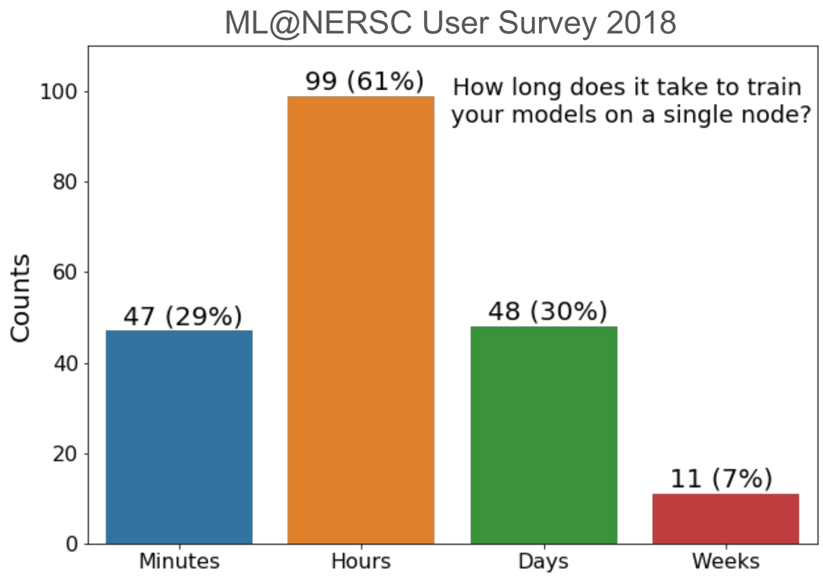
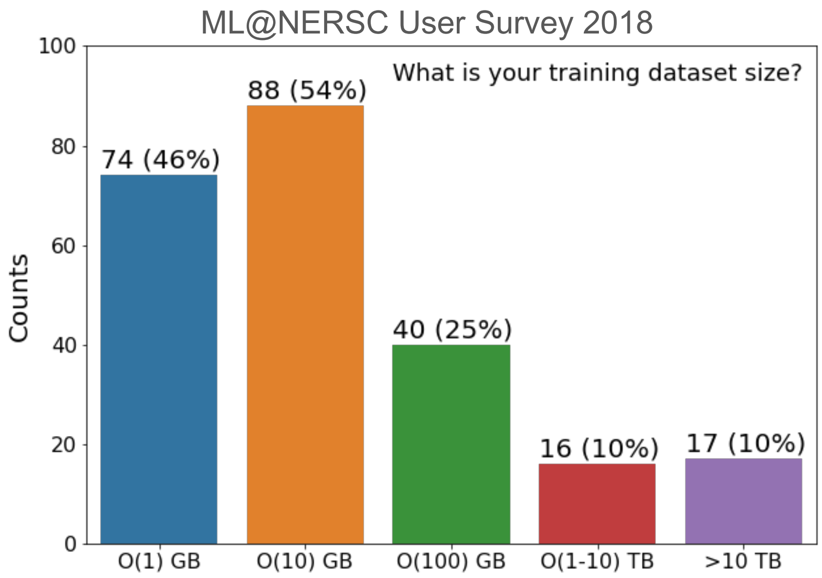
Scaling Deep Learning (on HPC)

Steve Farrell
NERSC, LBNL
(with material by Mustafa Mustafa)

Exa.TrX kickoff, 2019-06-04

Why do we need to scale deep learning applications?

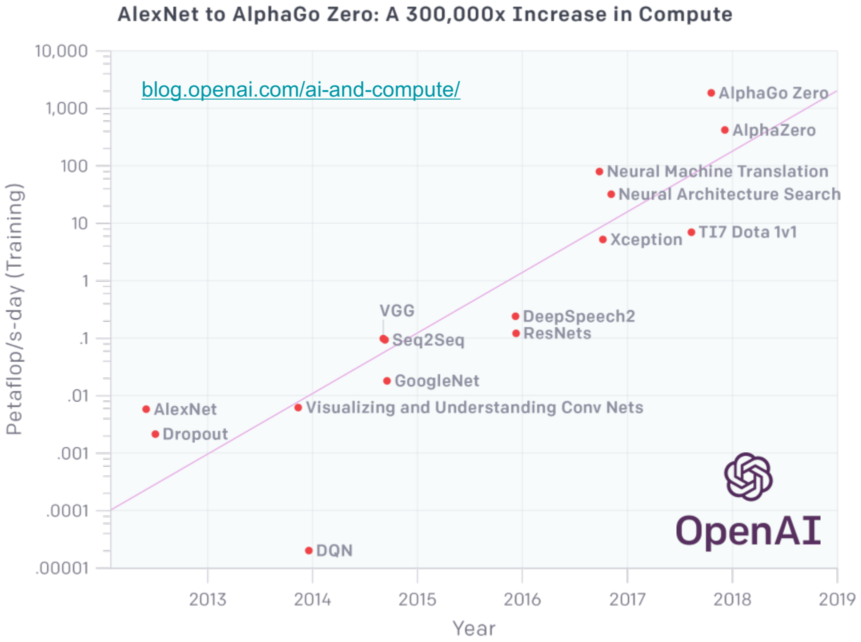




- Rapid prototyping/model evaluation
- Problem scale

- Volume of scientific datasets can be large
- Scientific datasets can be complex (multivariate, high dimensional)

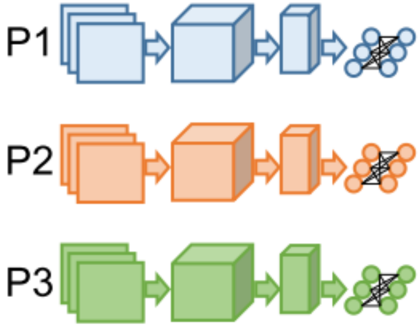
Why do we need to scale deep learning applications?



Models get bigger and more compute intensive as they tackle more complex tasks

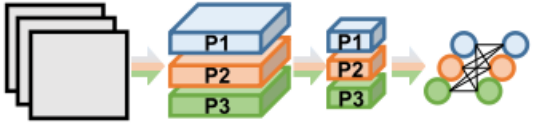
"... total amount of compute, in petaflop/s-days, that was used to train selected results ... A petaflop/s-day (pfs-day) = ... 10^{15} neural net operations per second for one day, or a total of about 10^{20} operations." -- OpenAI Blog

Parallelism strategies



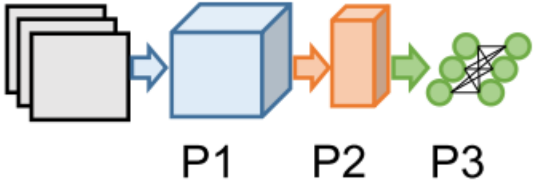
Data Parallelism

Distribute input samples.



Model Parallelism

Distribute network structure (layers).



Layer Pipelining

Partition by layer.

