELISM: HOW TO TRAIN DEEP LEARNING MODELS ON MULTIPLE GPUS

LAB 1, PART 2: MORE REALISTIC NETWORKS

NVIDIA | DEEP LEARNING INSTITUTE

MODERN NEURAL NETWORKS

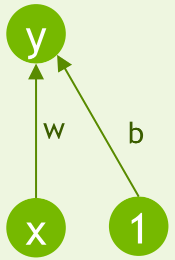
How do they differ from our trivial example?

Not significantly!

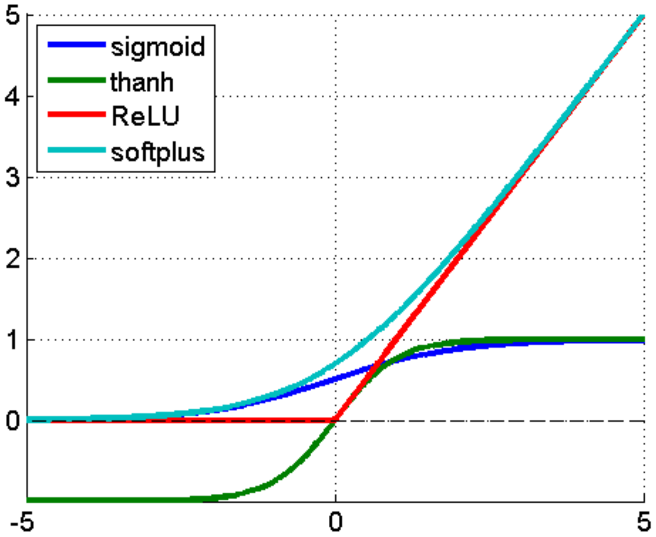
MODERN NEURAL NETWORKS

How do they differ from our trivial example?

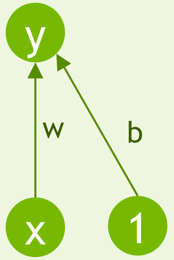
$y = w \cdot x + b$



Nonlinearity



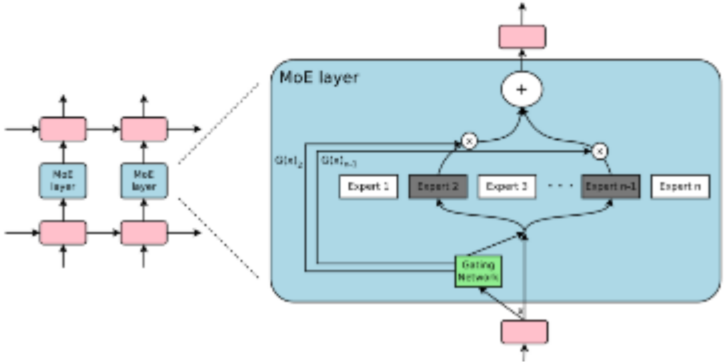
$y = f(w \cdot x + b)$

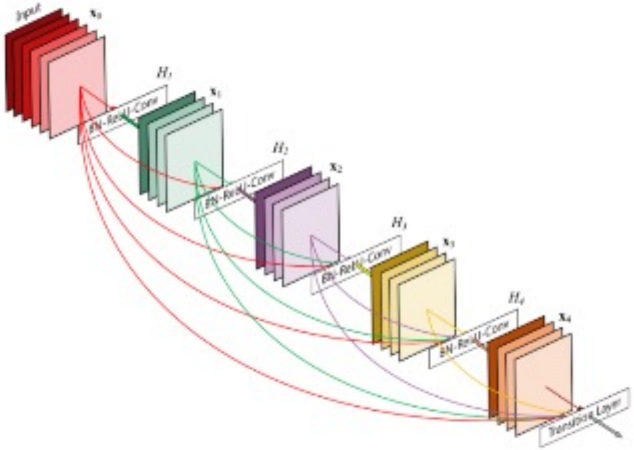


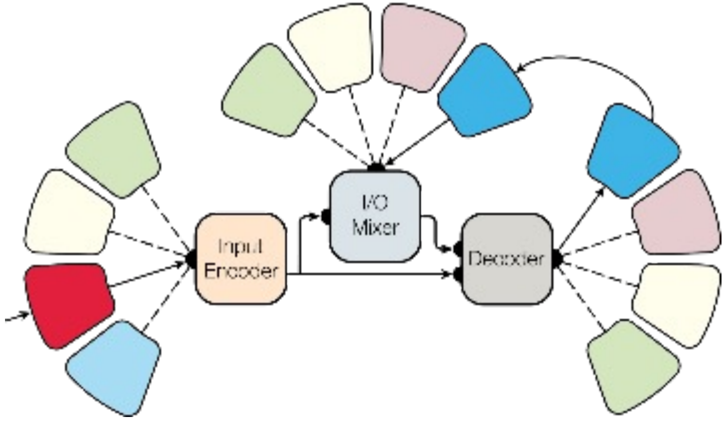
MODERN NEURAL NETWORKS

How do they differ from our trivial example?

More complex interconnection and many more parameters



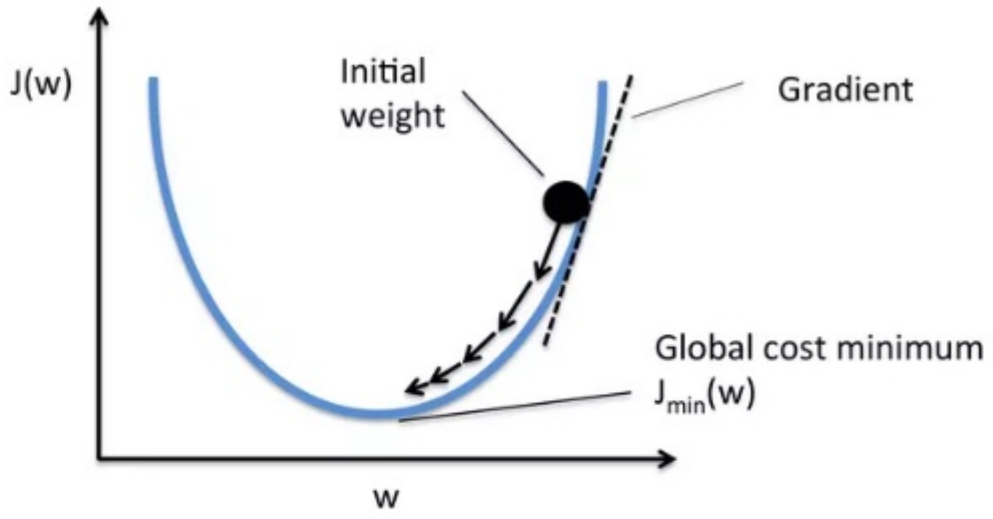


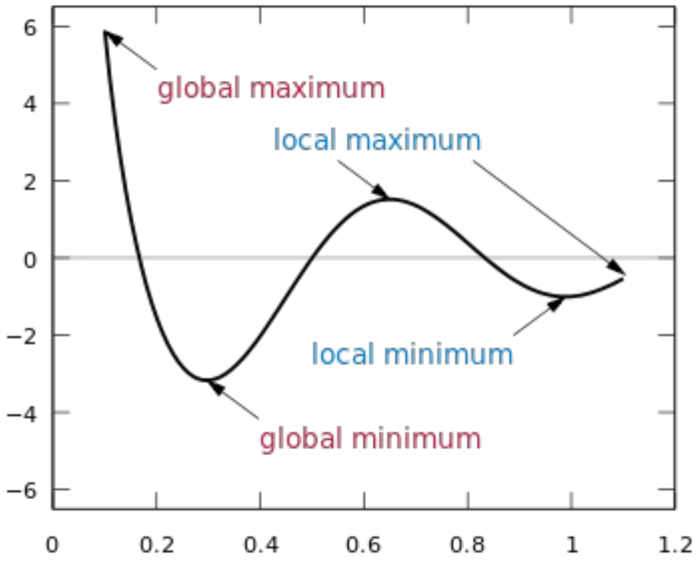


Kaiser, L., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., & Uszkoreit, J. (2017). One model to learn them all. *arXiv preprint arXiv:1706.05137*.
Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., & Keutzer, K. (2014). Densenet: Implementing efficient convnet descriptor pyramids. arXiv preprint arXiv:1404.1869.
Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

NON-CONVEX LOSS FUNCTIONS

Those differences make the optimization problem much more difficult

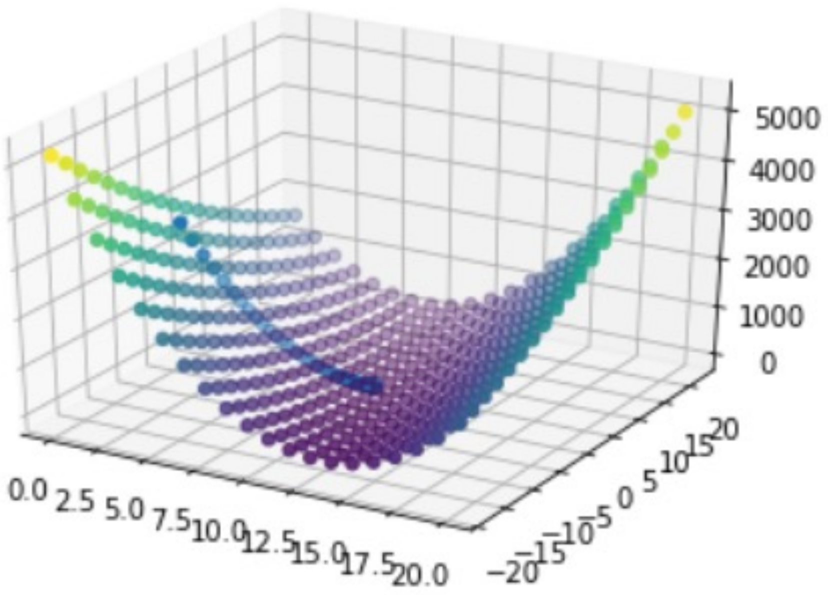




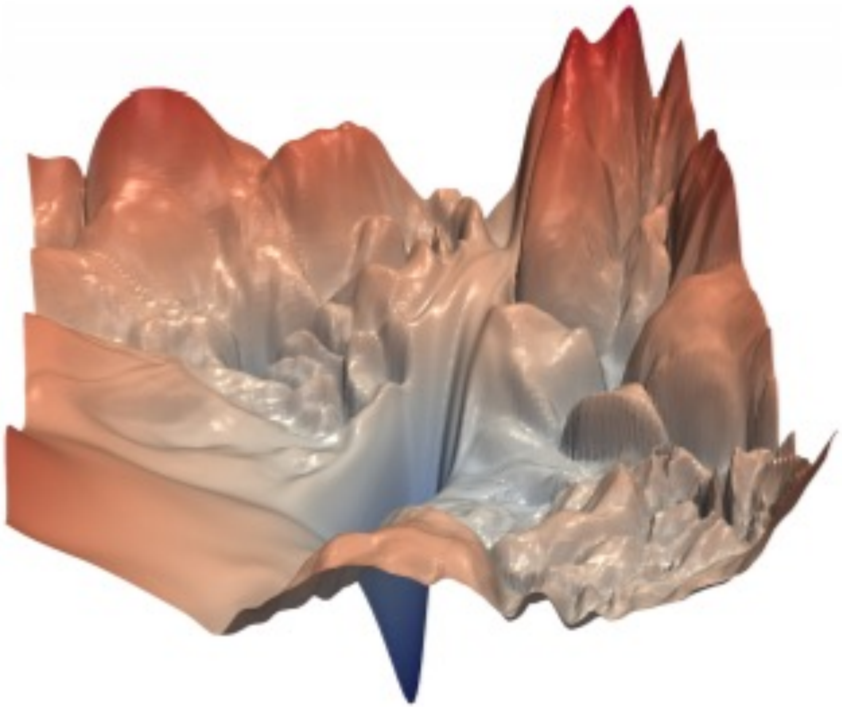
NON-CONVEX LOSS FUNCTIONS

Those differences make the optimization problem much more difficult

Linear model loss function



ResNet-56 loss function projection to 3D – no skip connections



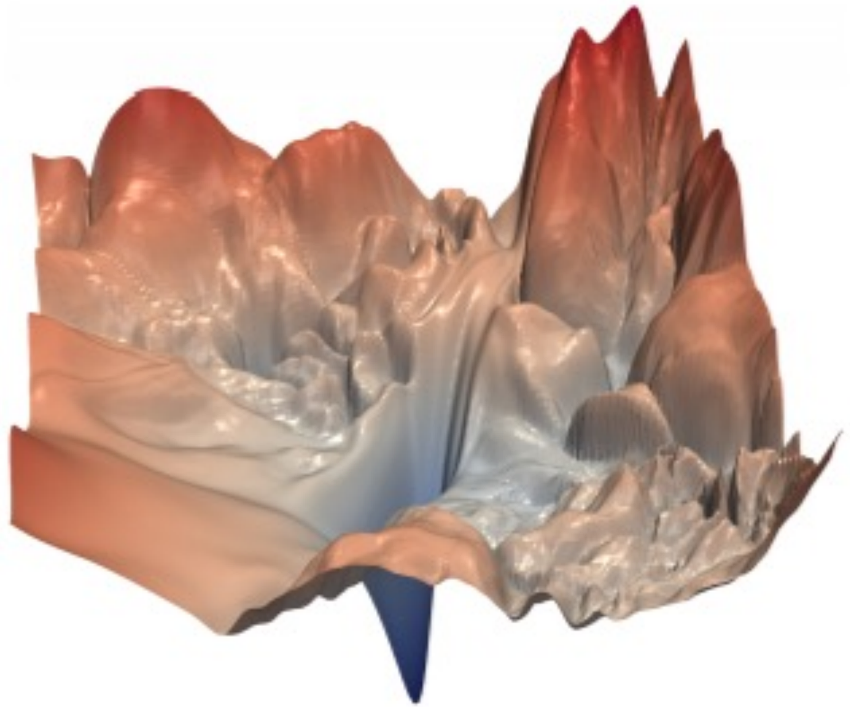
Li, H., Xu, Z., Taylor, G., & Goldstein, T. (2017). Visualizing the Loss Landscape of Neural Nets. *arXiv:1712.09913*.

NON-CONVEX LOSS FUNCTIONS

Those differences make the optimization problem much more difficult

Why do we succeed in finding good local minima?

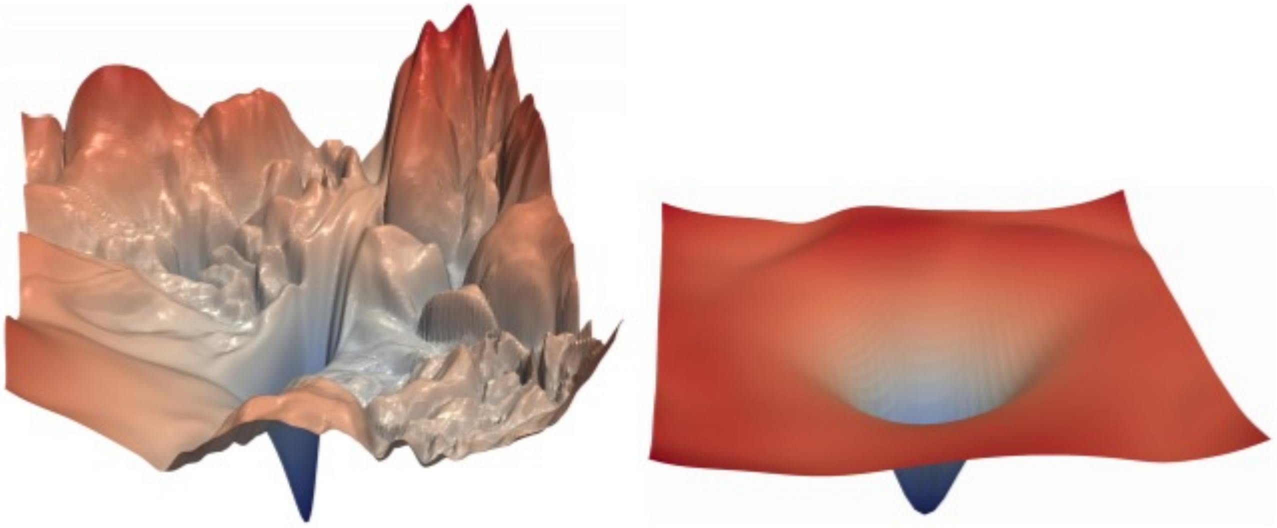
ResNet-56 loss function projection to 3D – no skip connections



Li, H., Xu, Z., Taylor, G., & Goldstein, T. (2017). Visualizing the Loss Landscape of Neural Nets. *arXiv:1712.09913*.

NON-CONVEX LOSS FUNCTIONS

Recent advances such as residual connections simplify optimization



(a) without skip connections (b) with skip connections

Li, H., Xu, Z., Taylor, G., & Goldstein, T. (2017). Visualizing the Loss Landscape of Neural Nets. arXiv:1712.09913.

DATA PARALLELISM: HOW TO TRAIN DEEP LEARNING MODELS ON MULTIPLE GPUS

LAB 1 CONCLUSION: DATA AND MODEL PARALLELISM

NVIDIA. DEEP LEARNING INSTITUTE

DATA PARALLELISM

Focus of this course

How can we take advantage of multiple GPUs to reduce the training time?

DATA VS MODEL PARALLELISM

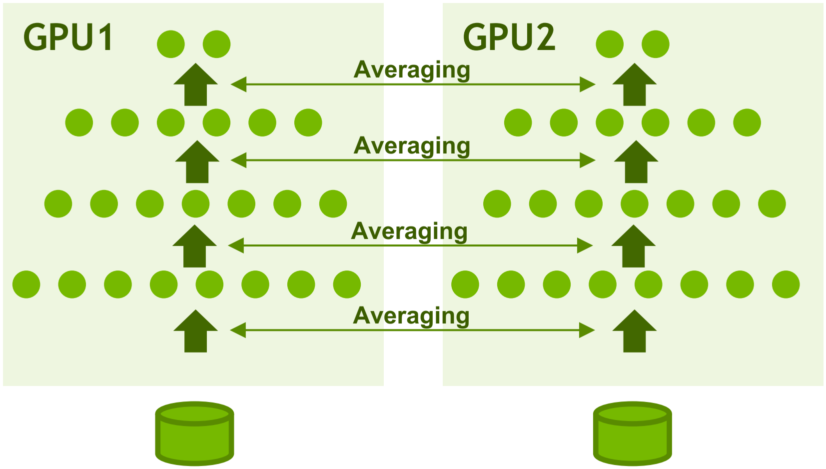
Comparison

- Data Parallelism
  - Allows you to speed up training
  - All workers train on different data
  - All workers have the same copy of the model
  - Neural network gradients (weight changes) are exchanged
- Model Parallelism
  - Allows you to use a bigger model
  - All workers train on the same data
  - Parts of the model are distributed across GPUs
  - Neural network activations are exchanged