Data parallelism, synchronous Updates

Gradients are computed locally and summed across nodes. Updates are propagated to all nodes

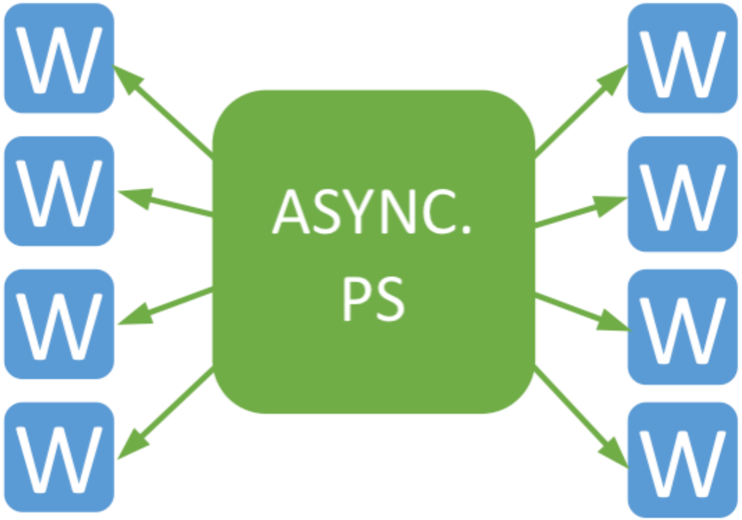
- stable convergence
- scaling is not optimal because all nodes have to wait for reduction to complete
- global (effective) batch size grows with number of nodes

Synchronous SGD, decentralized

Data parallelism, asynchronous Updates

Gradients are sent to parameters server. Parameters servers incorporates gradients into model as they arrive and sends back the updated model

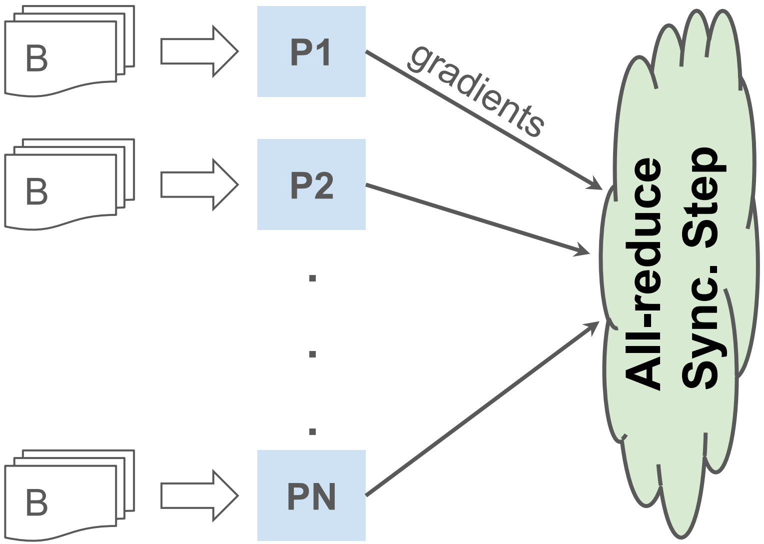
- nodes don’t wait (perfect scaling)
- resilient
- stale gradients impact convergence rate (depends on #workers)
- parameter server is a bottleneck



Asynchronous SGD, parameter-server

Large-Batch Training (LBT), synchronous weak scaling

- applies to SGD-type algorithms
  - data batch per node. Model updates are computed independently
  - updates are collectively summed and applied to the local model

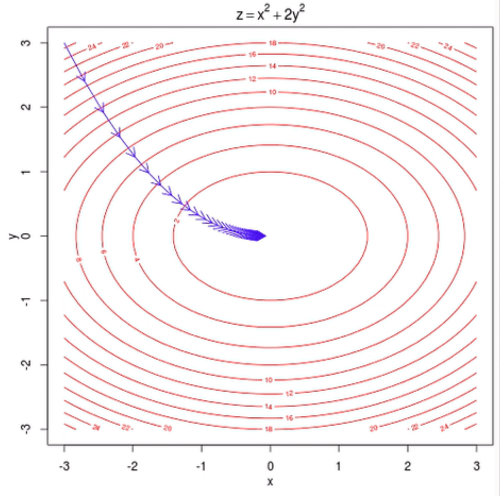


Local batch-size = B

Global batch-size = N * B

Stochastic Gradient Descent (SGD)

$$w_{t+1} \leftarrow w_t - \frac{\eta}{B}\sum_{i=1}^{B} \nabla L(x_i, w_t)$$



N is total sample size

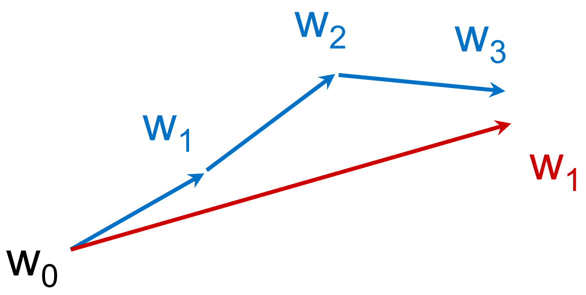
B is batch-size

η is learning rate

Δw is the parameter update in one gradient descent step

Linear learning-rate scaling

$$\eta \rightarrow N * \eta$$



Upper: 3 SGD steps w. learning-rate = η
Lower: 1 SGD step w. learning-rate = 3 * η

Sqrt learning-rate scaling

η → sqrt(N) * η

Motivated by the observation that the variance of the gradient scales with 1/batch-size:

$$\operatorname{cov}(\Delta w, \Delta w) \approx \frac{\eta^2}{B}\left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{g_i}\mathbf{g_i^T}\right)$$

Learning-rate scaling

In practice, we see anywhere between sub-sqrt (e.g.You et al. arXiv:1708.03888) to linear scaling (e.g. Goyal et al. arXiv:1706.02677)

Recent OpenAI (arXiv:1812.06162) study has illuminated the dependence of optimal learning-rate on batchsize:

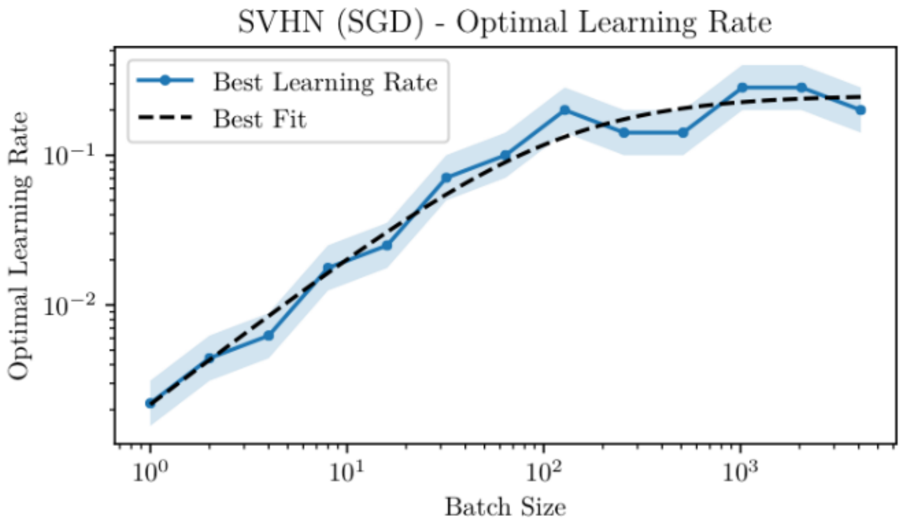
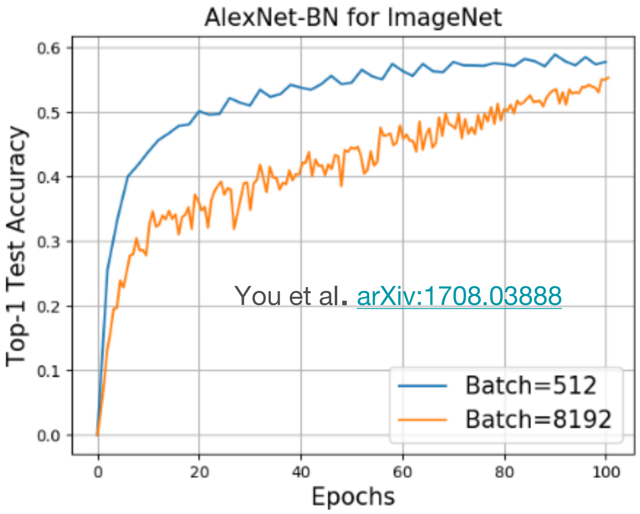