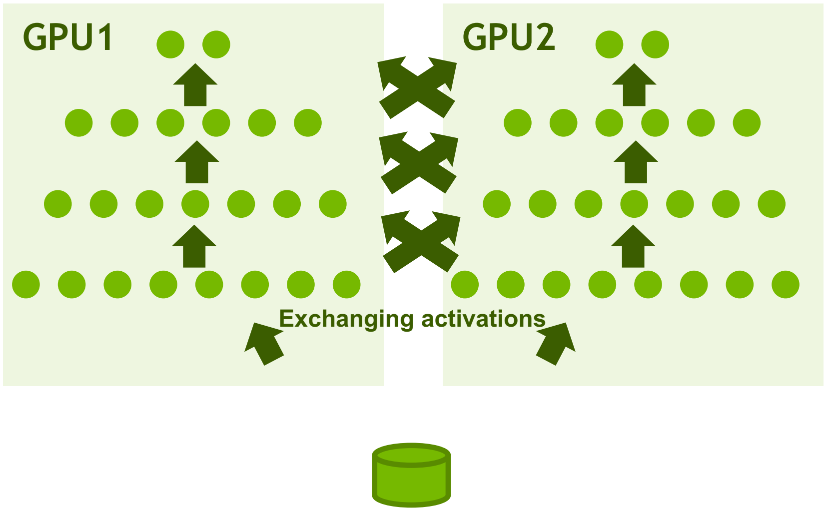
DATA VS MODEL PARALLELISM

Comparison

- Data Parallelism

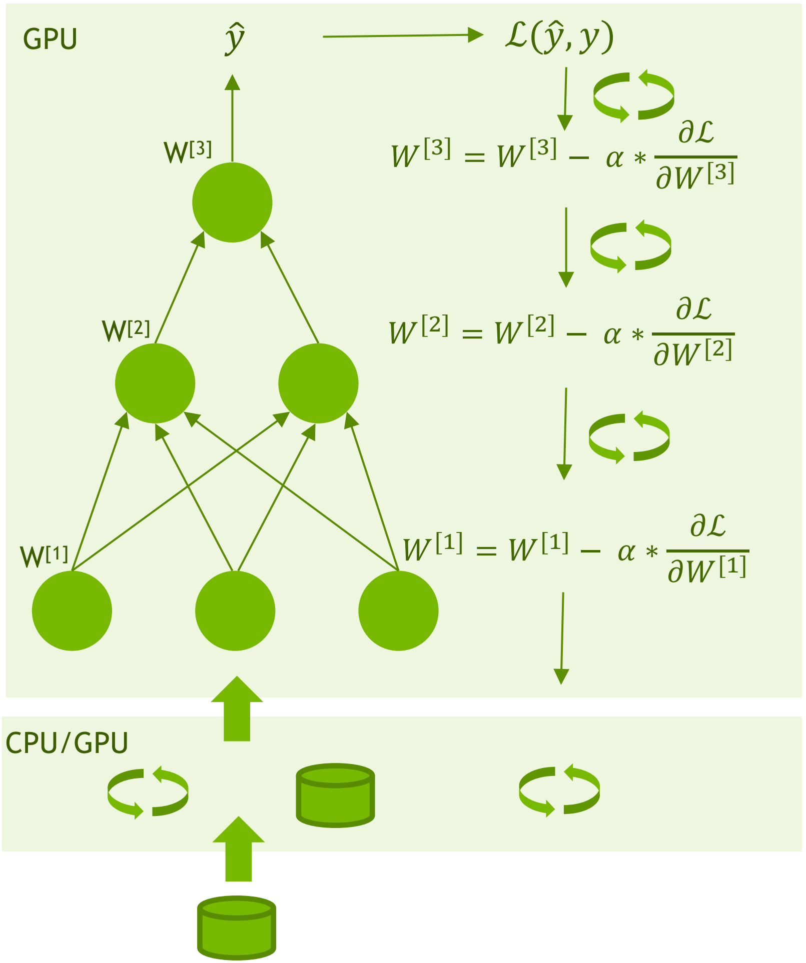


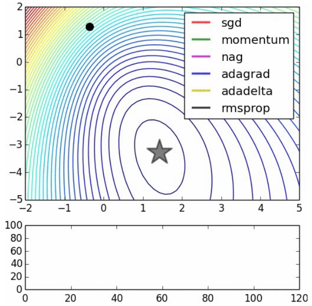
- Model Parallelism



TRAINING A NEURAL NETWORK

Single GPU

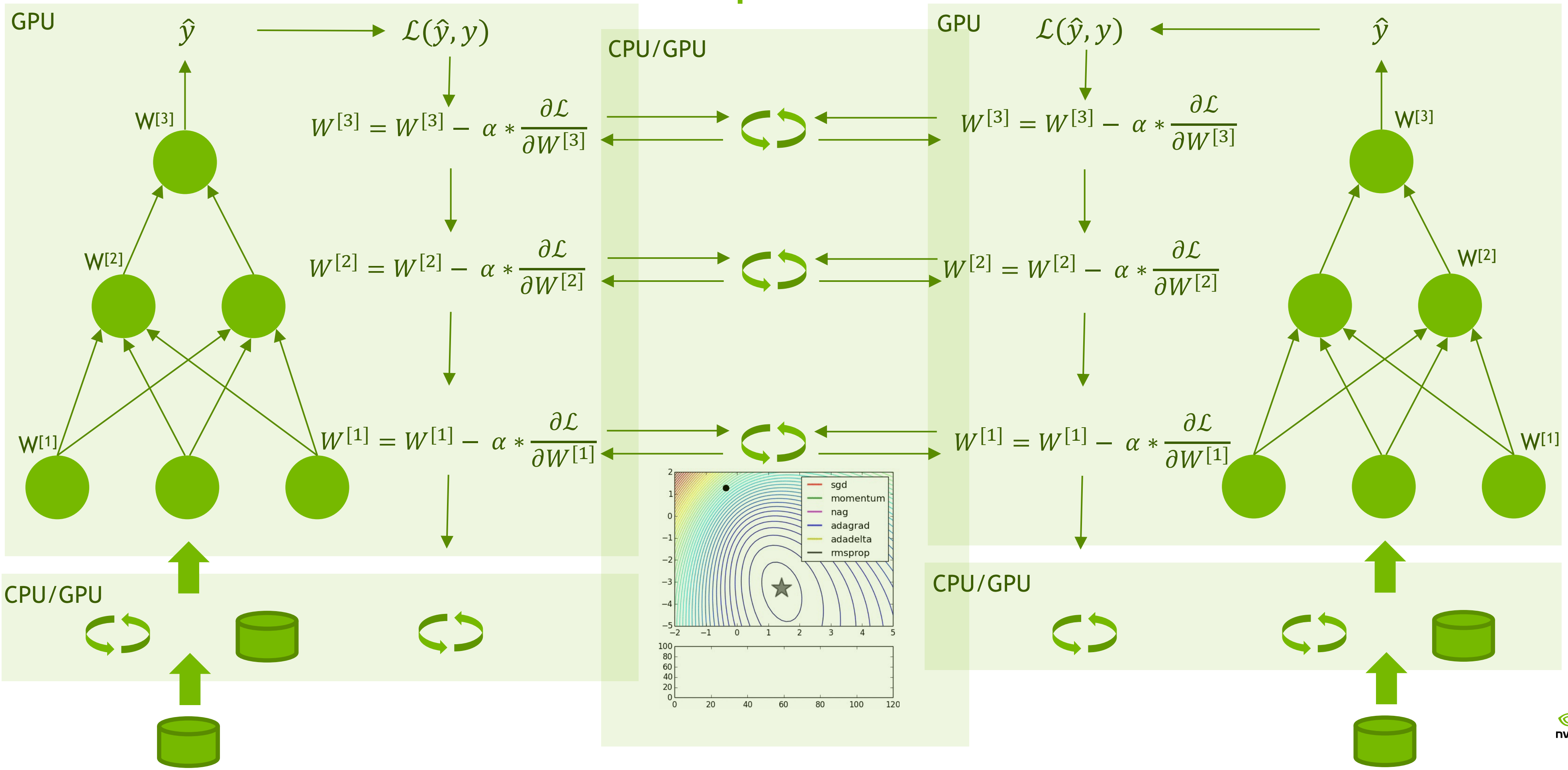




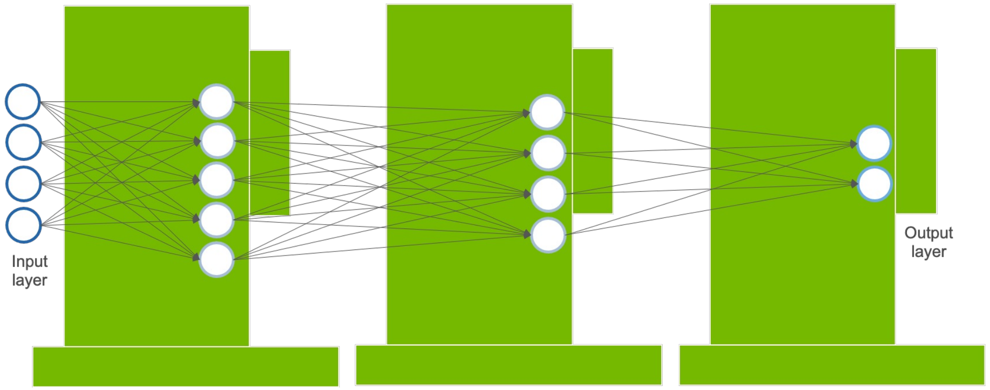
1. Read the data
2. Transport the data
3. Pre-process the data
4. Queue the data
5. Transport the data
6. Calculate activations for layer one
7. Calculate activations for layer two
8. Calculate the output
9. Calculate the loss
10. Backpropagate through layer three
11. Backpropagate through layer two
12. Backpropagate through layer one
13. Execute optimization step
14. Update the weights
15. Return control

TRAINING A NEURAL NETWORK

Multiple GPUs



PARALLEL/DISTRIBUTED ML TRAINING



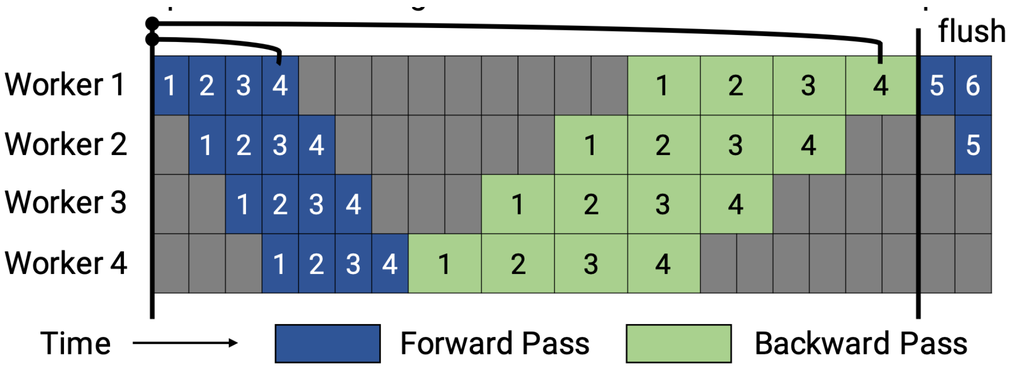
1. Model Parallelism: Memory usage and computation of a model distributed across devices

Two main variants:

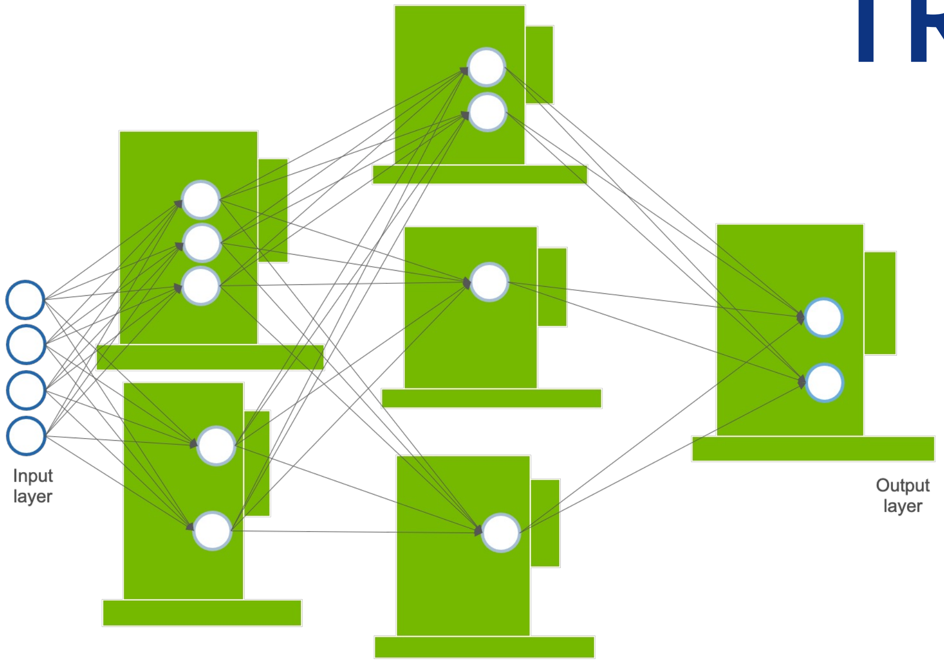
a) Pipeline parallelism
b) Tensor parallelism

Pipeline Model

- Complete layer per device
  - Weights stay within device
- Activations are communicated between GPUs
- Non efficient implementations may lead to inefficient usage of resources
  - Research area



PARALLEL/DISTRIBUTED ML TRAINING



1. Model Parallelism: Memory usage and computation of a model distributed across devices
Two main variants:
  a) Pipeline parallelism
  b) Tensor parallelism

Tensor Parallelism

- Tensor operations (e.g., computing a layer output) distributed across device
  - Allows larger, more computationally expensive models
- Activations are communicated between GPUs
- Further points for inefficiencies
  - A device might depend on the activations computed by more than one device