Fig. McCandlish, Kaplan and Amodei arXiv:1812.06162

Challenges with Large Batch Training

- Training with large learning rates is not stable in the initial stages of the training
  $\nabla L(w_{t+1}) \approx \nabla L(w_t)$ assumption breaks when parameters are changing rapidly
- A generalization gap appears: networks trained with small batches tend to optimize and generalize better

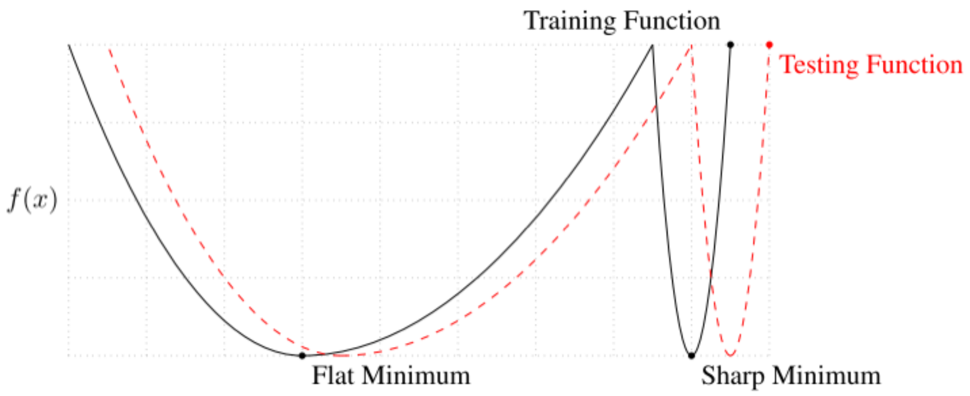


AlexNet You et al. arXiv:1708.03888

Batch	Base LR	accuracy,%
512	**0.02**	**60.2**
4096	0.16	58.1
4096	**0.18**	**58.9**
4096	0.21	58.5
4096	0.30	57.1
8192	0.23	57.6
8192	**0.30**	**58.0**
8192	0.32	57.7
8192	0.41	56.5

Explaining the generalization gap?

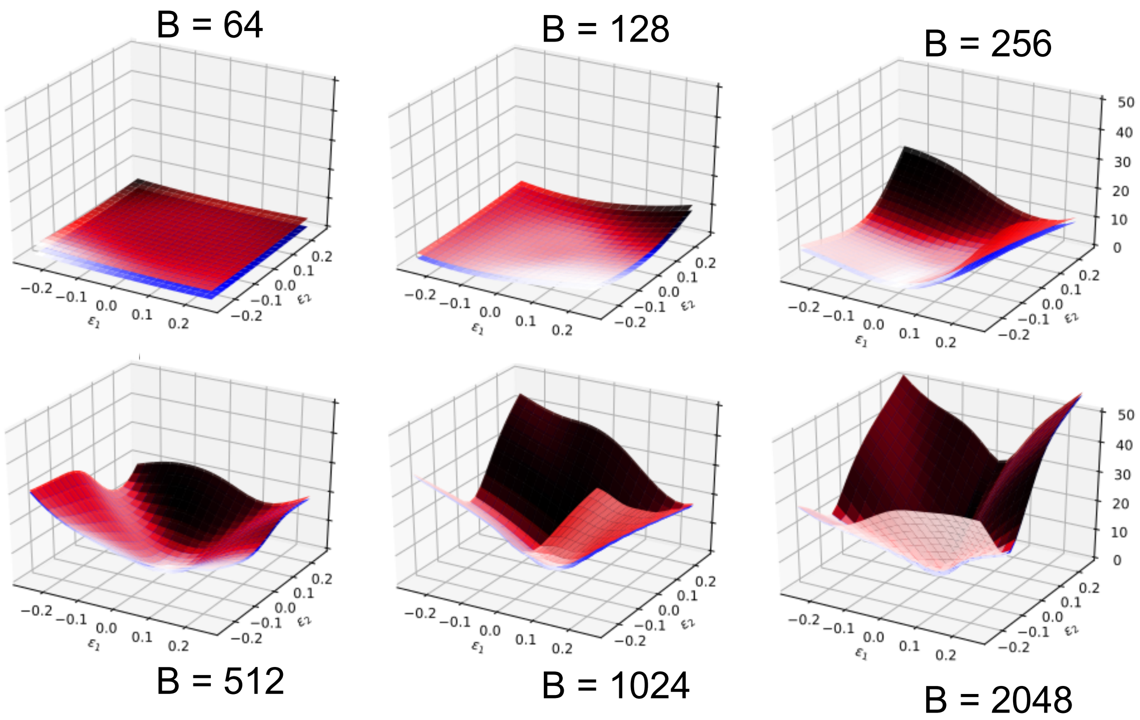
"*... large-batch … converge to sharp minimizers of the training function … In contrast, small-batch methods converge to flat minimizers*" -- Keskar et al, arXiv:1609.04836



Conceptual sketch of sharp and flat minimas of a loss function

Fig. credit: Keskar et al, arXiv:1609.04836

Explaining the generalization gap?



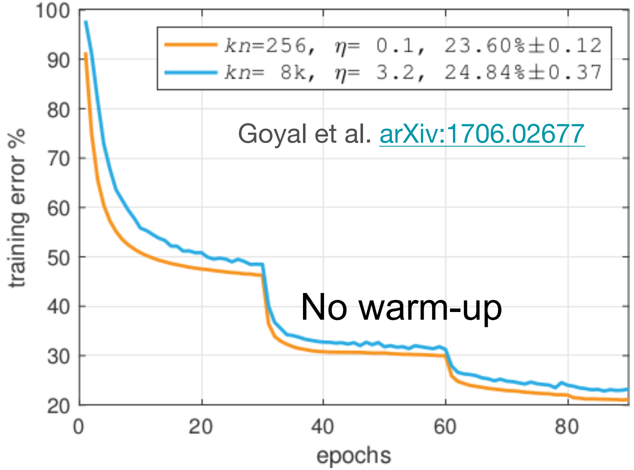
Loss at the end of training CIFAR-10 (axes are dominant eigenvectors of the Hessian)

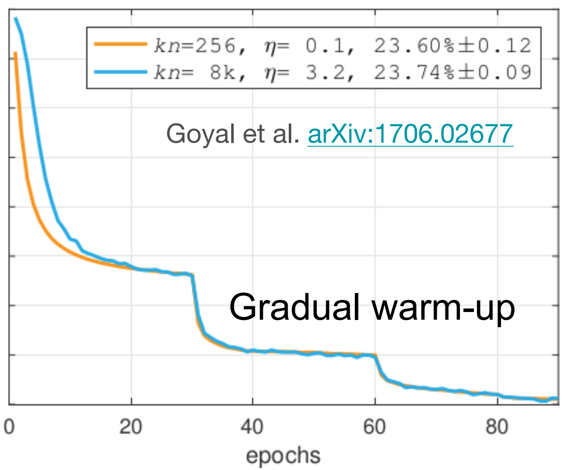
Z. Yao et al. arXiv:1802.08241

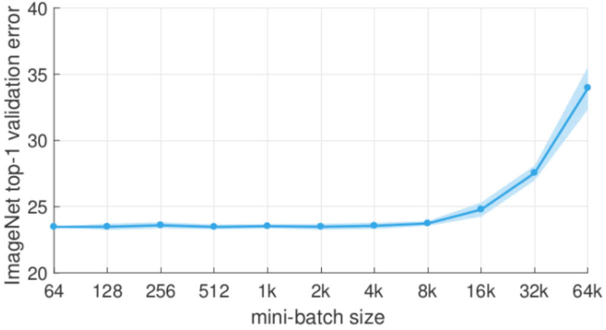
ResNet-50 ImageNet in 1 hour

FaceBook scaling result in 2017, batch-size=8k (using 256 GPUs):

- **Linear learning-rate warm-up** over 5 epochs to target rate
- **Linear scaling of learning-rate** (N * η) followed by original decay schedule
- The paper also clarifies subtleties and common pitfalls in distributed training





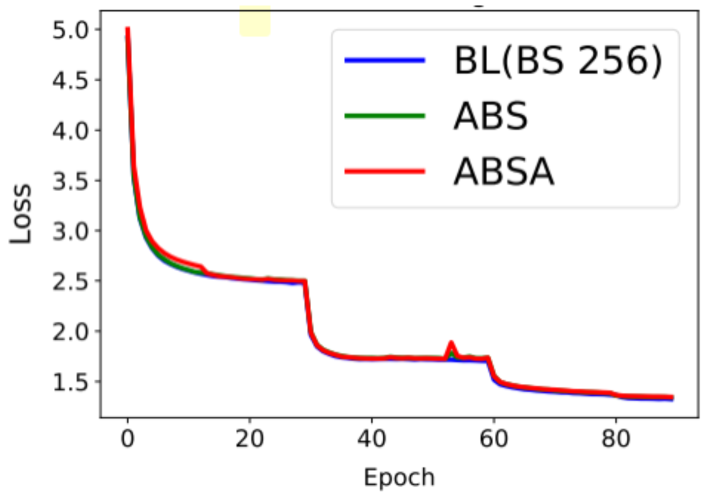


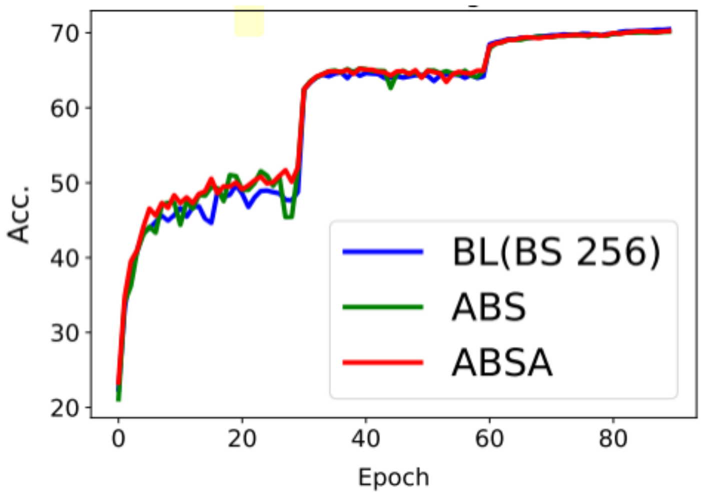
This scheme breaks down beyond batch-size = 8k for ResNet on ImageNet

Adaptive batch-size scaling with 2nd-order information (ABSA)

Z. Yao et al. arXiv:1810.01021 close the generalization gap for a wide range of architectures on image classification tasks, using

- 2nd-order info. (~ loss surface curvature) to adaptively increase the batch-size
- adversarial training to regularize against “sharp-minima”





ABS and ABSA with ResNet-18 on ImageNet dataset with up to 16k batch-size

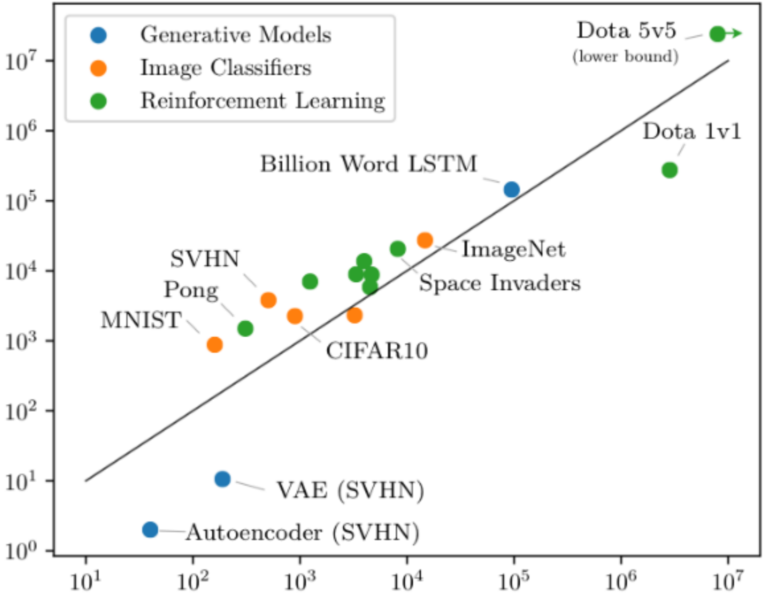
Limits of batch-size scaling

Recent empirical studies by OpenAI (arXiv:1812.06162) and Google Brain (arXiv:1811.03600) show that:

- A relationship between gradient noise scale and “critical” batch-size holds across many models, algorithms and datasets
- ”gradient noise scale predicts maximum useful batch-size”
- More complex datasets/tasks have higher gradient noise, thus can benefit from training with larger batch-sizes

McCandlish, Kaplan and Amodei arXiv:1812.06162

Gradient noise scale measures the variation of the gradients between different training examples



Critical batch size is the maximum batch size above which scaling efficiency decreases significantly

Scaling DL outlook

- Distributed training is imperative for larger and more complex models/datasets
- Data parallelism distributes more data among more workers
- Large batch training is unstable and may impact generalization error if hyper-parameters are not “tuned” well
- Use learning-warm up and linear scaling to scale to modest scales < 10x. No guarantees that it will work for all models
- Batch-size scaling seems to be more robust across many models
- A simple statistic, gradient noise scale, can predict maximum useful batch-size

Practical stuff
(systems, software, examples)