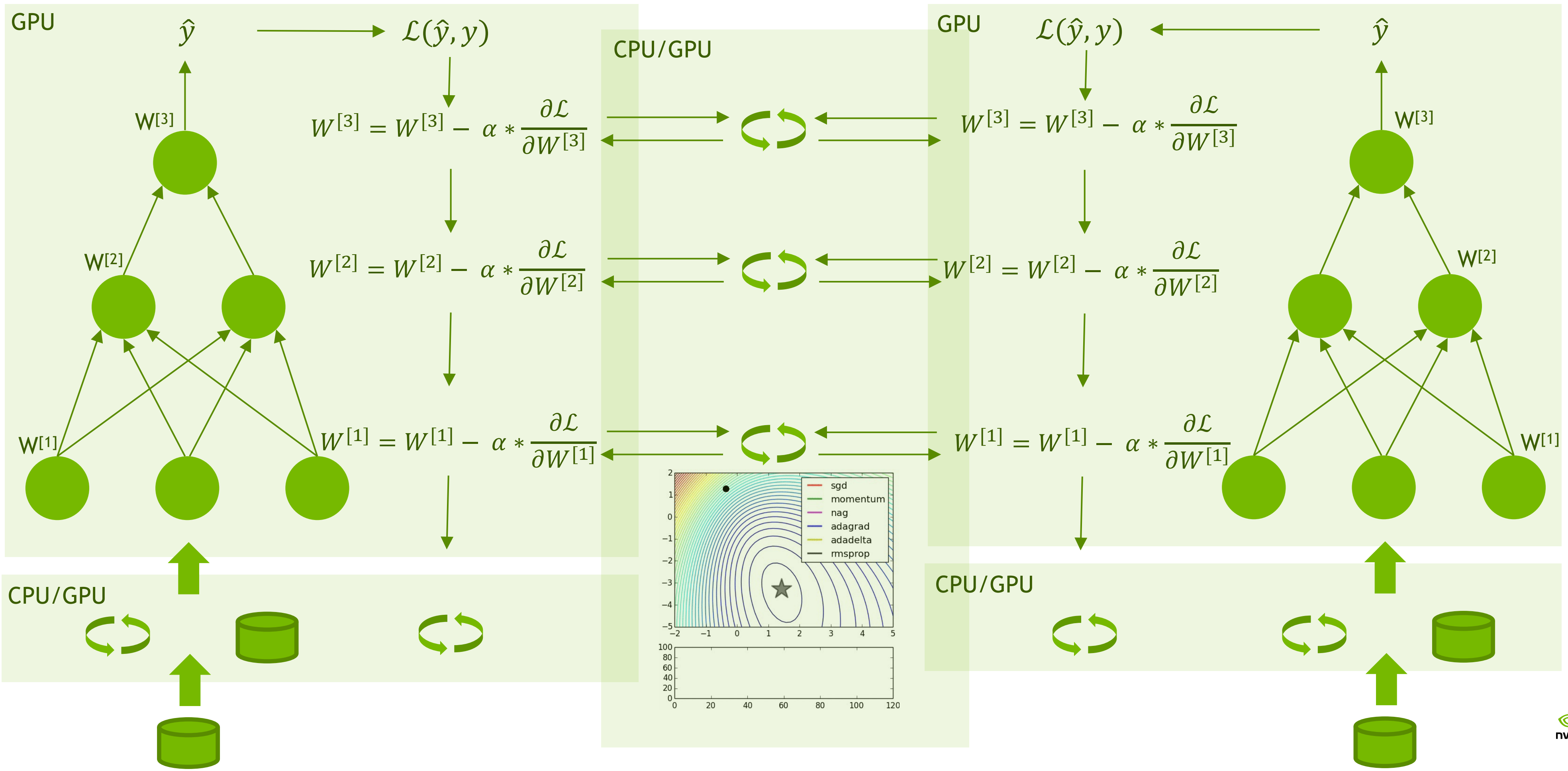
DATA PARALLELISM: HOW TO TRAIN DEEP LEARNING MODELS ON MULTIPLE GPUS

LAB 2, PART 1: INTRODUCTION TO DISTRIBUTED DATA PARALLEL (DDP)

NVIDIA | DEEP LEARNING INSTITUTE

TRAINING A NEURAL NETWORK

Multiple GPUs



MEET DDP

Library for distributed DL

Prepackaged into and optimized for PyTorch, an increasingly popular platform among ML engineers and researchers

USING DISTRIBUTED DATA PARALLEL (DDP)

INITIALIZE THE PROCESS

```
def setup(global_rank, world_size):
    dist.init_process_group(backend="nccl", rank=global_rank,
    world_size=world_size)
```


NVIDIA. DEEP LEARNING INSTITUTE

PIN GPU TO BE USED

```
device = torch.device("cuda:" + str(local_rank))
model = Net().to(device)
```


NVIDIA. DEEP LEARNING INSTITUTE

ENCAPSULATE MODEL WITH DDP

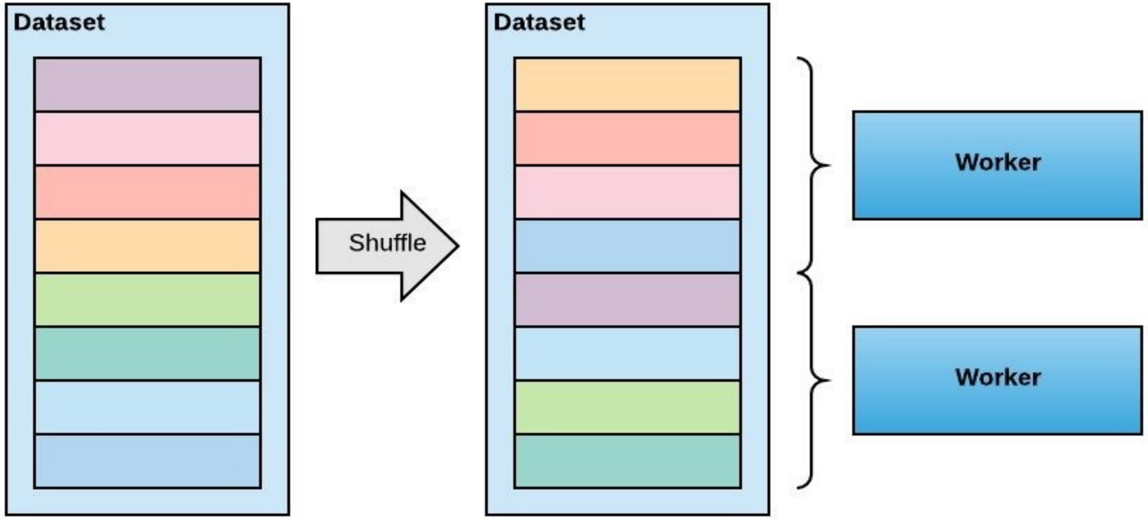
```
model = nn.parallel.DistributedDataParallel(model,
device_ids=[local_rank])
```

SYNCHRONIZE INITIAL STATE

Handled internally by DDP across processes and nodes!

NVIDIA. DEEP LEARNING INSTITUTE

DATA PARTITIONING



Shuffle the dataset

Partition records among workers

Train by sequentially reading the partition

After epoch is done, reshuffle and partition again

NVIDIA. DEEP LEARNING INSTITUTE

DATA PARTITIONING

```
train_sampler =
torch.utils.data.distributed.DistributedSampler(train_set,
num_replicas=world_size, rank=global_rank)

train_loader =
torch.utils.data.DataLoader(train_set,
batch_size=args.batch_size, sampler=train_sampler)
```


NVIDIA. DEEP LEARNING INSTITUTE

I/O ON ONLY ON ONE WORKER

```
download = True if local_rank == 0 else False
if local_rank == 0:
      train_set = torchvision.datasets.FashionMNIST("./data",
download=download)

--------------------

if global_rank == 0:
      print("Epoch = {:2d}: Validation Loss = {:5.3f},
      Validation Accuracy = {:5.3f}".format(epoch+1, v_loss,
      val_accuracy[-1]))
```