Software for Deep Learning

Several popular Deep Learning frameworks

- TensorFlow, Keras, PyTorch, …

Backed by hardware-optimized libraries

- MKL-DNN, cuDNN

Scalable distributed training libraries

- Horovod, Cray ML Plugin, Mesh-TensorFlow, MPI-Learn

Support for various kinds of hardware

- CPUs, GPUs, TPUs, FPGAs, other custom chips

Supercomputers

Big machines with diverse, heterogeneous, high-performance hardware

- Big FLOPS
- High-speed interconnects
- High-bandwidth file systems

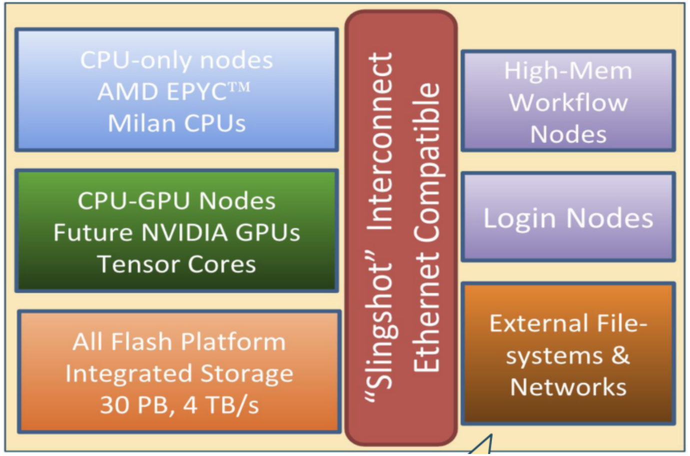
https://www.top500.org/

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384

- **Traditionally support large, parallel "simulation" applications**
- **Growing interest and support for data analytics and ML workloads**

The Perlmutter supercomputer

- **Next-gen NERSC system optimized for science**
- **First Cray Shasta system**
- **GPU-accelerated (4x NVIDIA) nodes and CPU-only (AMD) nodes**
- **Cray Slingshot high performance network**
- **Single-tier All-Flash Lustre based file system**



Delivered late 2020

https://www.nersc.gov/systems/perlmutter/

NERSC

Distributed training tutorial

Latest code version:

https://github.com/sparticlesteve/sea19-dl-tutorial

Has basic CNN single-node example on CIFAR10 dataset

Uses ResNet on CIFAR10 to demonstrate distributed training

Uses Keras and Horovod for distributed training

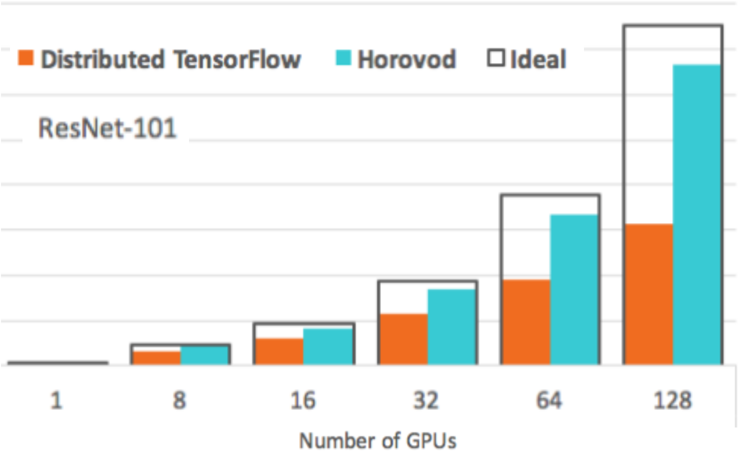
- Easiest to use/teach

Horovod

Enables distributed synchronous data-parallel training with minimal changes to user code

Uses ring all-reduce and MPI to collectively combine gradients across workers

Such approaches shown to scale better than parameter-server approaches (e.g. distributed TensorFlow with gRPC)



https://eng.uber.com/horovod/

NERSC

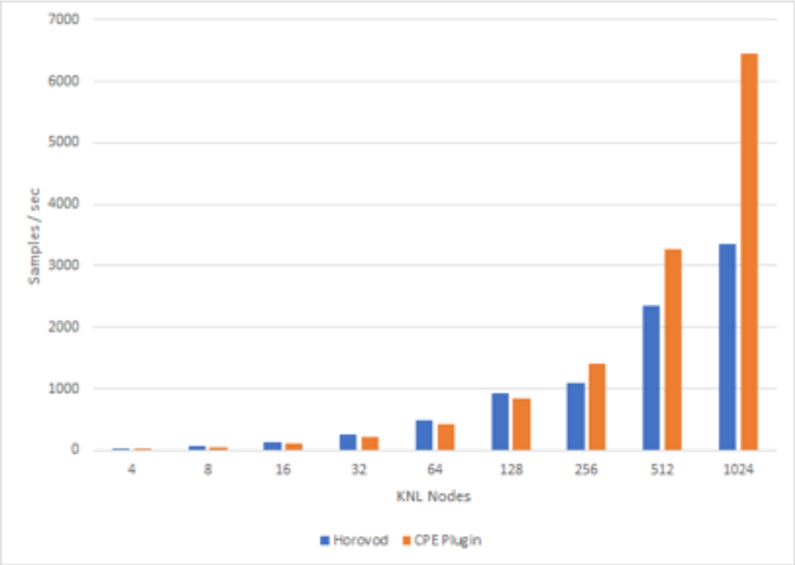
Cray ML Plugin

Enables distributed synchronous data-parallel training with minimal changes to user code

Uses RDMA operations or reductions

Might perform better than Horovod on large networks and large scales

Includes some advanced features for performance: gradient lag, hiding communication



NERSC CosmoFlow

Ingredients for multi-node training with Horovod

Initialize Horovod and MPI:

```
hvd.init()
```

Wrap your optimizer in the Horovod distributed optimizer:

```
opt = keras.optimizers.SGD(lr=lr*hvd.size(), ...)
opt = hvd.DistributedOptimizer(opt)
```

Linearly scaling the learning rate

Construct the variables broadcast callback:

```
callbacks = [
    hvd.callbacks.BroadcastGlobalVariablesCallback(0)
]
```

Ingredients for multi-node training

Train model as usual; it should now synchronize at every mini-batch step:

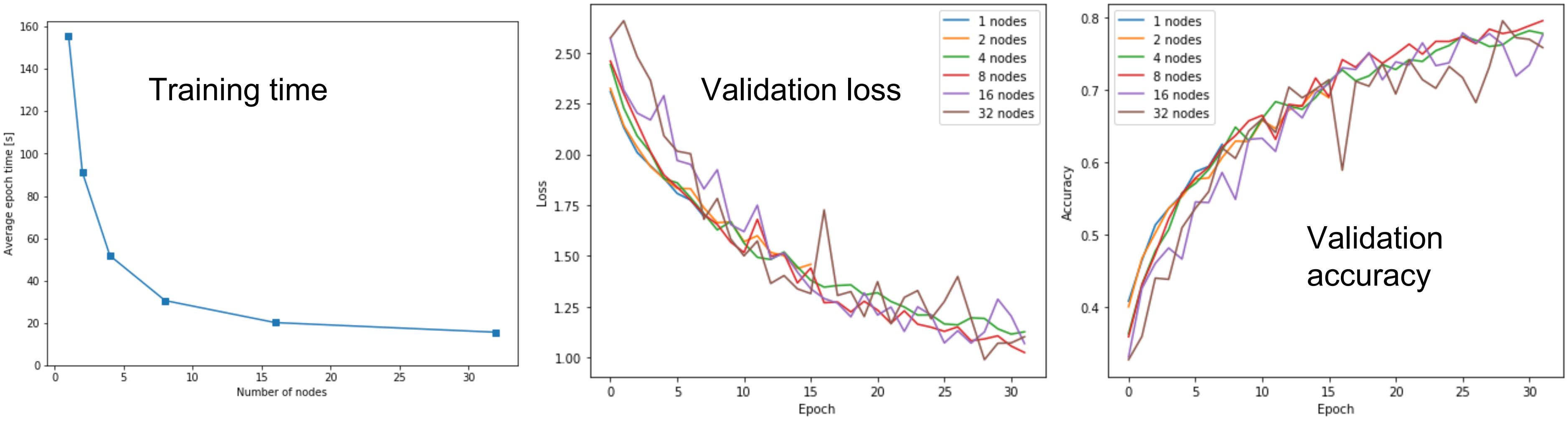
```
model.fit(..., callbacks=callbacks)
```

Launch your script with MPI

```
mpirun -n NUM_RANKS … python train.py …
```

(we’ll use SLURM and srun instead of mpirun)

Scaling results for ResNet CIFAR10



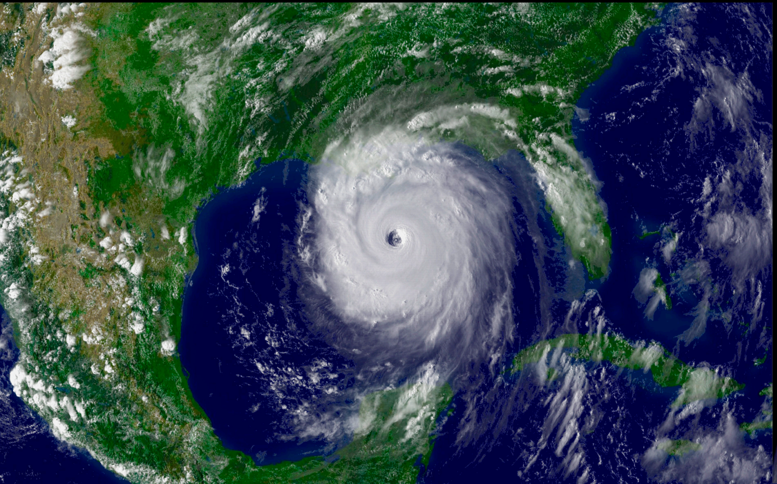
Training time goes down

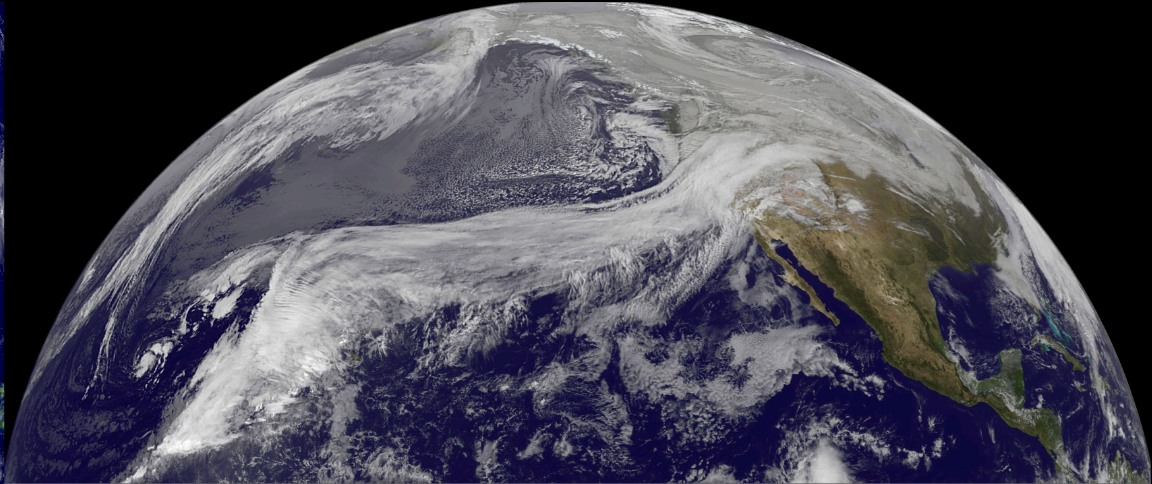
Training loss and accuracy are still converging at similar rates

Science examples
(climate, cosmology)

Deep Learning for climate analytics

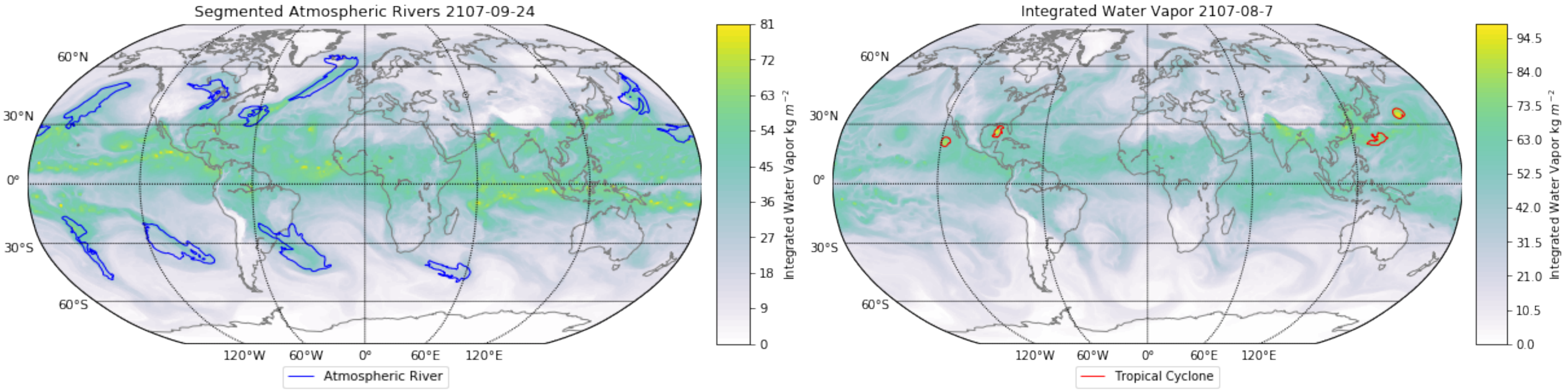
- **Global warming increases rate of extreme weather phenomena**
- **We want to make quantifiable predictions about these effects**
- **Need to identify these phenomena in simulated climate data**
  - Heuristic labeling algorithms are imperfect
  - Hand labeling is too tedious





Segmentation

https://arxiv.org/abs/1810.01993



Collaboration between NERSC, NVIDIA, UCB, OLCF

Pixel-level classification of extreme weather phenomena

- 3 classes: atmospheric river, tropical cyclone, background
- High class imbalance, mostly background

Labels acquired via heuristic algorithms

Segmentation

https://arxiv.org/abs/1810.01993

Architectures

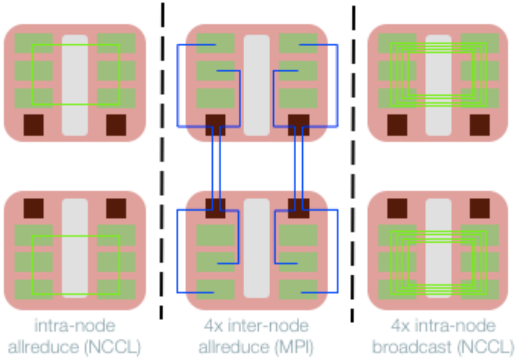
- Modified Tiramisu and DeepLabV3+

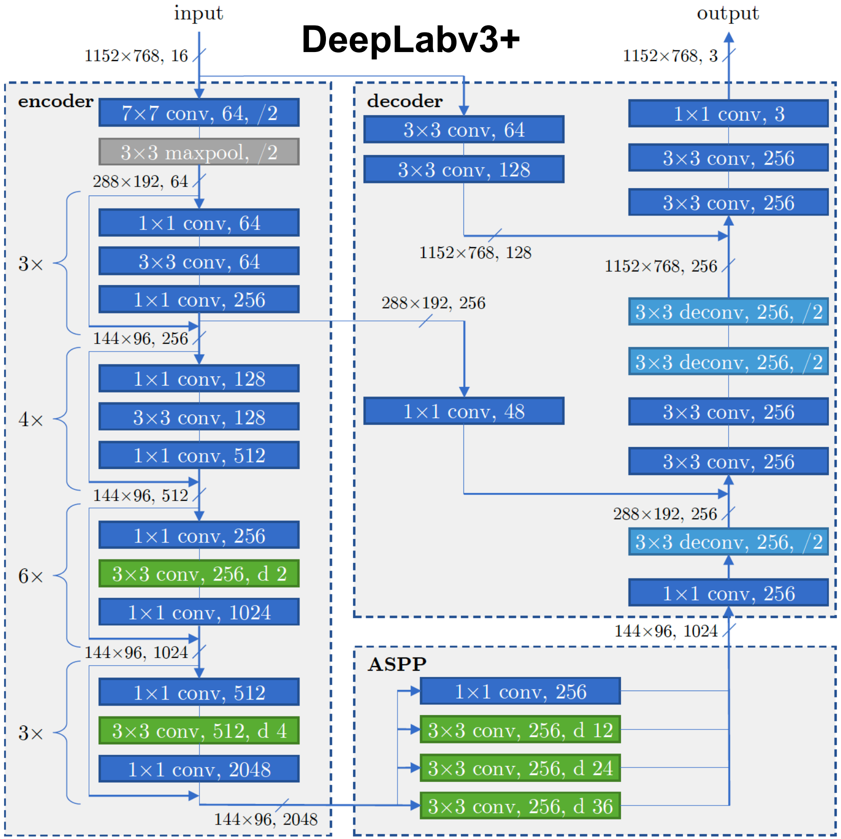
Software

- TensorFlow, Horovod

Customizations for scaling

- gradient lag, hybrid all-reduce, hierarchical control plane

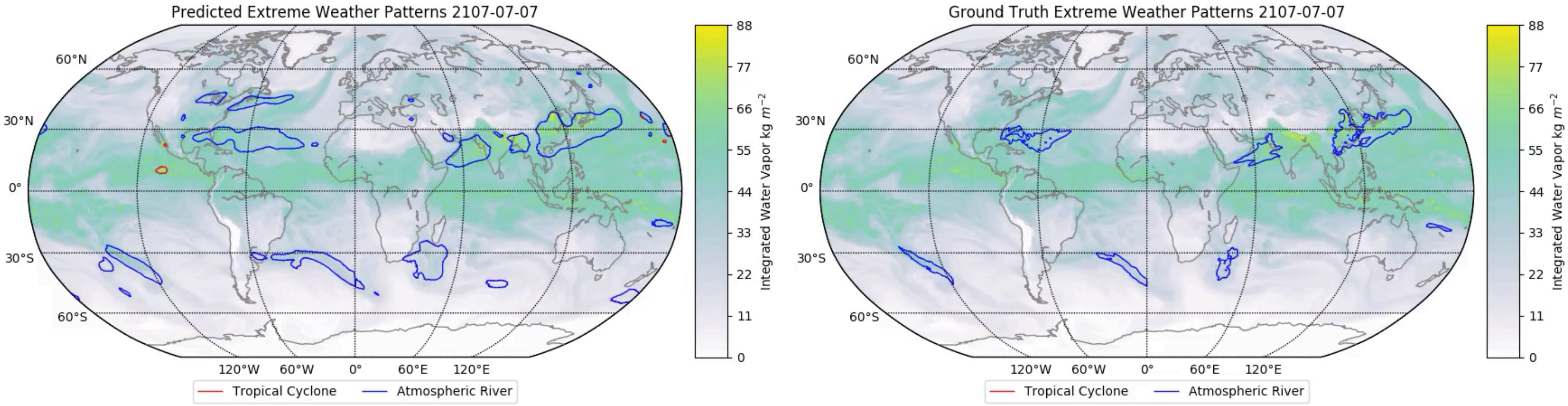




NERSC

Segmentation results

https://arxiv.org/abs/1810.01993



Best result for intersection-over-union (IoU) obtained: ~73%
Deep learning results are smoother than heuristic labels

Scaling performance

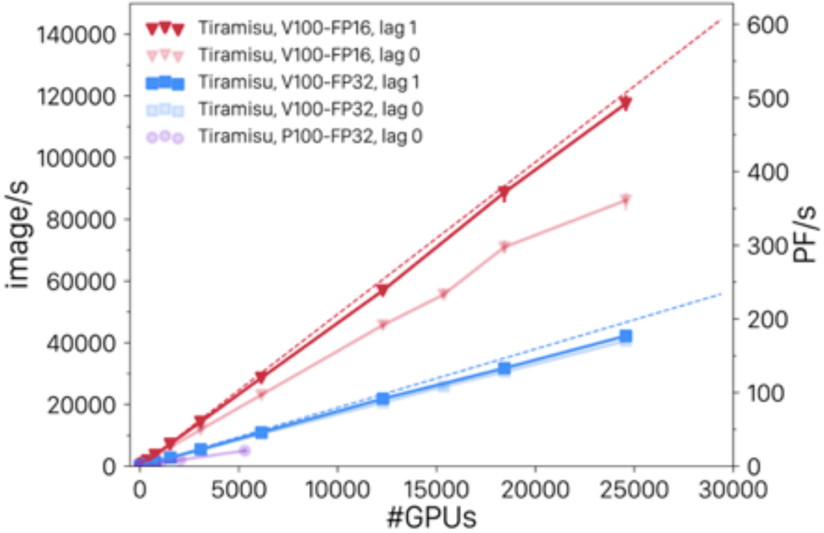
Excellent scaling on Summit

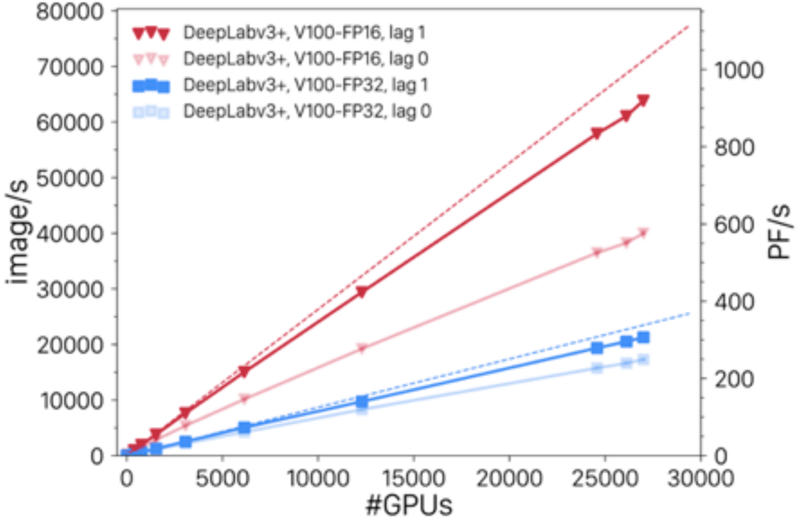
With DeepLabV3+ on all 27,369 GPUs

- **999 PetaFlop/s (FP16) sustained**
- **1.13 ExaFlop/s (FP16) peak**

Shared the 2018 ACM Gordon Bell Prize!

https://arxiv.org/abs/1810.01993

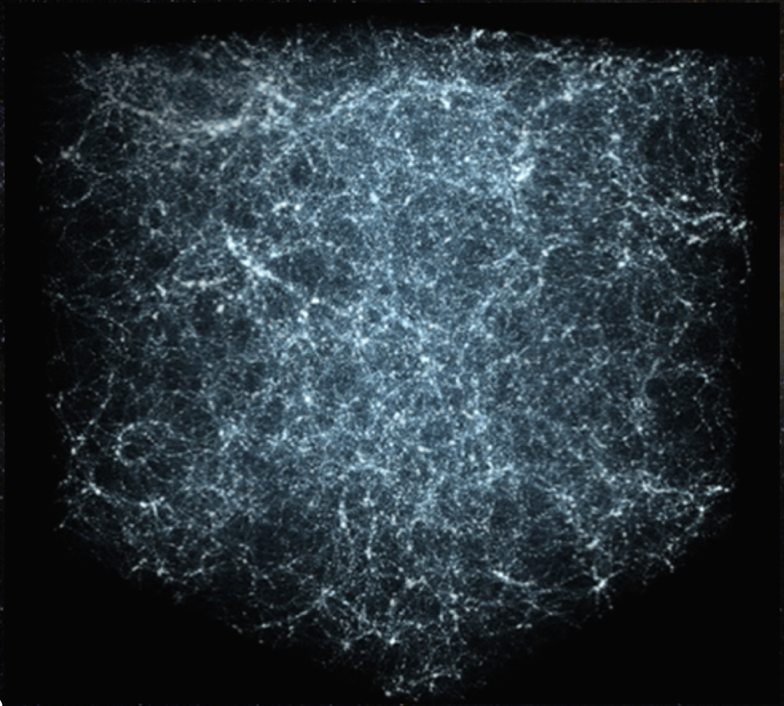




Deep Learning for cosmology

Cosmology seeks to answer questions about

- The nature of dark matter
- The nature of dark energy
- The inflation of the early universe



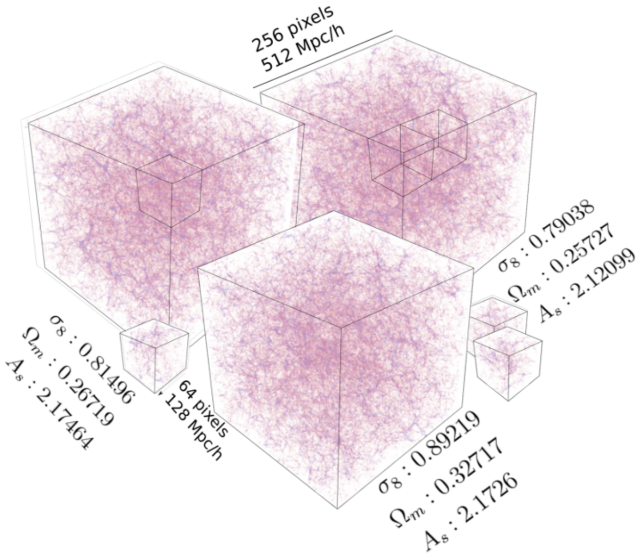
The answers are encoded in the structure of the universe

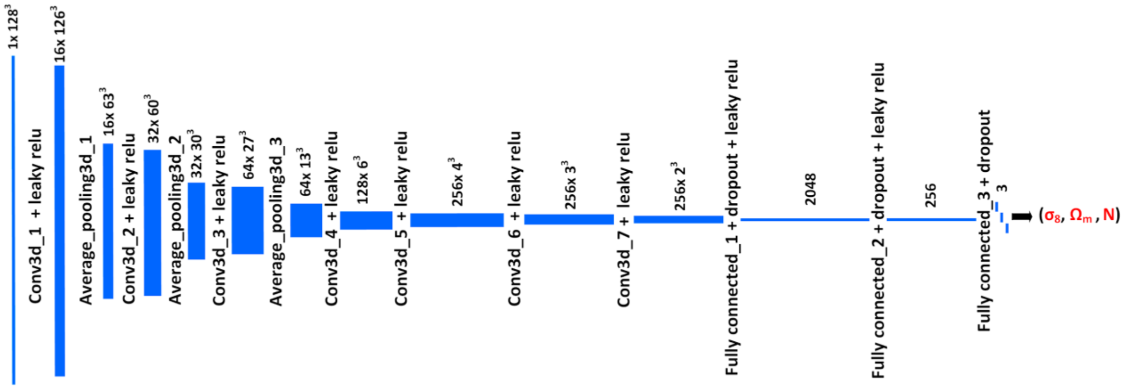
- How can Deep Learning help?

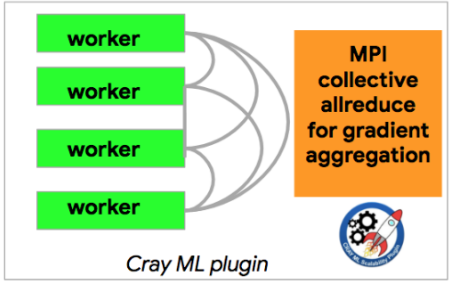
CosmoFlow

Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He, Tuomas Karna, Daina Moise, Simon J. Pennycook, Kristyn Maschoff, Jason Sewall, Nalini Kumar, Shirley Ho, Mike Ringenburg, Prabhat, Victor Lee

- **Collaboration between NERSC, Cray, Intel**
- **Predicting cosmological parameters from 3D voxels of Dark Matter simulations**
- **Uses TensorFlow and Cray PE ML Plugin for scalable distributed training**







[arXiv:1808.04728]

CosmoFlow results

Successful prediction of 3 cosmological parameters

- Comparable to experimental uncertainties for Ω_m and σ_8, almost 5x better for N_s.

Scaled to 3.5PF on Cori with 8k KNL nodes

- Convergence issues at scale for further study