DATA PARALLELISM: HOW TO TRAIN DEEP LEARNING MODELS ON MULTIPLE GPUS

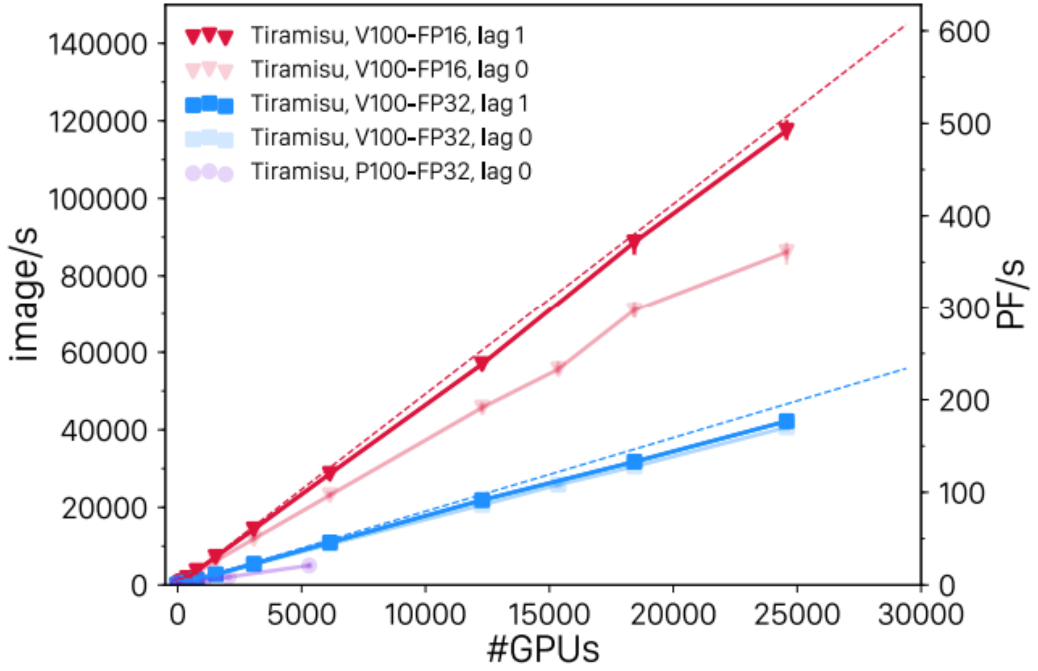
LAB 3, PART 1: SCALING THE BATCH SIZE

NVIDIA. DEEP LEARNING INSTITUTE

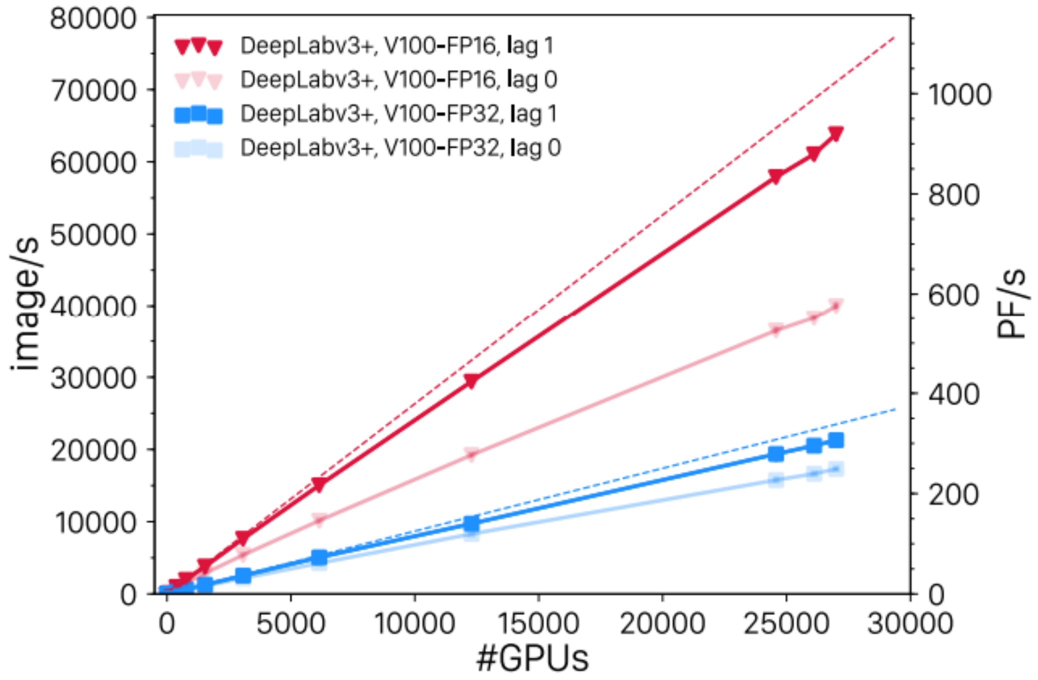
CAN WE INCREASE THE BATCH SIZE INDEFINITELY?

IN TERMS OF IMAGES / SECOND?

Yes



(a) Tiramisu



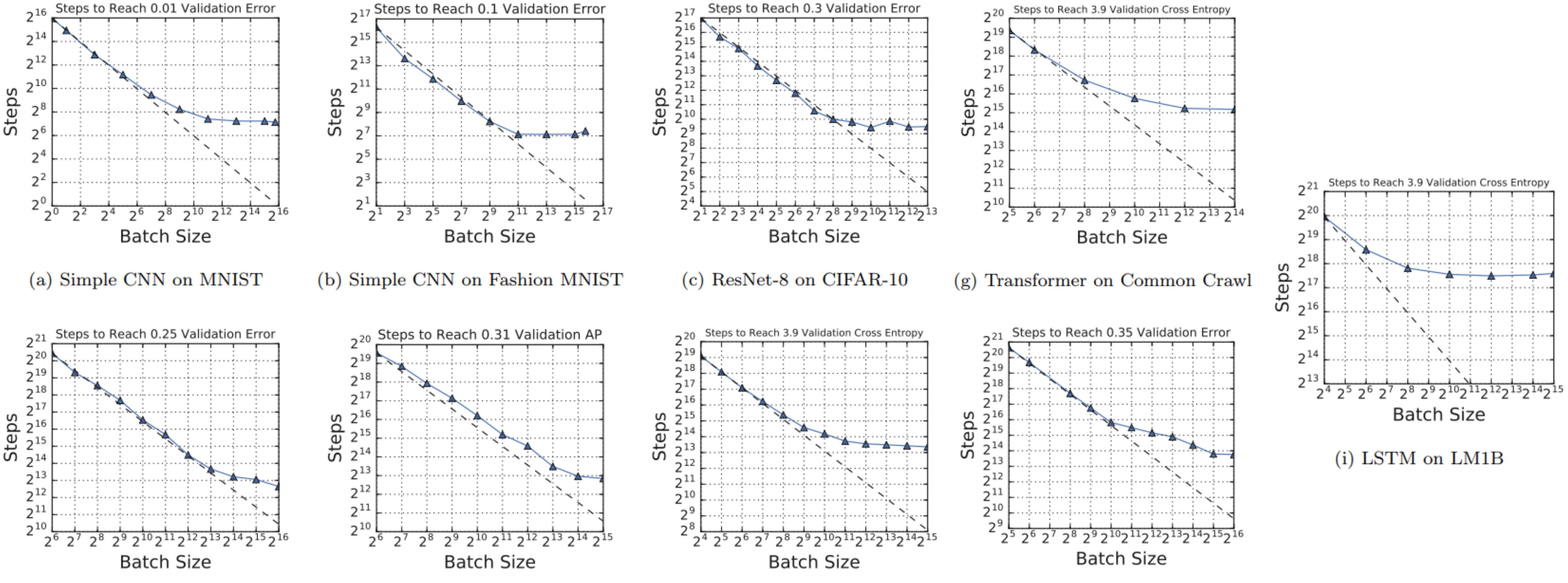
(b) DeepLabv3+

Kurth, T., Treichler, S., Romero, J., Mudigonda, M., Luehr, N., Phillips, E., ... & Houston, M. (2018, November). Exascale deep learning for climate analytics. In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (p. 51). IEEE Press. arXiv:1810.01993

NVIDIA. DEEP LEARNING INSTITUTE

IN TERMS OF STEPS TO CONVERGENCE?

There are limits



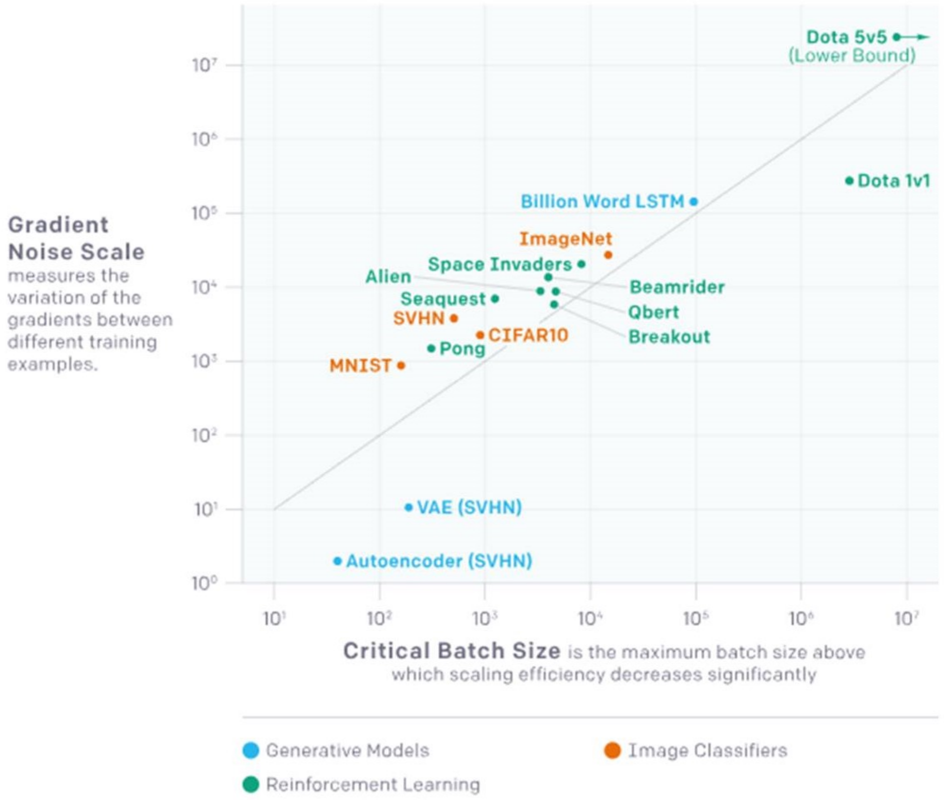
(a) Simple CNN on MNIST (b) Simple CNN on Fashion MNIST (c) ResNet-8 on CIFAR-10 (g) Transformer on Common Crawl (i) LSTM on LM1B

(d) ResNet-50 on ImageNet (e) ResNet-50 on Open Images (f) Transformer on LM1B (h) VGG-11 on ImageNet

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., & Dahl, G. E. (2018). Measuring the effects of data parallelism on neural network training. arXiv:1811.03600

IN TERMS OF STEPS TO CONVERGENCE?

There are limits

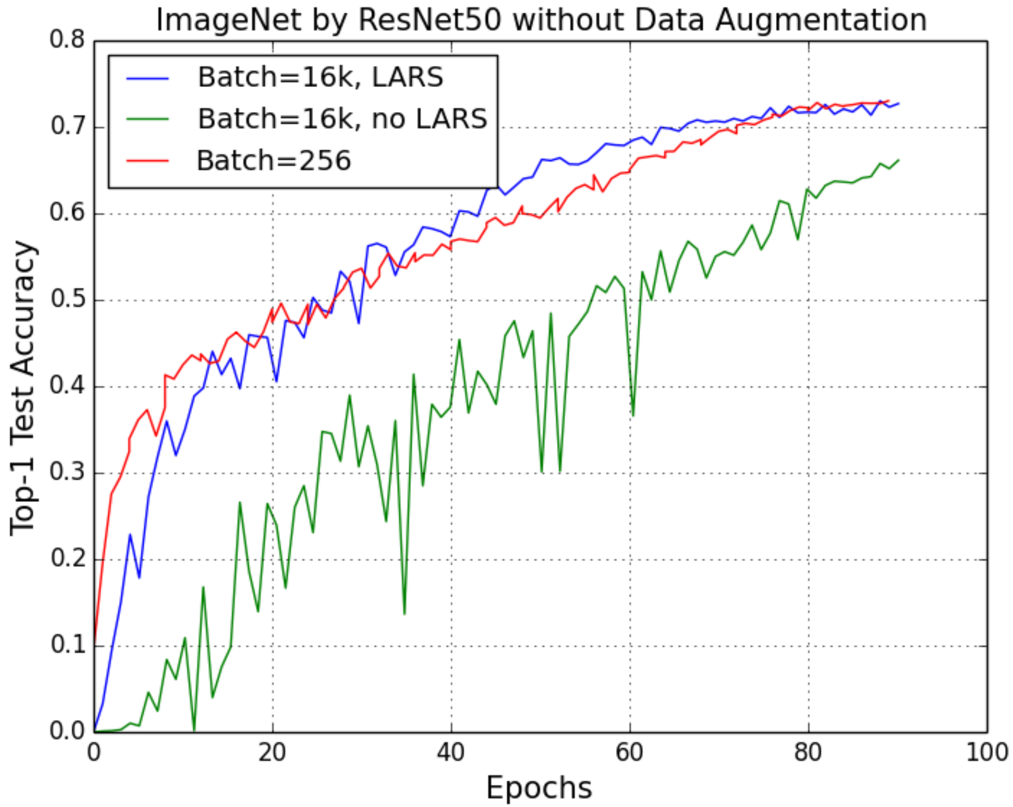


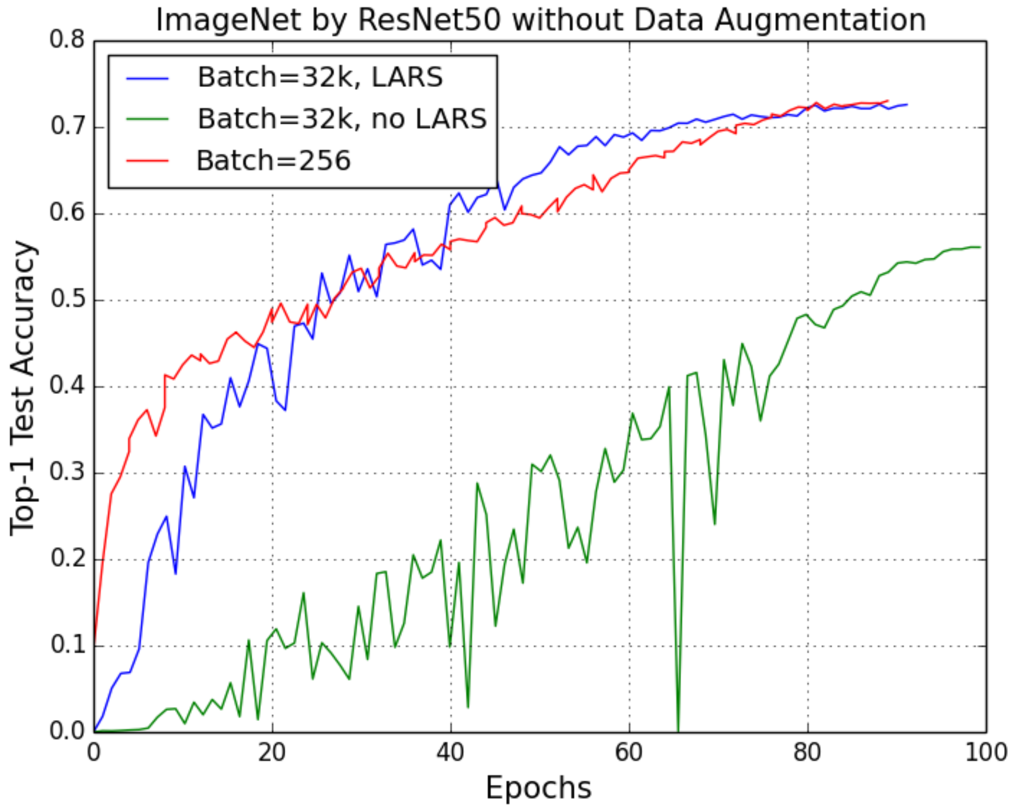
https://blog.openai.com/science-of-ai/

LARGE MINIBATCH AND ITS IMPACT ON ACCURACY

IMPACT ON ACCURACY

Naïve approaches lead to degraded accuracy

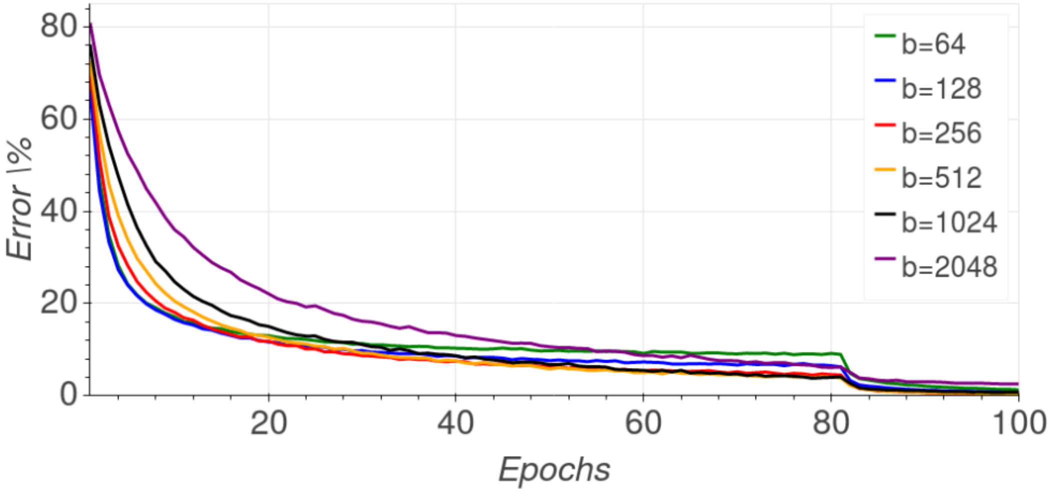




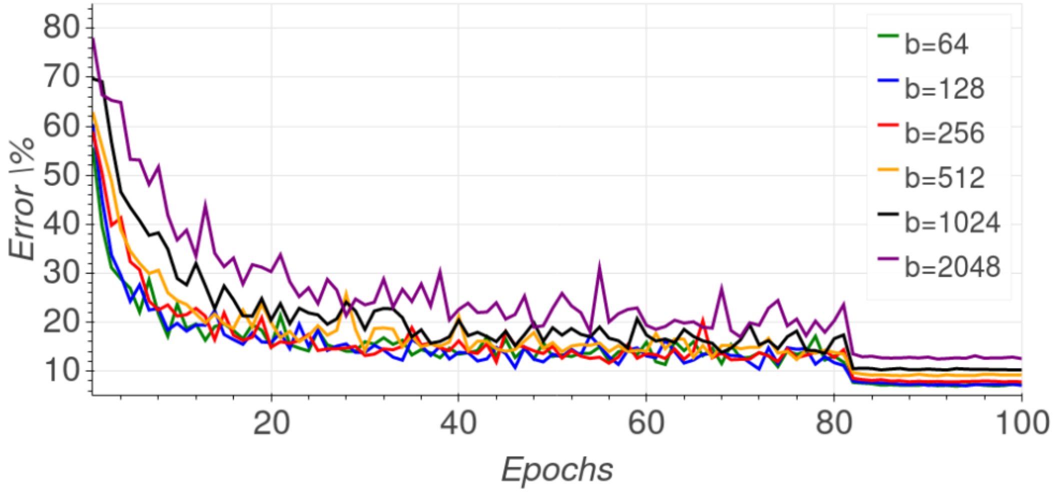
You, Y., Zhang, Z., Hsieh, C., Demmel, J., & Keutzer, K. (2017). ImageNet training in minutes. arXiv: 1709.05011

IMPACT ON ACCURACY

Naïve approaches lead to degraded accuracy



(a) Training error

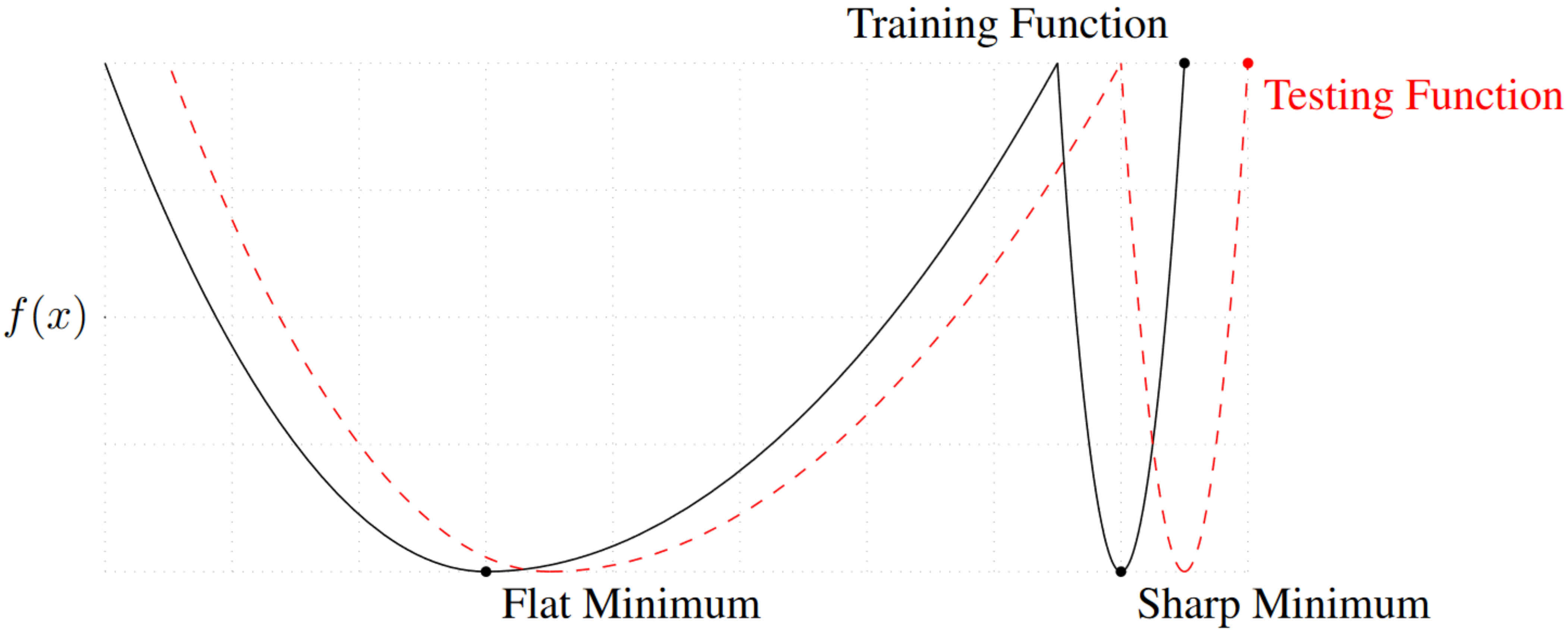


(b) Validation error

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741

IMPACT ON ACCURACY

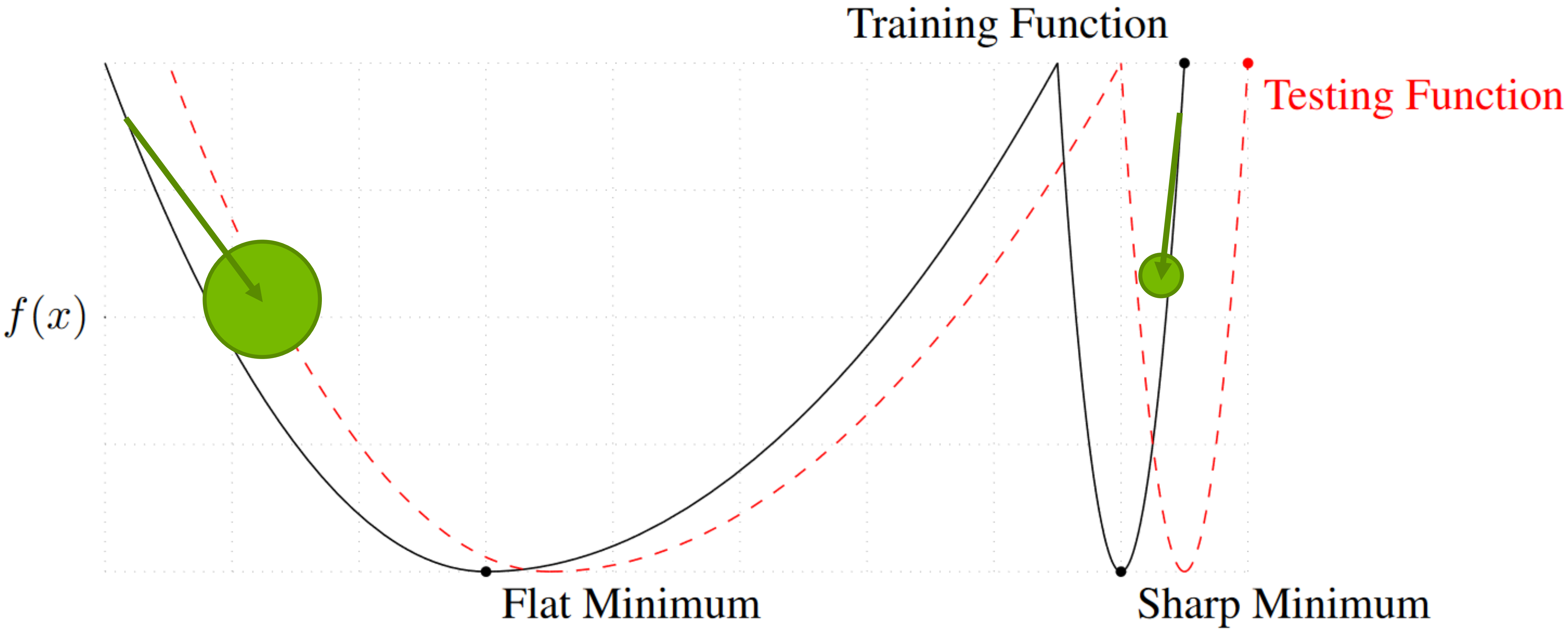
Why? Generalization and flatness of minima?



Keskar, N. S., et al. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. arXiv:1609.04836

IMPACT ON ACCURACY

Why does it happen? Noise in the gradient update.



Keskar, N. S., et al. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. arXiv:1609.04836

IMPACT ON ACCURACY

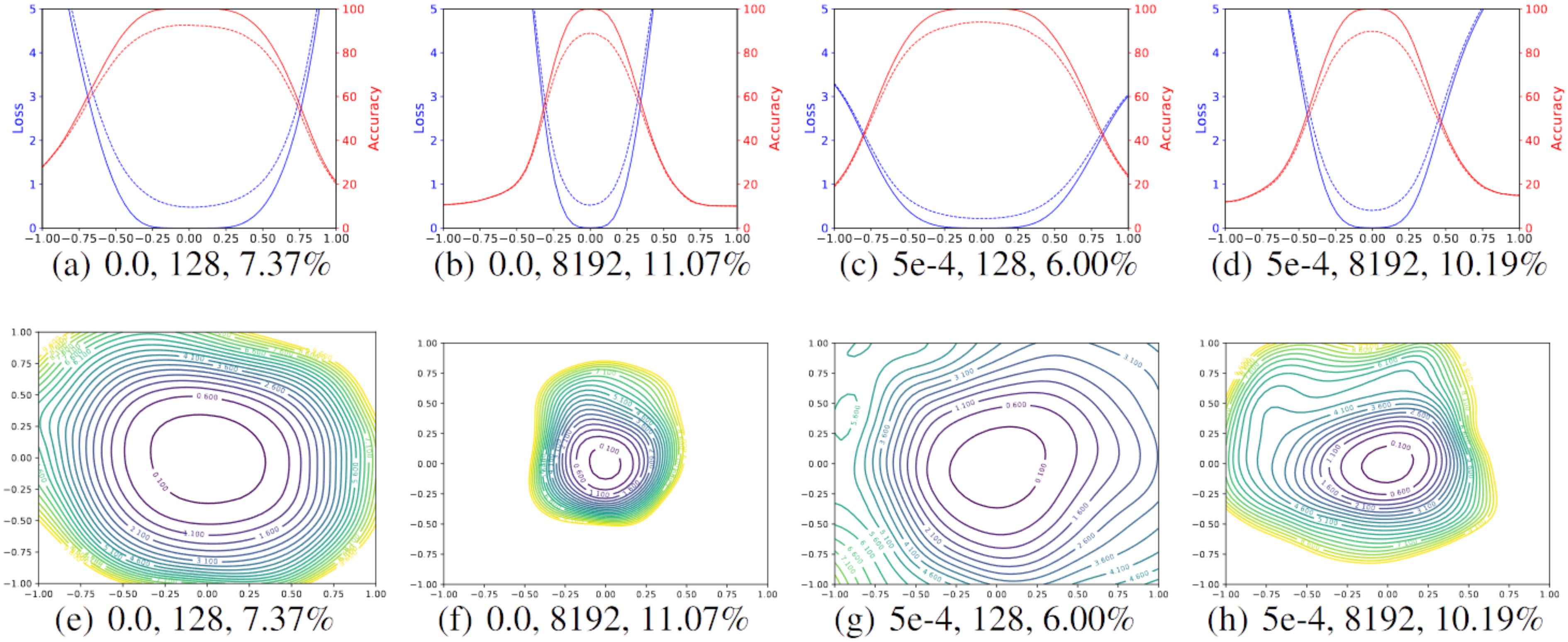


Figure 3: The 1D and 2D visualization of solutions obtained using SGD with different weight decay and batch size. The title of each subfigure contains the weight decay, batch size, and test error.

Li, H., Xu, Z., Taylor, G., & Goldstein, T. (2017). Visualizing the Loss Landscape of Neural Nets. arXiv:1712.09913

DATA PARALLELISM: HOW TO TRAIN DEEP LEARNING MODELS ON MULTIPLE GPUS

LAB 3, PART 2: OPTIMIZATION STRATEGIES

DEEP
LEARNING
INSTITUTE

WHAT CAN WE DO TO IMPROVE THE OPTIMIZATION PROCESS?

- Manipulate the learning rate?
- Add noise to the gradient?
- Manipulate the batch size?
- Change the learning algorithm?

WHAT CAN WE DO ABOUT IT?

Early approaches: scaling the learning rate

"Theory suggests that when multiplying the batch size by k, one should multiply the learning rate by √(k) to keep the variance in the gradient expectation constant.

$$\mathrm{cov}\left(\Delta\mathbf{w}, \Delta\mathbf{w}\right) \approx \frac{\eta^2}{M}\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{g}_n\mathbf{g}_n^\top\right) \longrightarrow \eta \propto \sqrt{M}$$

…

Theory aside, for the batch sizes considered in this note, the heuristic that I found to work the best was to multiply the learning rate by k when multiplying the batch size by k. I can't explain this discrepancy between theory and practice."

In practice linear scaling is still frequently used.

Krizhevsky, A. (2014). One weird trick for parallelizing convolutional neural networks. [arXiv:1404.5997]

NVIDIA DEEP LEARNING INSTITUTE

WHAT CAN WE DO ABOUT IT?

Warmup strategies

- A lot of networks will diverge early in the learning process
- Warmup strategies address this challenge

Gradual warmup. We present an alternative warmup that *gradually* ramps up the learning rate from a small to a large value. This ramp avoids a sudden increase of the learning rate, allowing healthy convergence at the start of training. In practice, with a large minibatch of size kn, we start from a learning rate of η and increment it by a constant amount at each iteration such that it reaches $\hat{\eta} = k\eta$ after 5 epochs (results are robust to the exact duration of warmup). After the warmup, we go back to the original learning rate schedule.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., ... & He, K. (2017). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv:1706.02677

WHAT CAN WE DO ABOUT IT?

Batch Normalization

Batch normalization improves the learning process by minimizing drift in the distribution of inputs to a layer

It allows higher learning rates and reduces the need to use dropout

The idea is to normalize the inputs to all layers in every batch (this is more sophisticated than simply normalizing the input dataset)

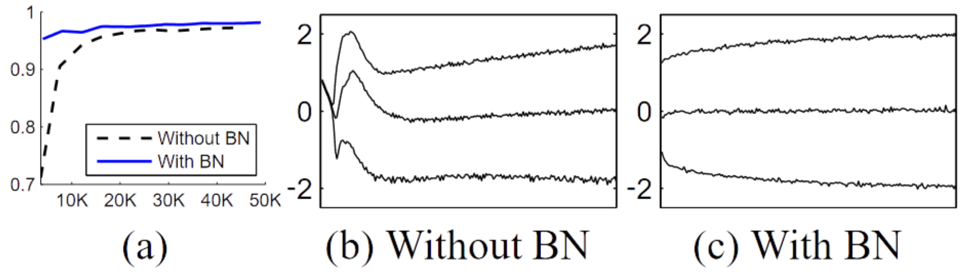


Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

Ioffe and Szegedy (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167

WHAT CAN WE DO ABOUT IT?

Ghost Batch Normalization

- The original batch normalization paper suggests using the statistics for the entire batch, but what should that mean when we have multiple GPUs?
- We can introduce additional noise by calculating smaller batch statistics (“ghost batches”).
- Batch normalization is thus carried out in isolation on a per-GPU basis.

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741

WHAT CAN WE DO ABOUT IT?

Adding noise to the gradient

- Keeps the covariance constant with changing batch size (as $\sigma^2 \propto M$)
- Does not change the mean

Furthermore, we can match both the first and second order statistics by adding multiplicative noise to the gradient estimate as follows:

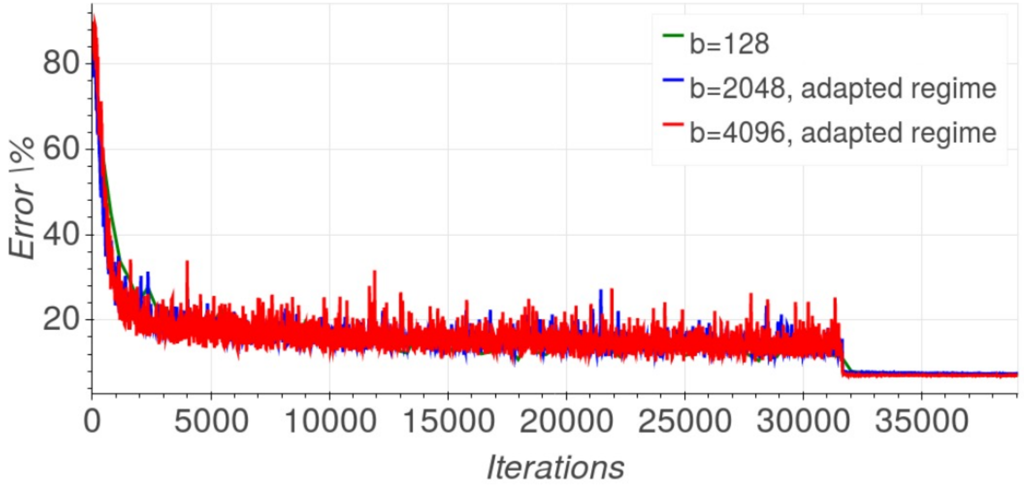
$$\hat{\mathbf{g}} = \frac{1}{M} \sum_{n \in B}^{N} \mathbf{g}_n z_n \,,$$

where $z_n \sim \mathcal{N}\left(1, \sigma^2\right)$ are independent random Gaussian variables for which $\sigma^2 \propto M$. This can be verified by using similar calculation as in appendix section A. This method keeps the covariance constant when we change the batch size, yet does not change the mean steps $\mathbb{E}\left[\Delta \mathbf{w}\right]$.

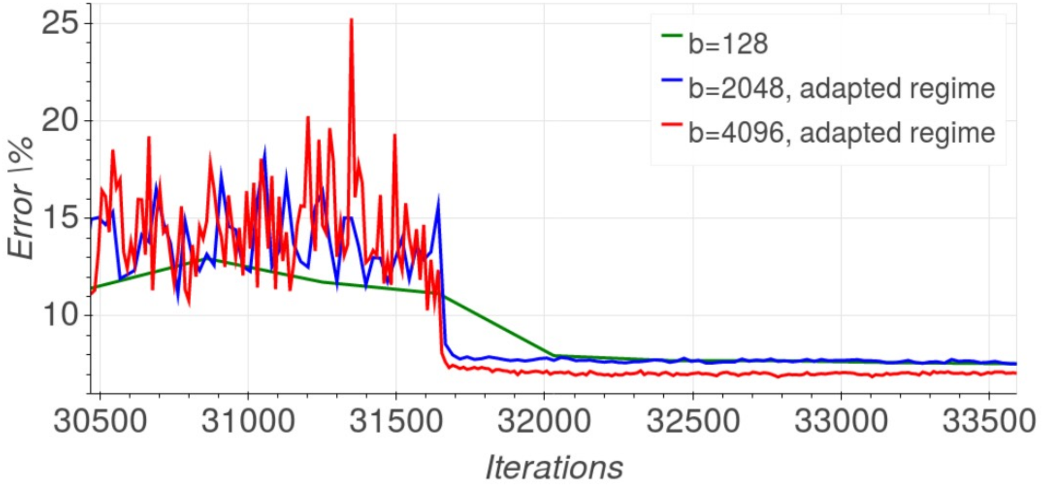
Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741

WHAT CAN WE DO ABOUT IT?

Longer training with larger learning rate



(a) Validation error

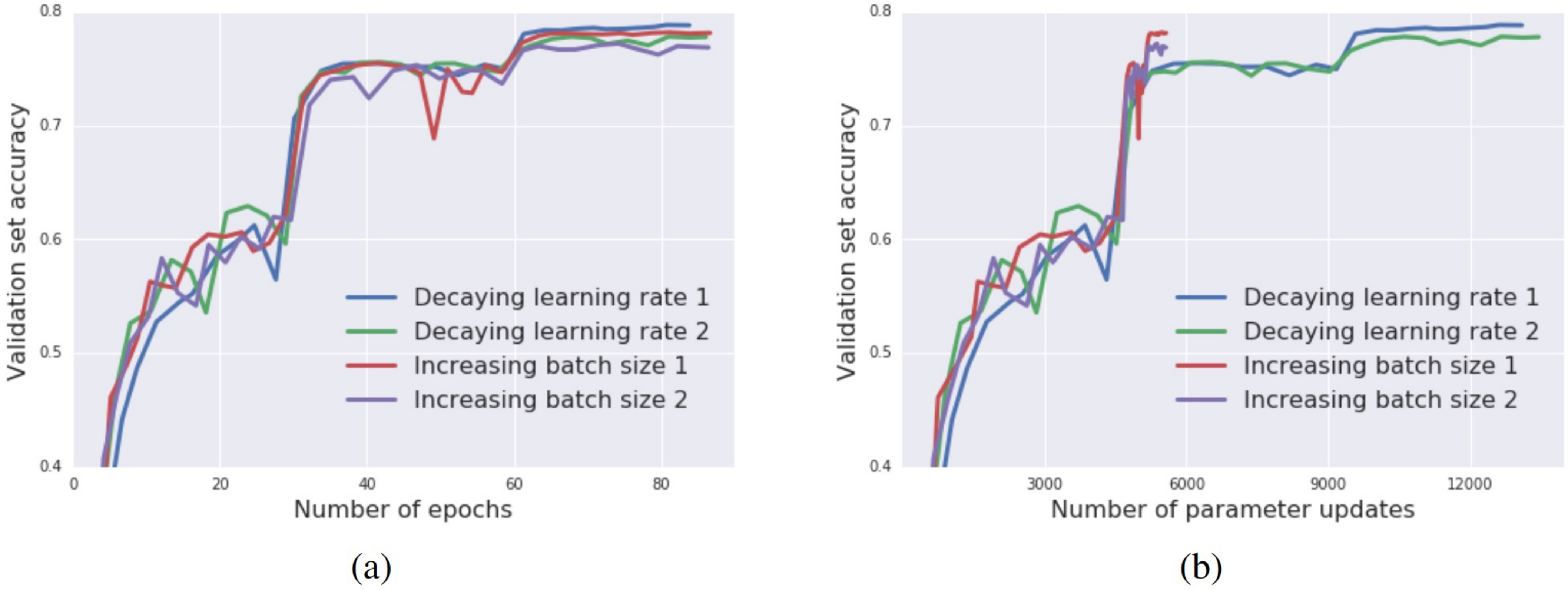


(b) Validation error - zoomed

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741

WHAT CAN WE DO ABOUT IT?

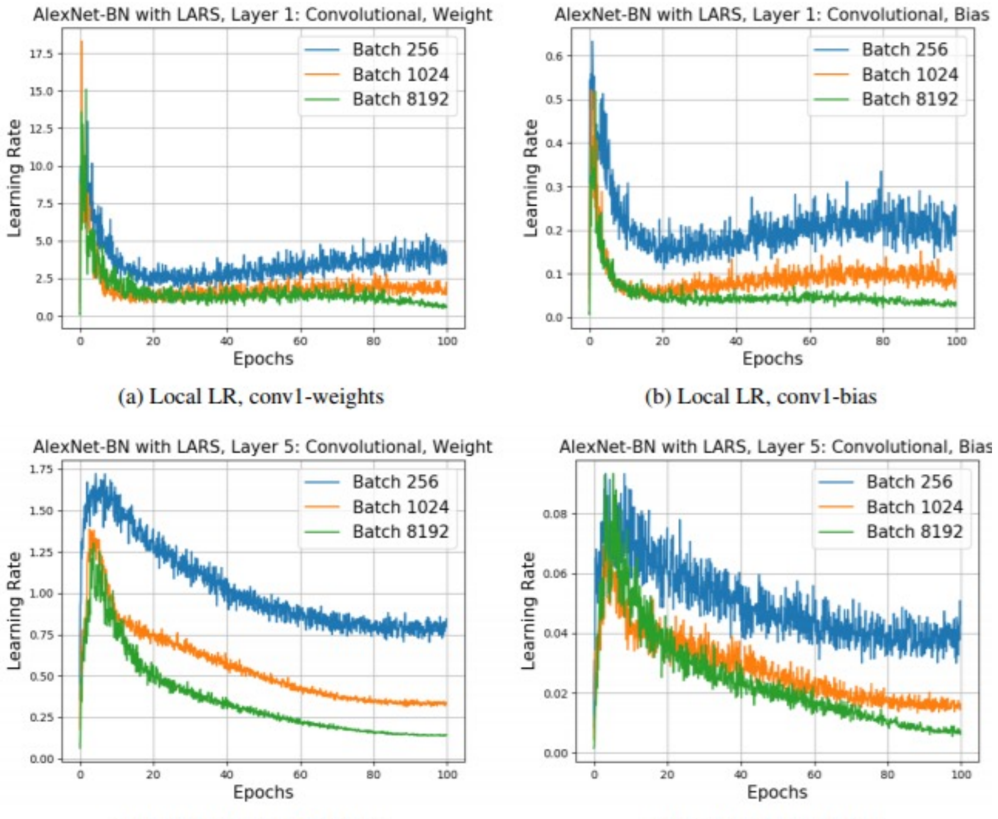
Increasing the batch size, instead of learning rate decay



Smith, S. L., Kindermans, P. J., & Le, Q. V. (2017). Don't Decay the Learning Rate, Increase the Batch Size. [arXiv:1711.00489]()

WHAT CAN WE DO ABOUT IT?

LARS: Layer-wise Adaptive Rate Scaling



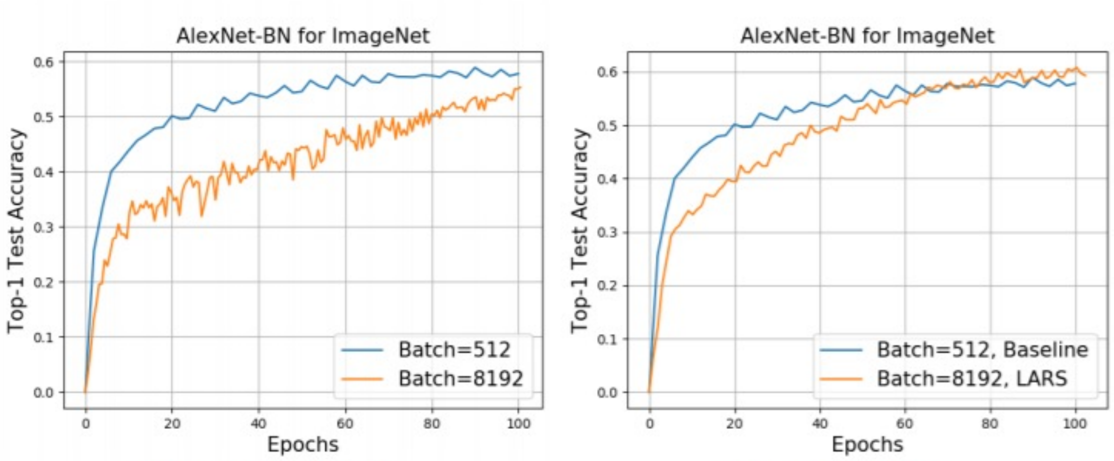
(a) Local LR, conv1-weights

(b) Local LR, conv1-bias

(c) Local LR , conv5-weights

(d) Local LR, conv5-bias

Figure 2: LARS: local LR for different layers and batch sizes



(a) Training without LARS

(b) Training with LARS

You, Y., Gitman, I., & Ginsburg, B. Large batch training of convolutional networks. arXiv:1708.03888

NVIDIA DEEP LEARNING INSTITUTE

WHAT CAN WE DO ABOUT IT?

LARS: Layer-wise Adaptive Rate Scaling

Control magnitude of the layer *k* update through local learning rate λ_k:

$$\Delta\, w_k(t+1) = \lambda_k * G_k(w(t))$$

where:

$G_k(w(t))$: stochastic gradient of L with respect to w_k,

λ_k: local learning rate for layer k, defined as

$$\lambda_k = min(\gamma,\ \eta \cdot \frac{||w_k(t)||_2}{||G_k(w(t))||_2})$$

where

η is trust coefficient (how much we trust stochastic gradient)

γ is global learning rate policy (steps, exponential decay, …)

You, Y., Gitman, I., & Ginsburg, B. Large batch training of convolutional networks. arXiv:1708.03888

WHAT CAN WE DO ABOUT IT?

LARC: Layer-wise learning rates with clipping; SGD with momentum is base optimizer

LAMB: Layer-wise learning rates; Adam as base optimizer

- More successful than LARC at language models like BERT

NovoGrad: Moving averages calculated on a per-layer basis

- Also useful in several different domains

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli

NVIDIA

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli