Efficient Model Selection for Deep Neural Networks on Massively Parallel Processing Databases

Frank McQuillan
Feb 2020

FOSDEM'20

Nikhil Kak

Ekta Khanna

Orhan Kislal

Domino Valdano

Pivotal + vmware

Professor
Arun Kumar

Yuhao Zhang

UC San Diego

Free and Open Source

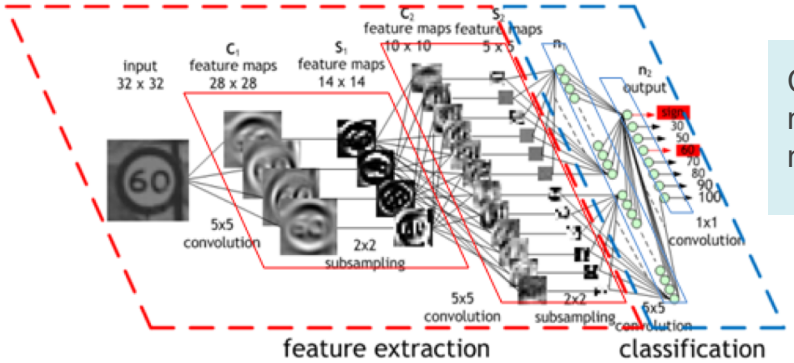
Agenda

1. Training deep neural networks in parallel
2. Model hopper parallelism
3. Model hopper on MPP
4. Examples
   - Grid search
   - AutoML (Hyperband)

Deep Learning

- Type of machine learning inspired by biology of the brain

- Artificial neural network have multiple layers between input and output



Convolutional neural network

Problem: Training deep nets is painful

Model accuracy = f(model architecture, hyperparameters, datasets)

Trial and error

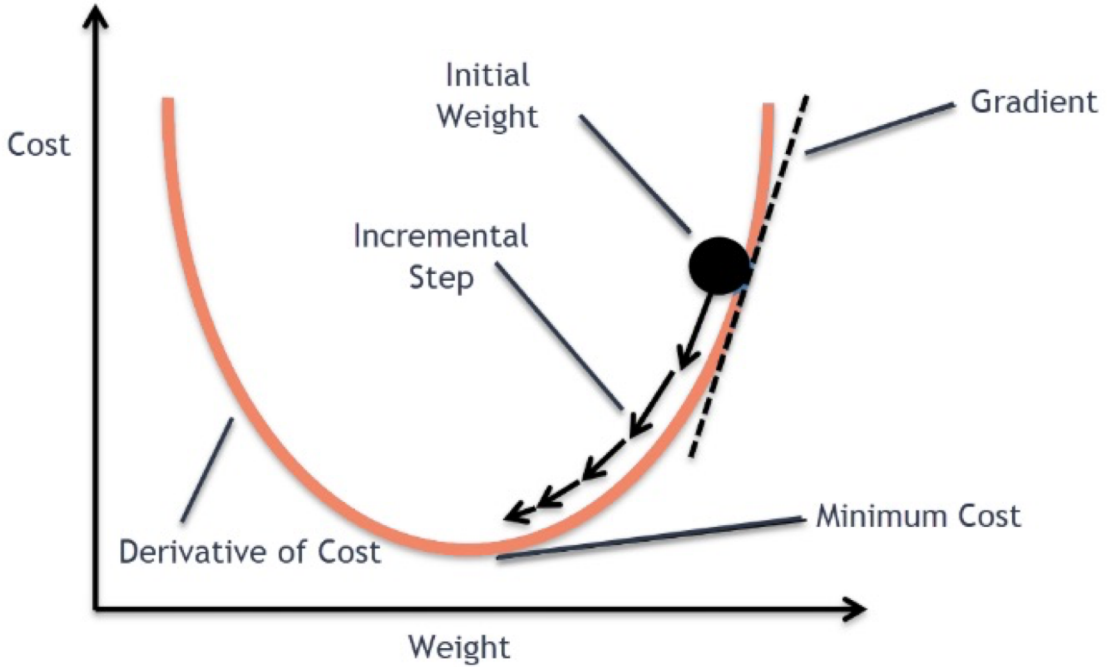
Model architecture	Batch size	Learning rate	Regularization
3 layer CNN, 5 layer CNN, LSTM …	8, 16, 64, 256 ...	0.1, 0.01, 0.001, 0.0001 ...	L2, L1, dropout, batch norm ...
4	4	4	4

256 different configurations!

Need for speed → parallelism

1. Training Deep Neural Nets in Parallel

Gradient Descent



Source: https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1

Mini-batch SGD on a Single Machine

One mini-batch

x1	x2	y
1.1	2.3	0
0.9	1.6	1
0.6	1.3	1
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...

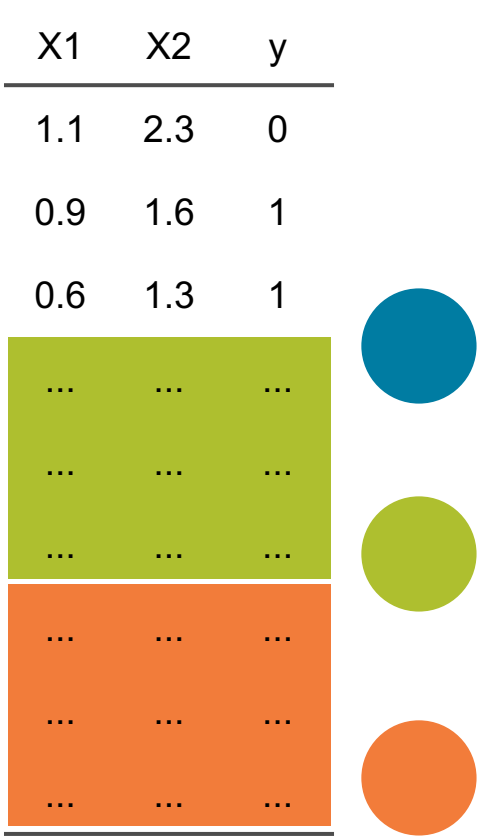
Model

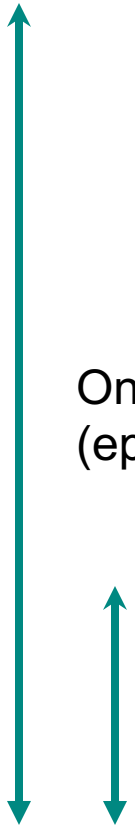
Updated Model $= -\eta \times \nabla$

Learning rate

Avg. of gradients

Mini-batch SGD on a Single Machine





One iteration
(epoch)

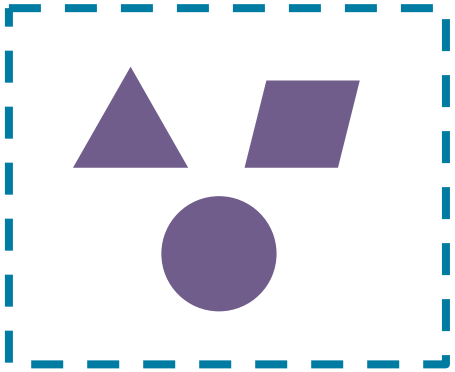
One minibatch

Sequential

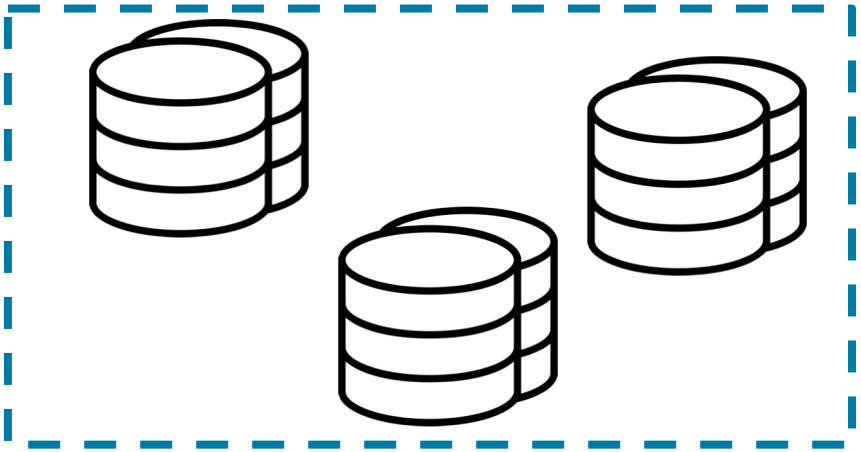
How to parallelize?

Task Parallelism

Train multiple models at the same time

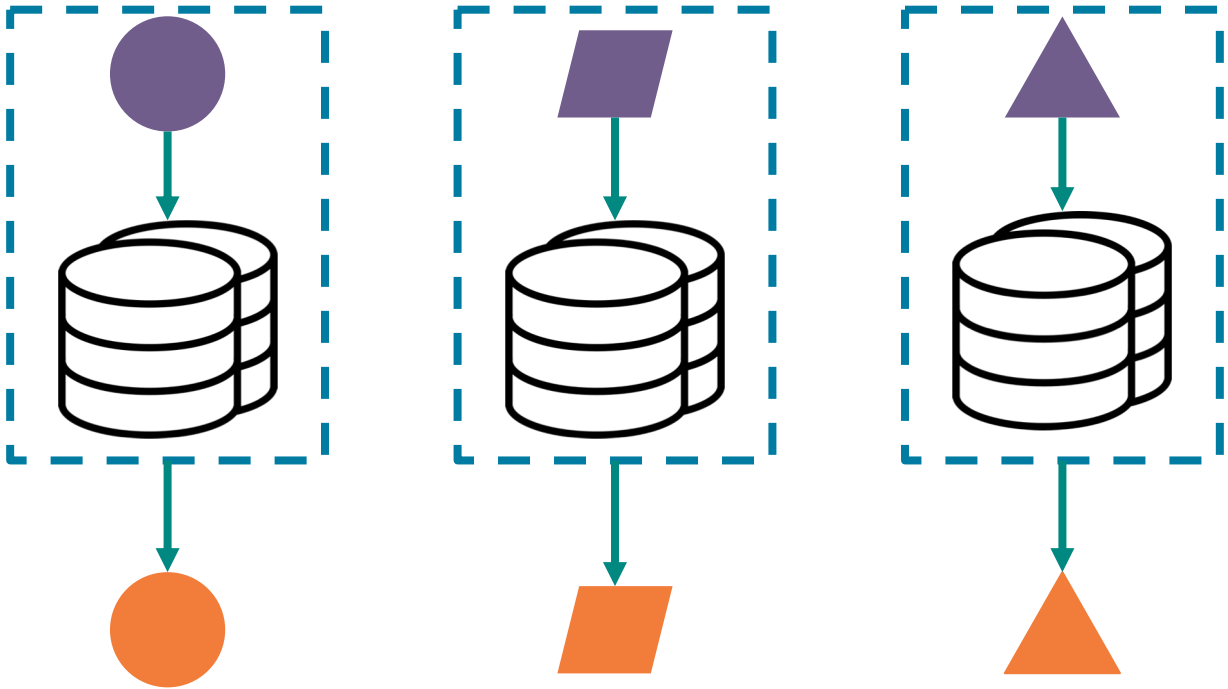


Models (tasks)



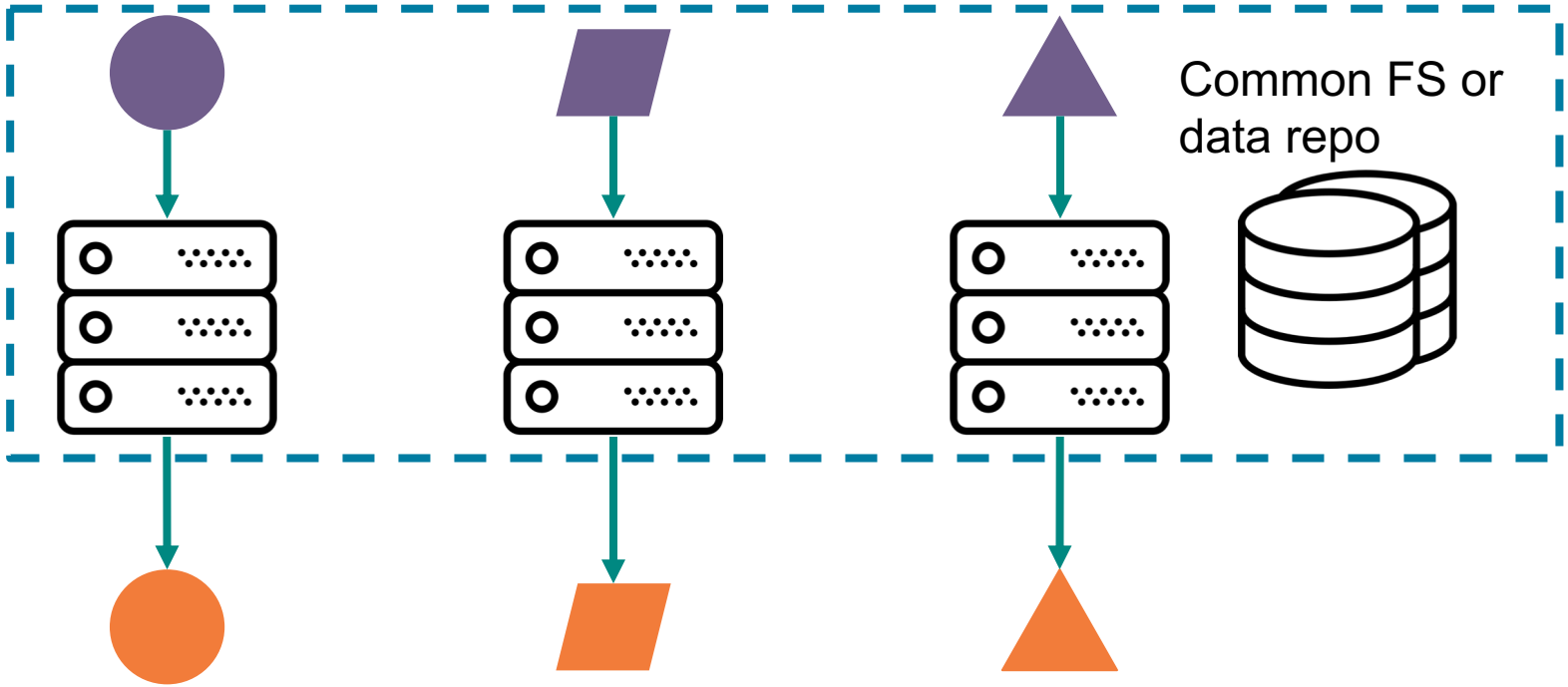
Replicate datasets on each machine

Task Parallelism



Con: wasted memory and storage

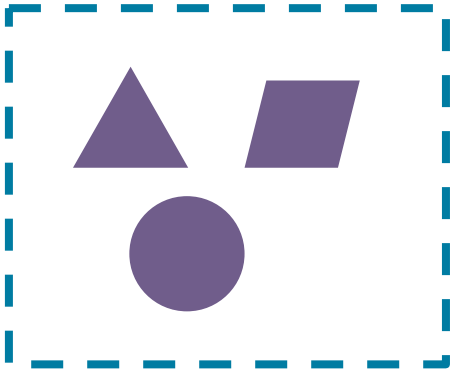
Task Parallelism



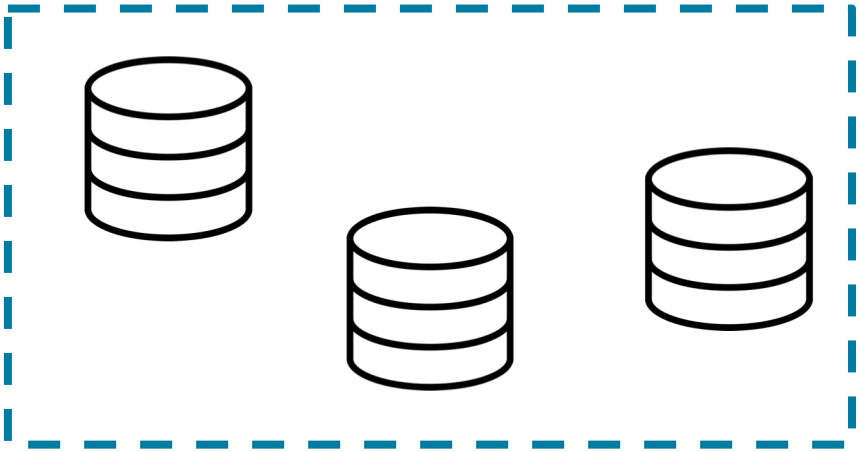
Con: wasted network bandwidth

Data Parallelism

Train models one at a time



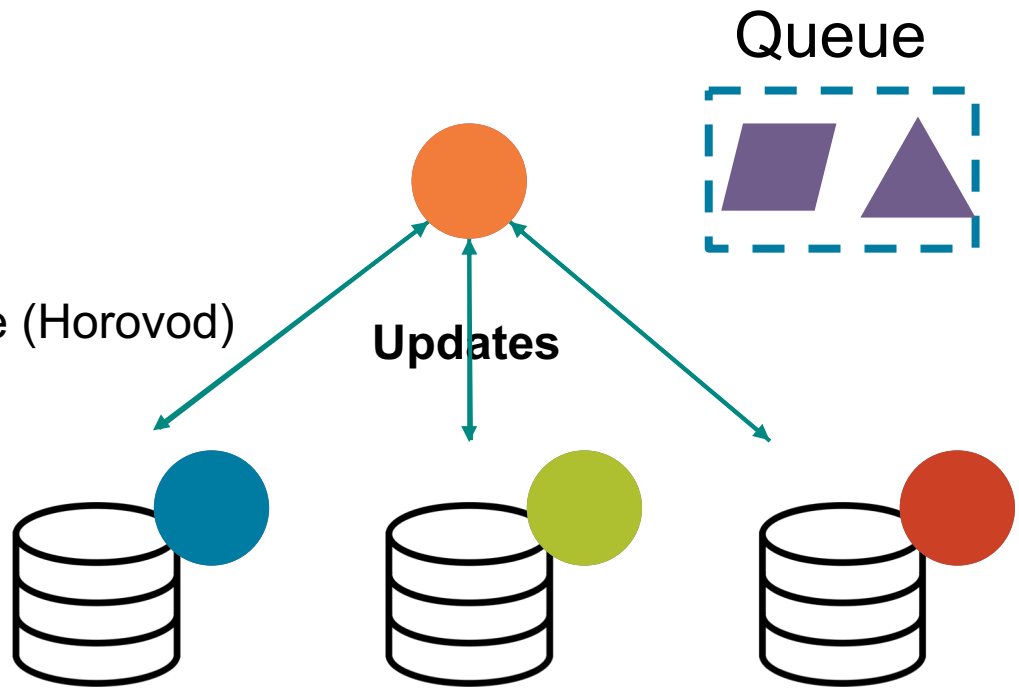
Models (tasks)



Partition data
across machines

Data Parallelism

- Update model every iteration: bulk synchronous parallelism (model averaging)
  - Convergence can be poor
- Update per mini-batch:
  - Sync parameter server
  - Async parameter server
  - Decentralized: MPI allreduce (Horovod)
  - High communication cost



Training on one mini-batch or full partition

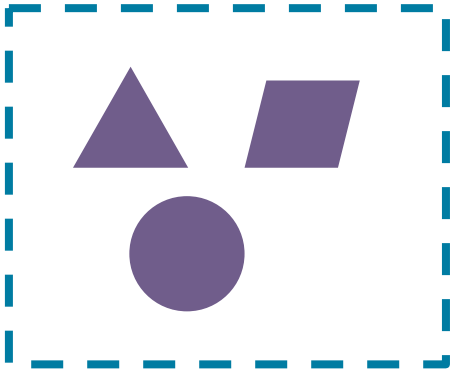
Let’s try to get the best of both worlds

Task Parallelism + **Data Parallelism**

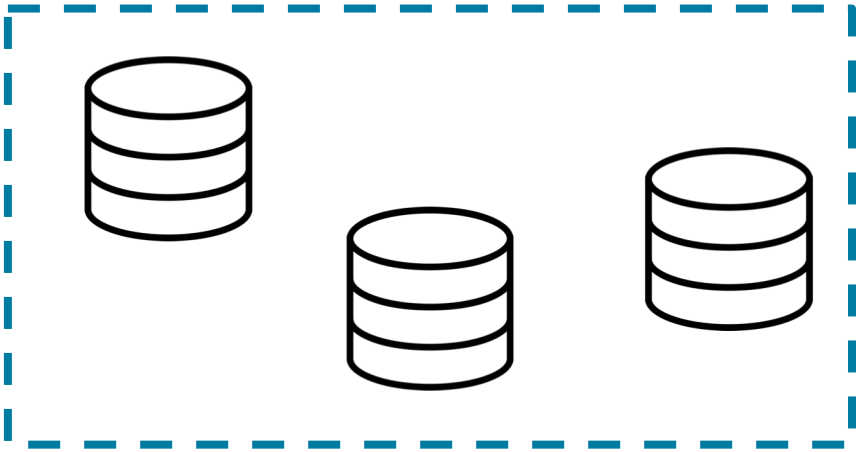
Model Hopper Parallelism

2. Model Hopper Parallelism

Model Hopper Parallelism



Models (tasks)

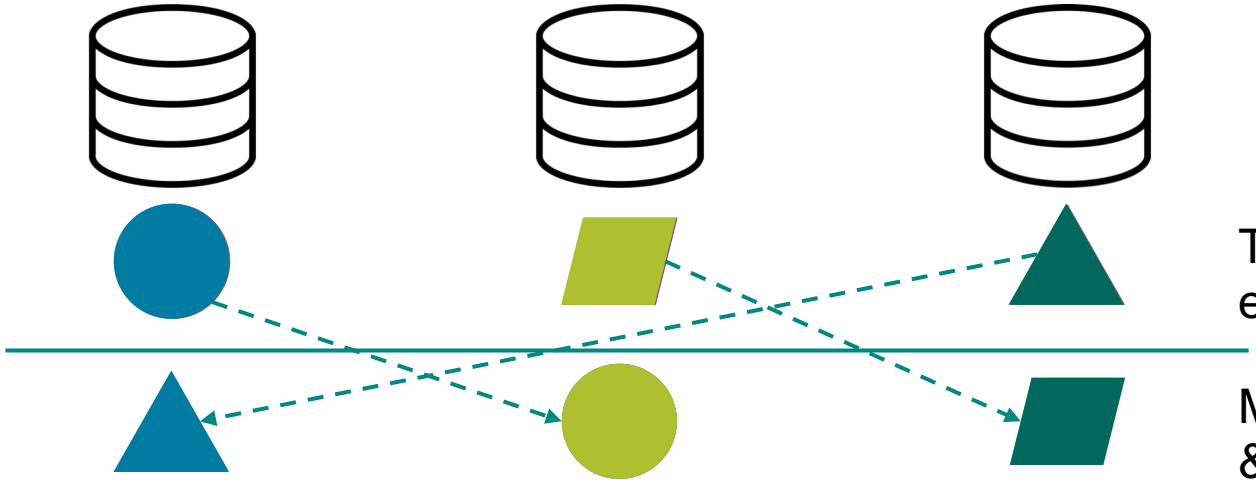


Partition data
across machines

Model Hopper Parallelism

Training on
entire partition

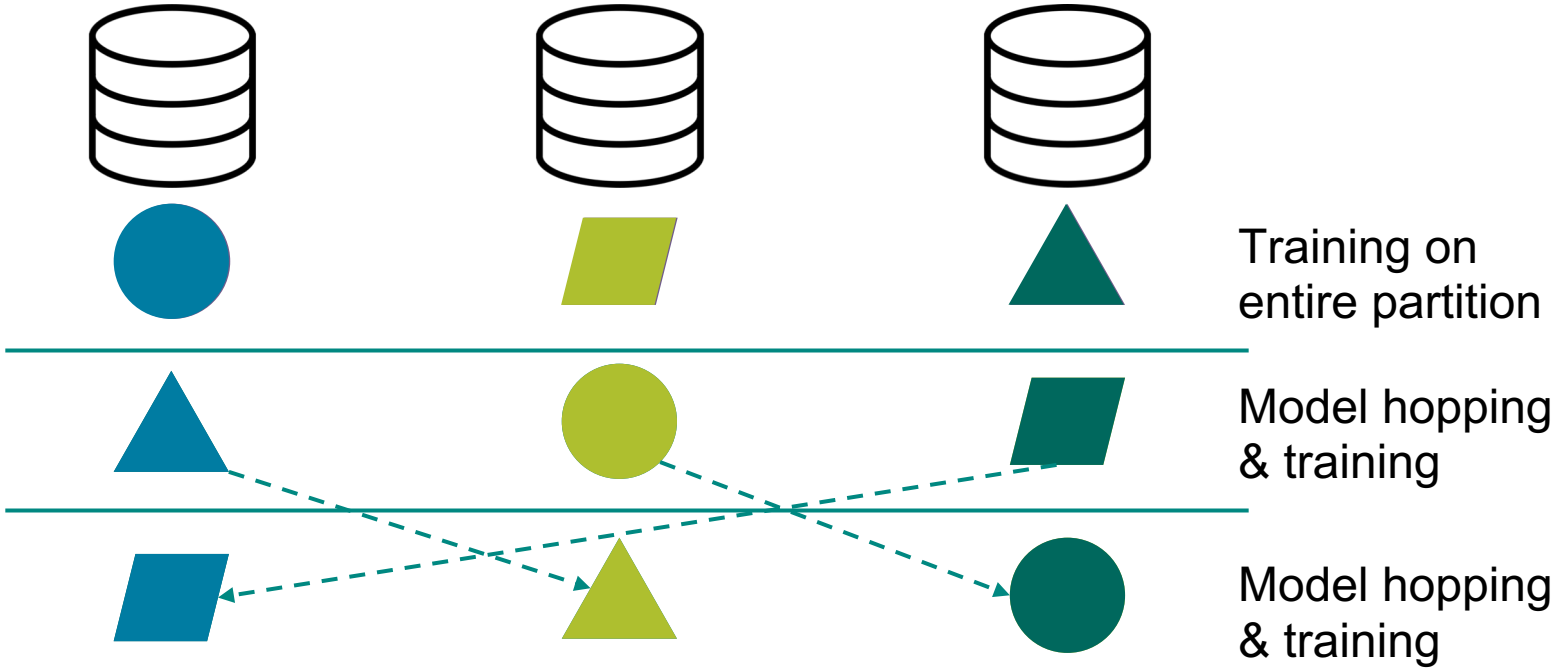
Model Hopper Parallelism



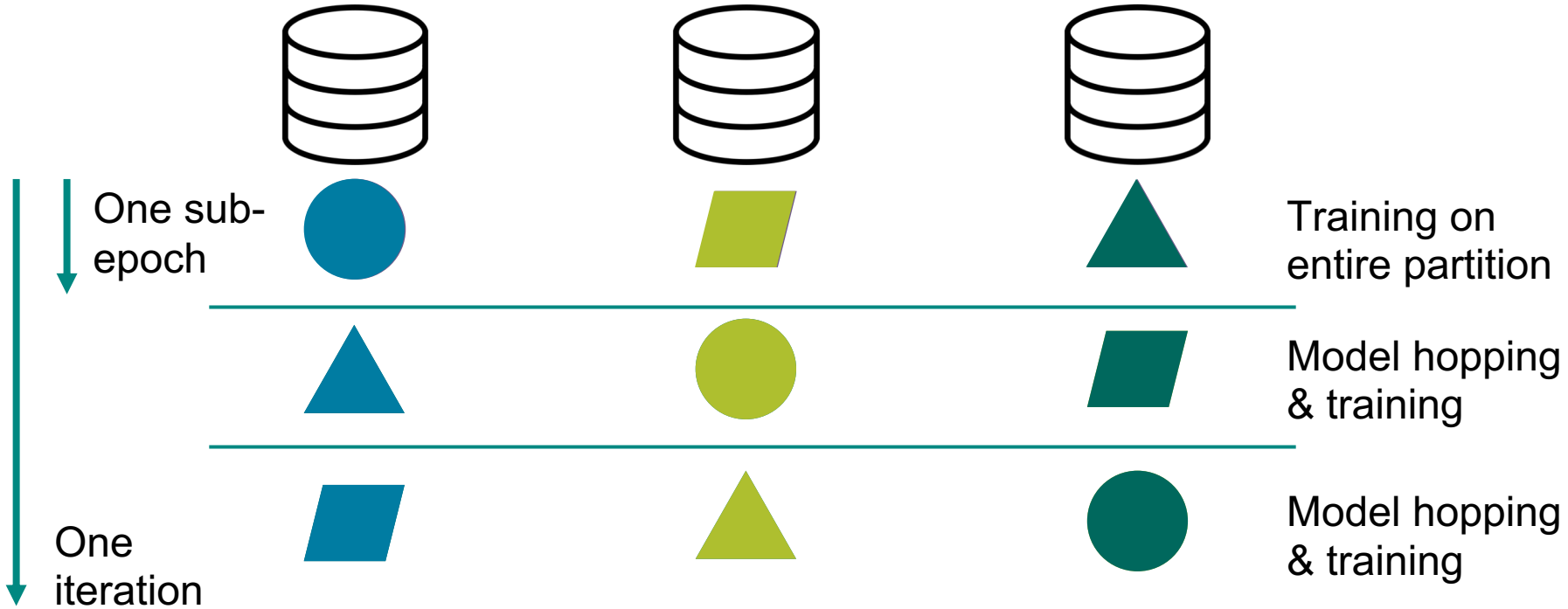
Training on entire partition

Model hopping & training

Model Hopper Parallelism



Model Hopper Parallelism



Model Hopper Parallelism

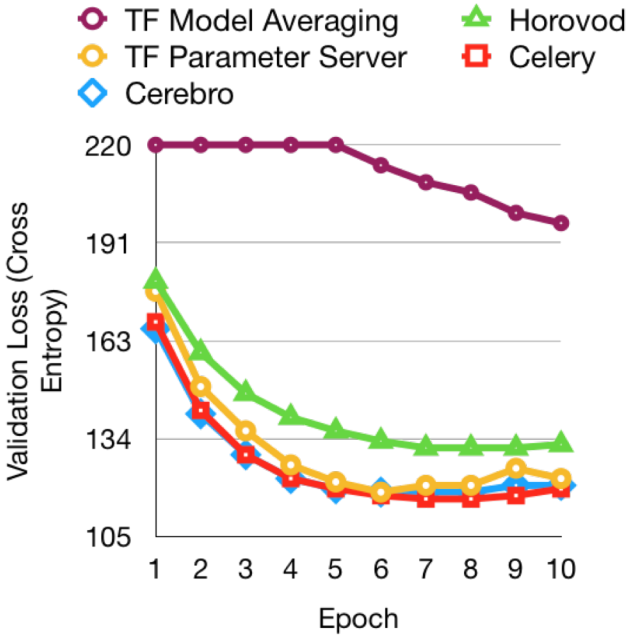
- 8-node GPU cluster
- Nvidia P100 GPUs
- Imagenet dataset
- 16 model configurations using VGG16 and ResNet50

Runtime Efficiency

System	Runtime (hrs)	GPU Utili.
TF Parameter Server	190.0	8.6%
Horovod (Dec. sync. parallel)	54.2	*92.1%
TF Model Averaging (BSP)	19.70	72.1%
Celery (Task parallel)	19.5	73.2%
Cerebro (MOP)	17.7	79.8%

Model hopper

Convergence Efficiency

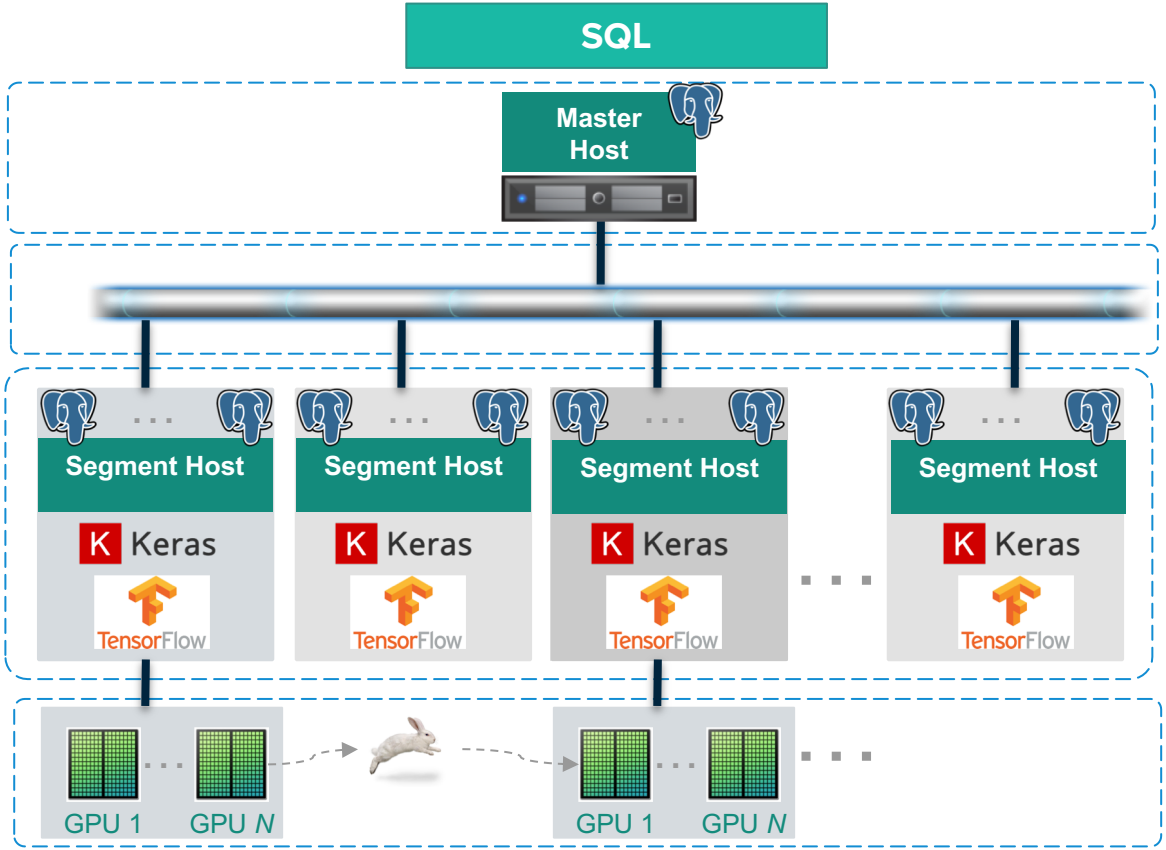


* Horovod uses GPU kernels for communication. Thus, it has high GPU utilization.

https://adalabucsd.github.io/papers/2019_Cerebro_DEEM.pdf

3. Model Hopper on Greenplum Database

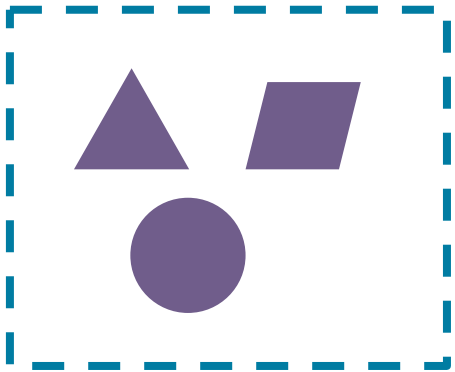
Greenplum Database with GPUs



GPUs only on certain hosts for cost reasons

API -- load_model_selection_table()

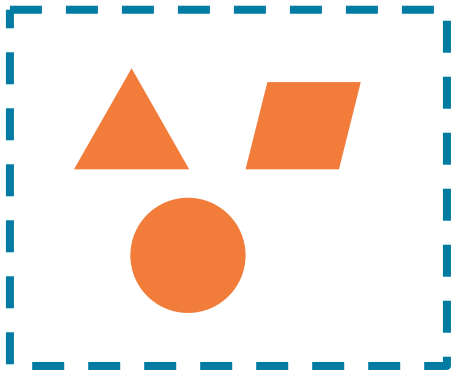
```
SELECT load_model_selection_table(
    'model_arch_table',             -- model architecture table
    'model_selection_table',        -- output table
    ARRAY[...],                     -- model architecture ids
    ARRAY[...],                     -- compile hyperparameters
    ARRAY[...]                      -- fit hyperparameters
);
```



model_selection_table

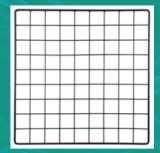
API -- madlib_keras_fit_multiple_model()

```
SELECT madlib_keras_fit_multiple_model(
    'data',                         -- data table
    'trained_models',               -- output table name
    'model_selection_table',        -- model selection table
    100,                            -- number of iterations
    True,                           -- use GPUs
    ...                             -- optional parameters
);
```



trained_models

4a. Grid Search Example Using Model Hopper

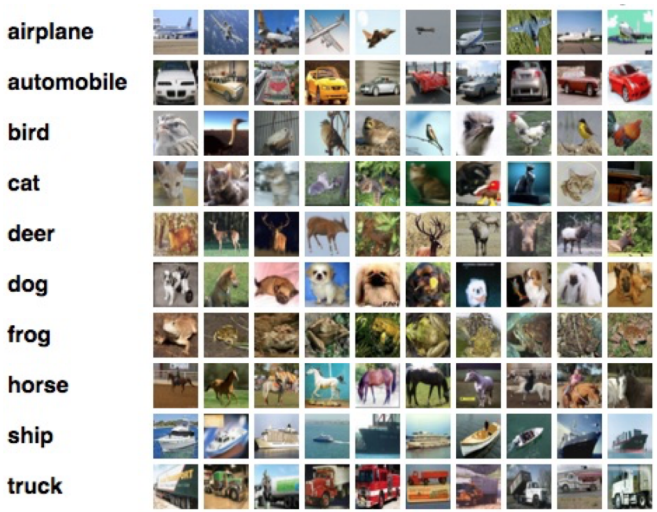


Test Setup

- 60k 32x32 color images in 10 classes, with 6k images per class
- 50k training images and 10k test images

https://www.cs.toronto.edu/~kriz/cifar.html

CIFAR-10

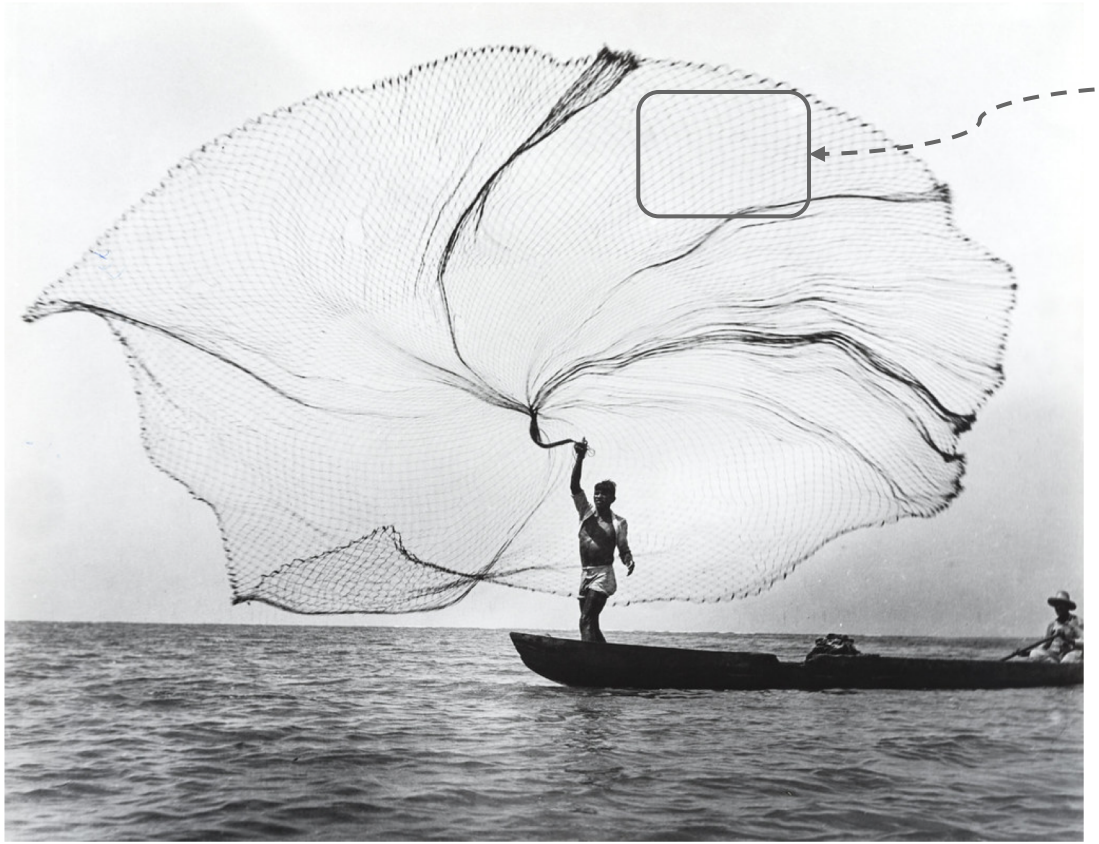


- 4 hosts each with 32 vCPUs & 150 GB memory & 4 NVIDIA Tesla P100 GPUs
- Greenplum 5 with 4 segments per host
- Apache MADlib 1.17
- Keras 2.2.4, TensorFlow 1.13.1

Google Cloud Platform

Pivotal

Approach to Training



Area of promise to fine tune

Henri Cartier-Bresson

Model Configurations

3
Model architecture

Model id	Type	Weights
1	CNN	1.2M
2	CNN	552K
3	CNN	500K

96 configs

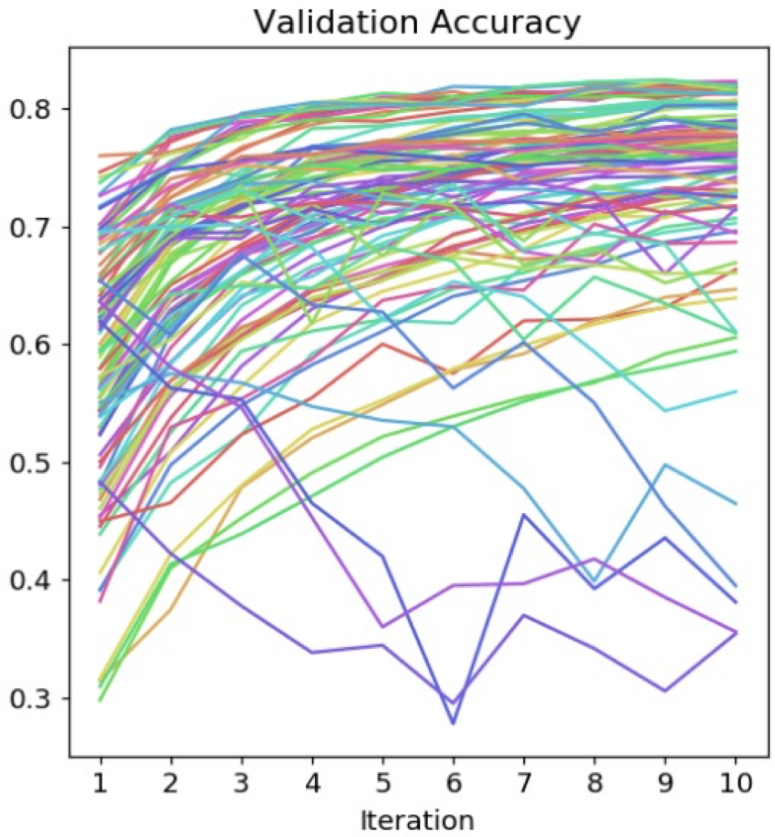
Optimizer 8

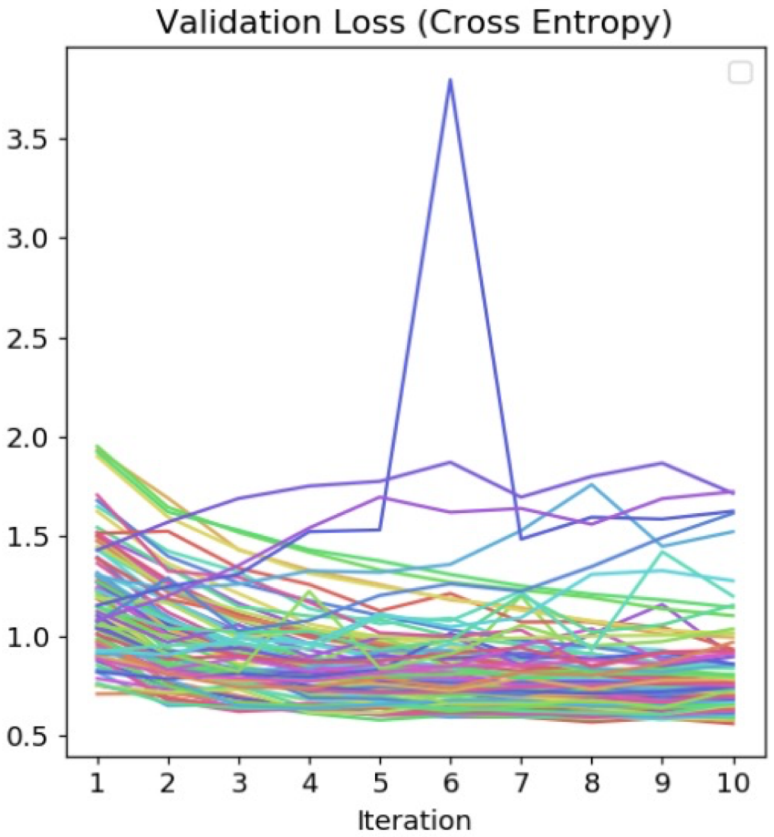
Type	Params
RMSprop	lr=.0001 decay=1e-6
RMSprop	lr=.001 decay=1e-6
Adam	lr=.0001
Adam	lr=.001
SGD	r=.001 momentum=0.9
SGD	r=.01 momentum=0.9
SGD	r=.001 momentum=0.95
SGD	r=.01 momentum=0.95

4
Batch size

Model id
32
64
128
256

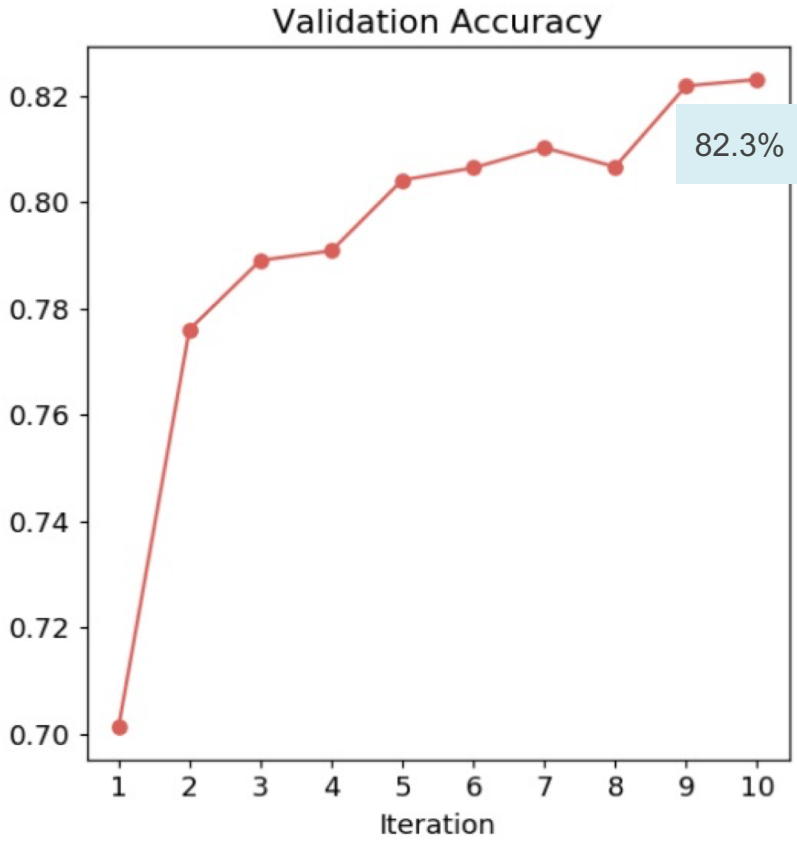
Too much information!

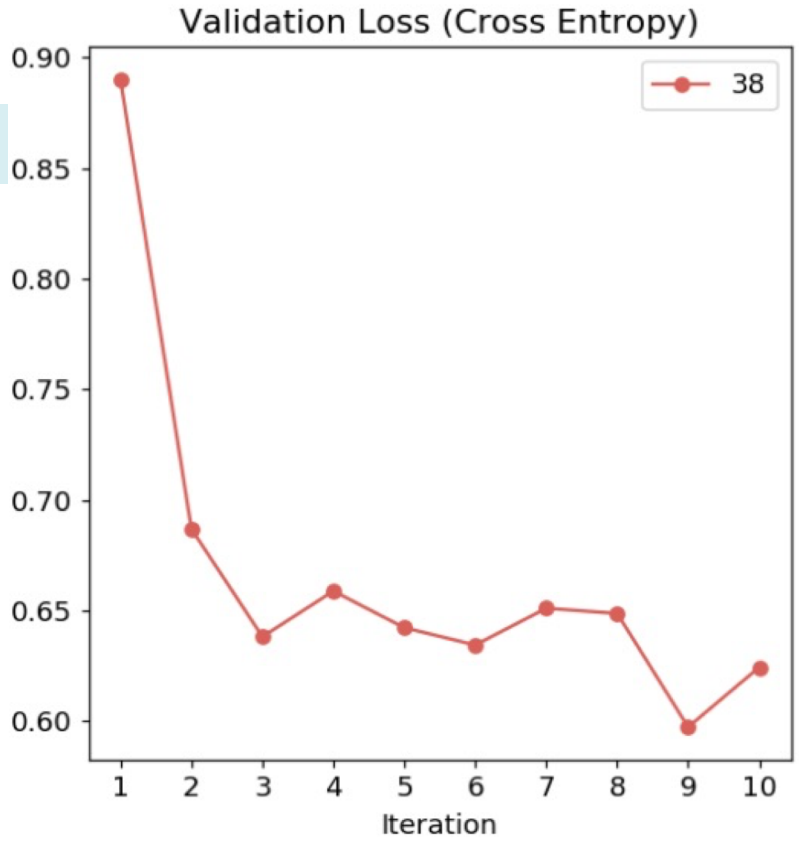




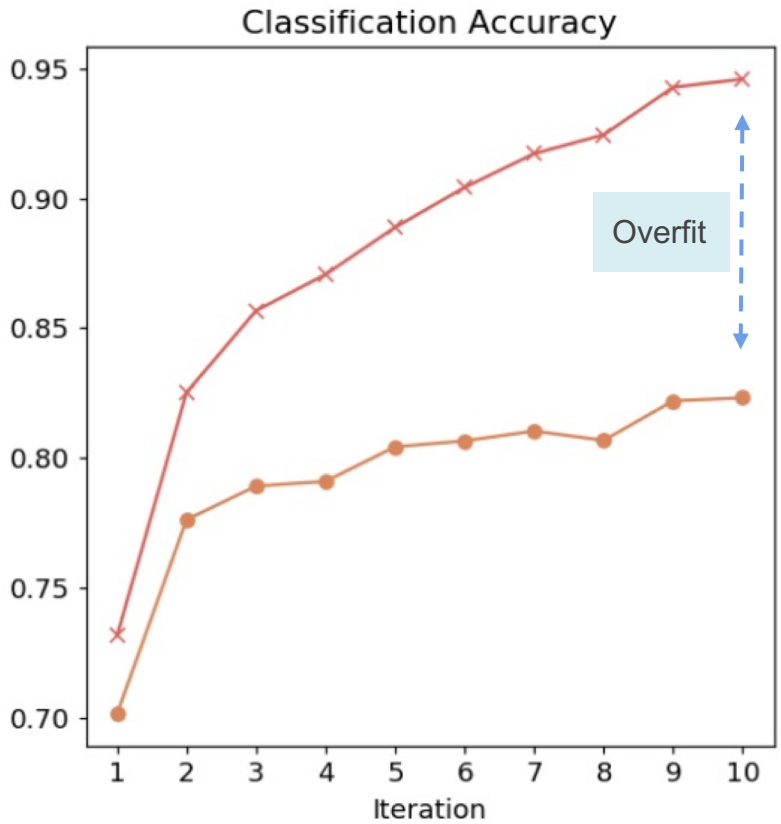
Run time: 2:47:57 on 16 workers/16 GPUS

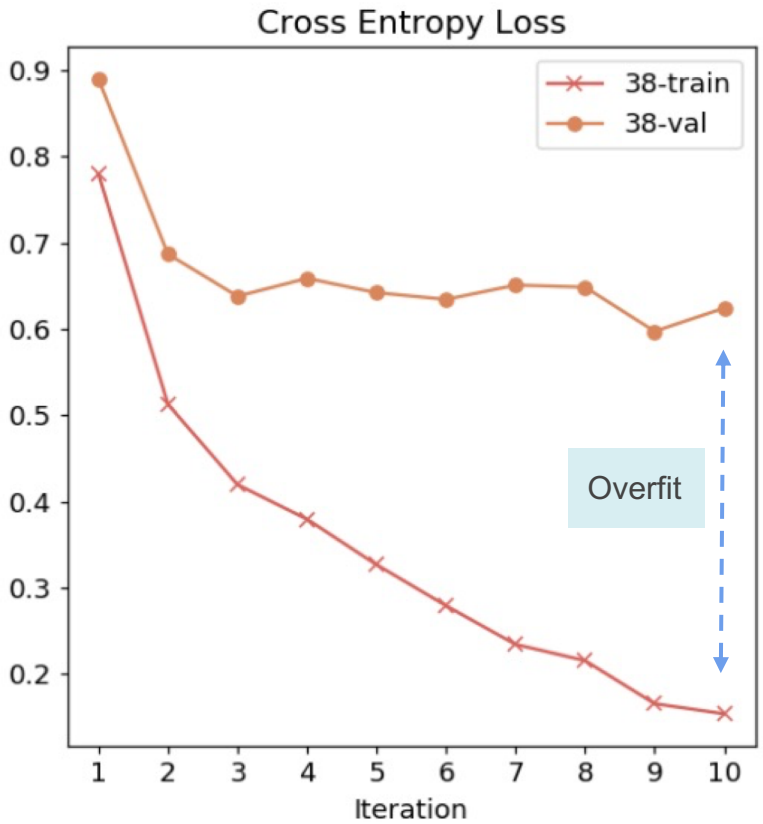
Why not just use the most accurate?



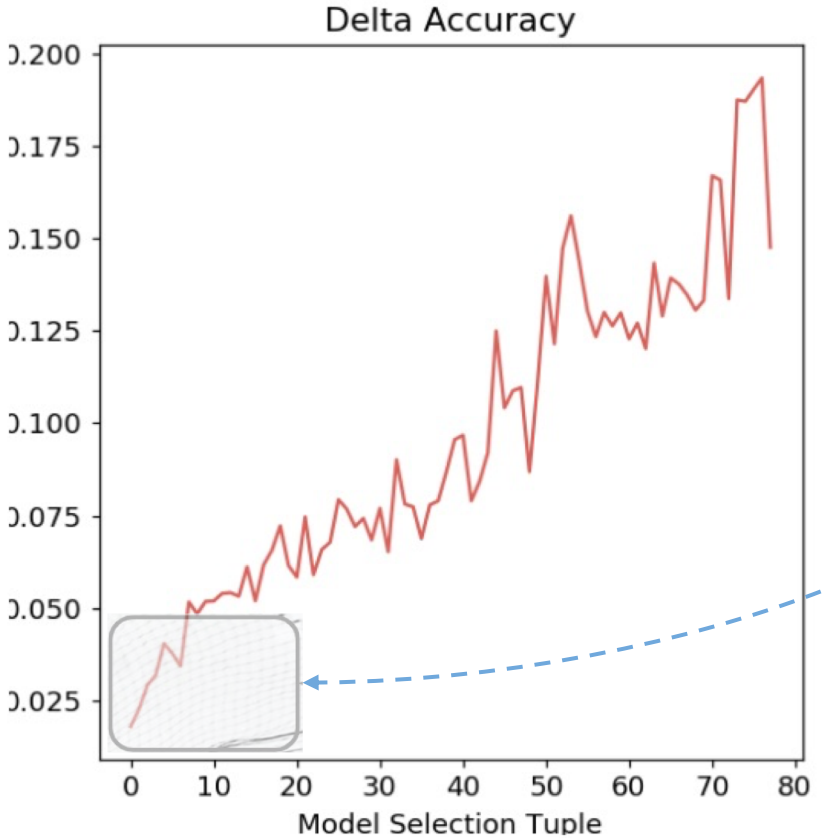


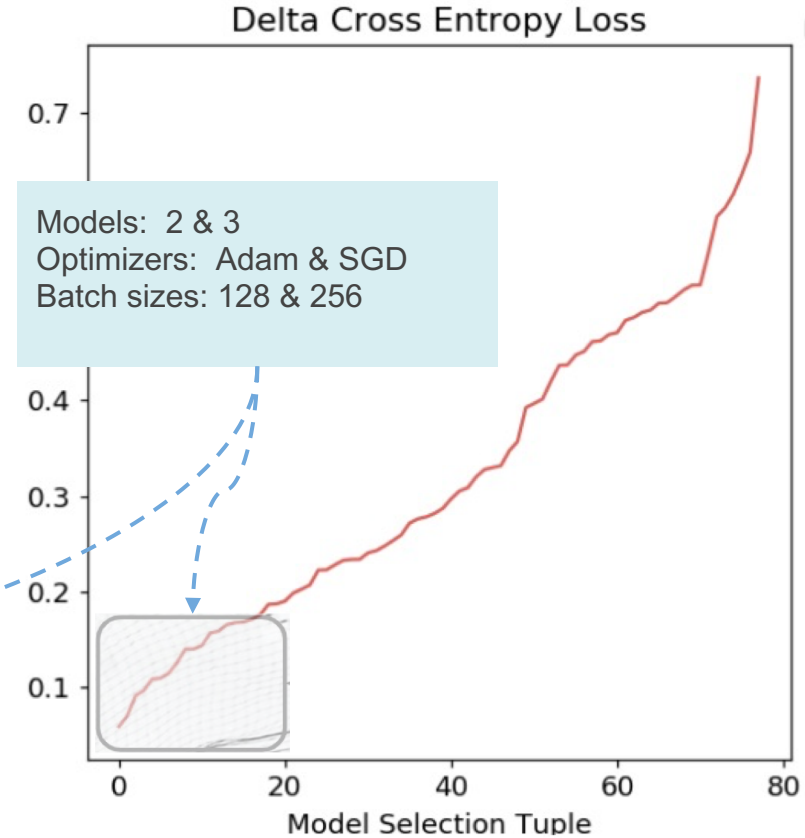
Because it is overfitting the data





Focus on models with low overfit

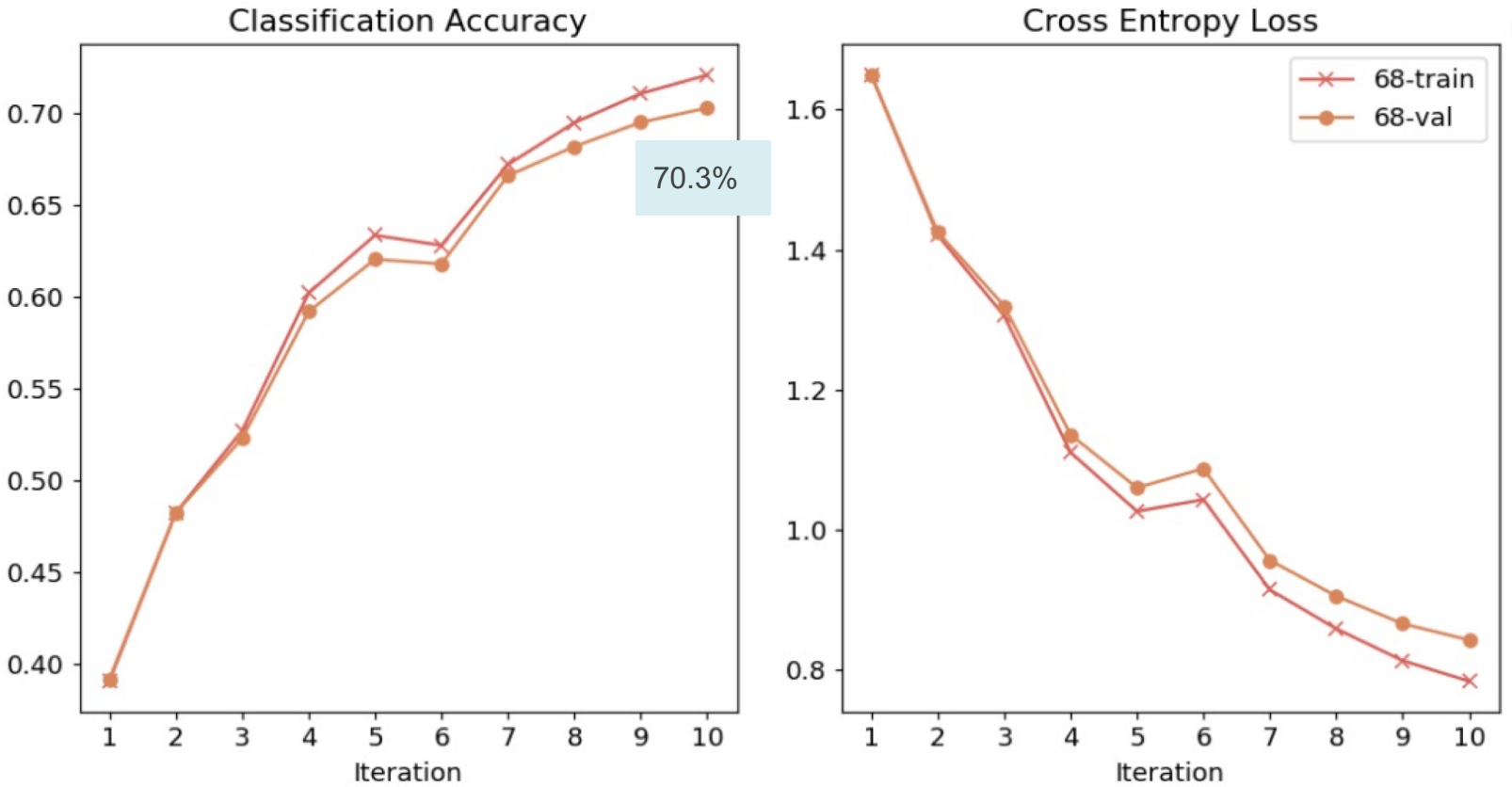




Models: 2 & 3
Optimizers: Adam & SGD
Batch sizes: 128 & 256

These charts only consider models with accuracy > 70%

Example with less overfitting



Refine Model Configs

2
Model architecture

Model id	Type	Weights
1	CNN	1.2M ✖
2	CNN	552K
3	CNN	500K

12 configs
(was 96)

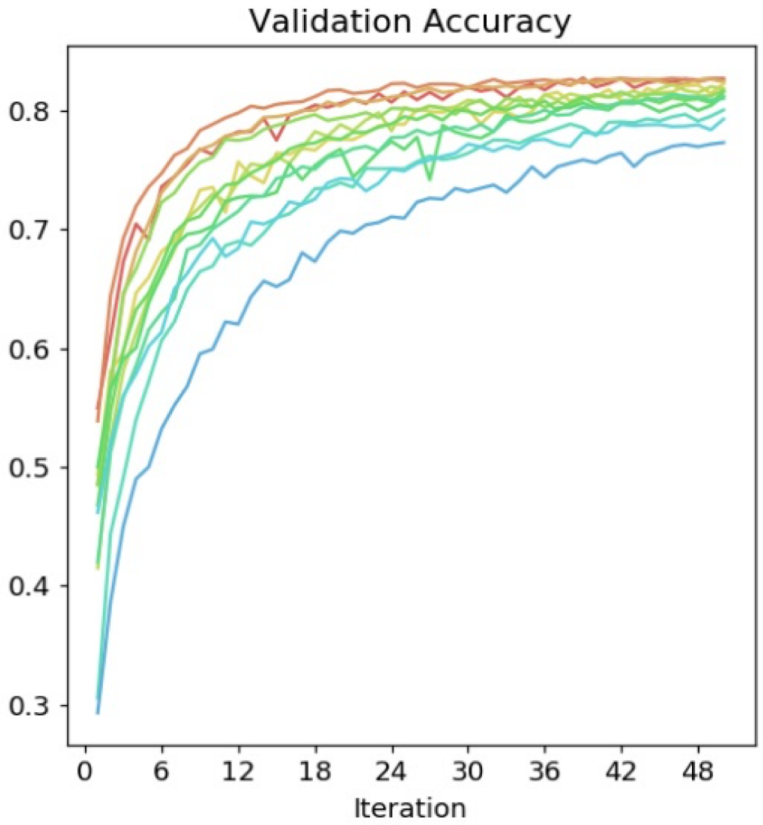
Optimizer 3

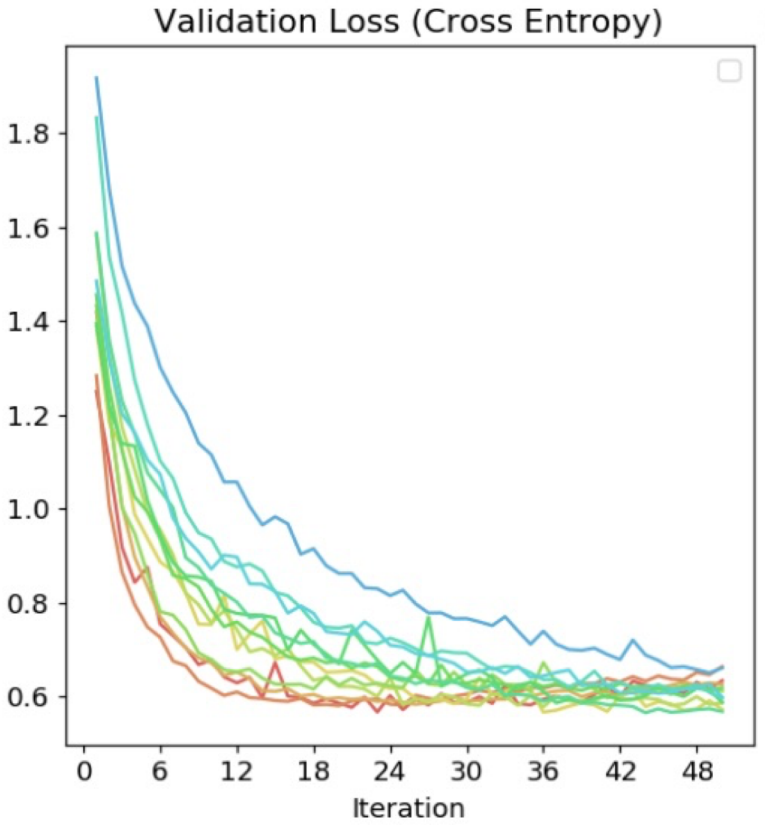
Type	Params
RMSprop	lr=.0001 decay=1e-6 ✖
RMSprop	lr=.001 decay=1e-6 ✖
Adam	lr=.0001
Adam	lr=.001 ✖
SGD	r=.001 momentum=0.9
SGD	r=.01 momentum=0.9 ✖
SGD	r=.001 momentum=0.95
SGD	r=.01 momentum=0.95 ✖

2
Batch size

Model id
32 ✖
64 ✖
128
256

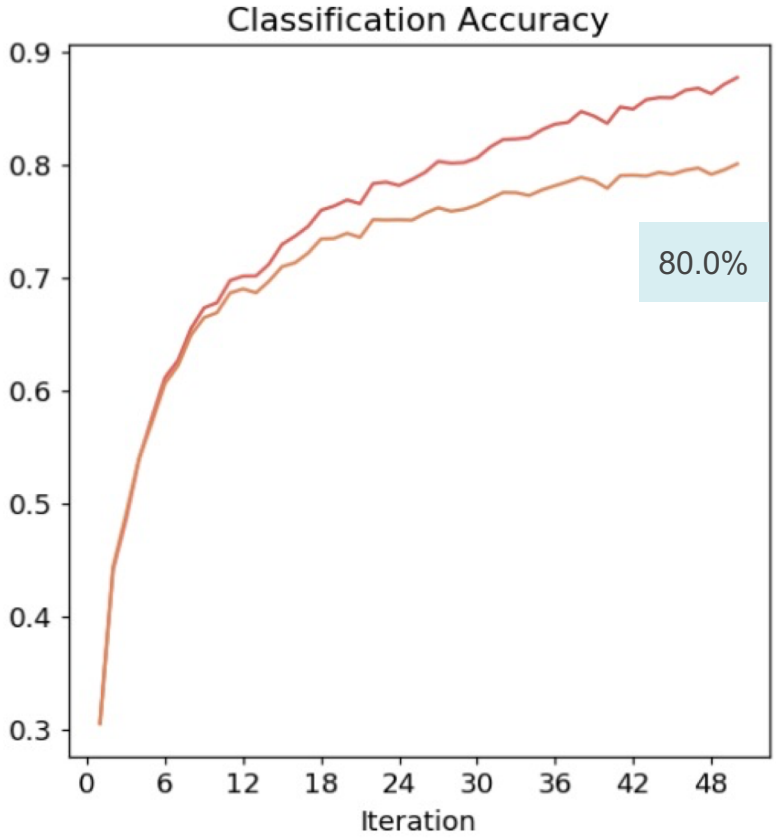
Re-run the best 12 with more iterations

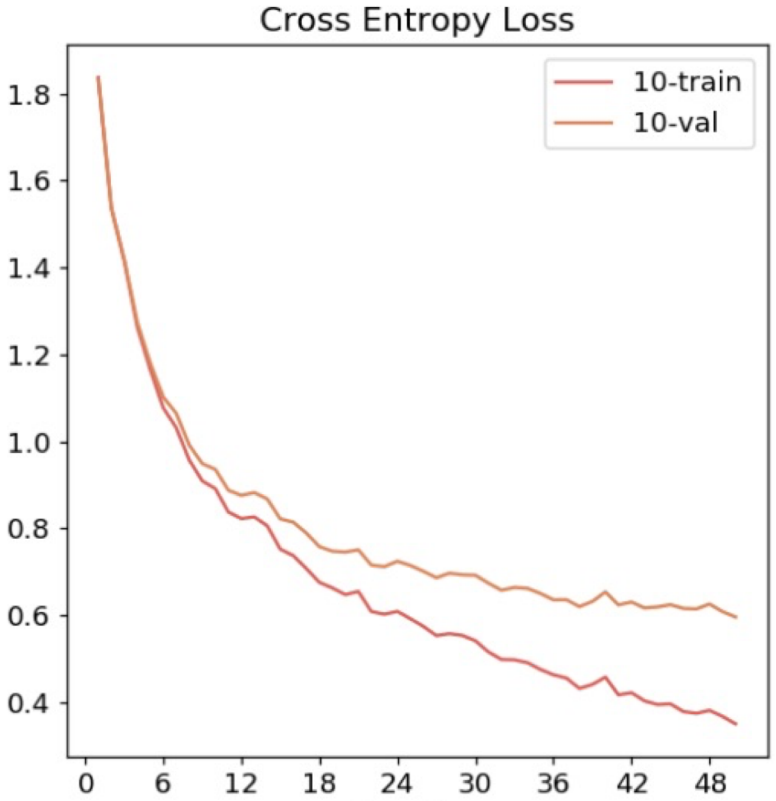




Run time: 1:53:06 on 16 workers/16 GPUS

Example good result

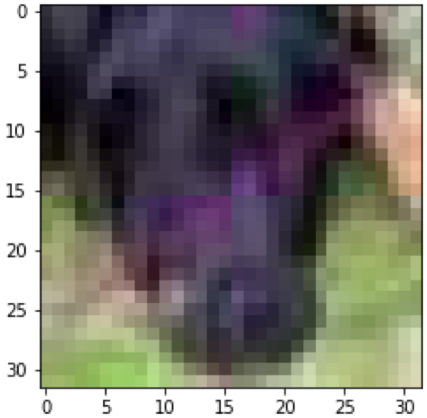




sgd (lr=0.001,momentum=0.95)
batch size=256

Inference

Who am I ???



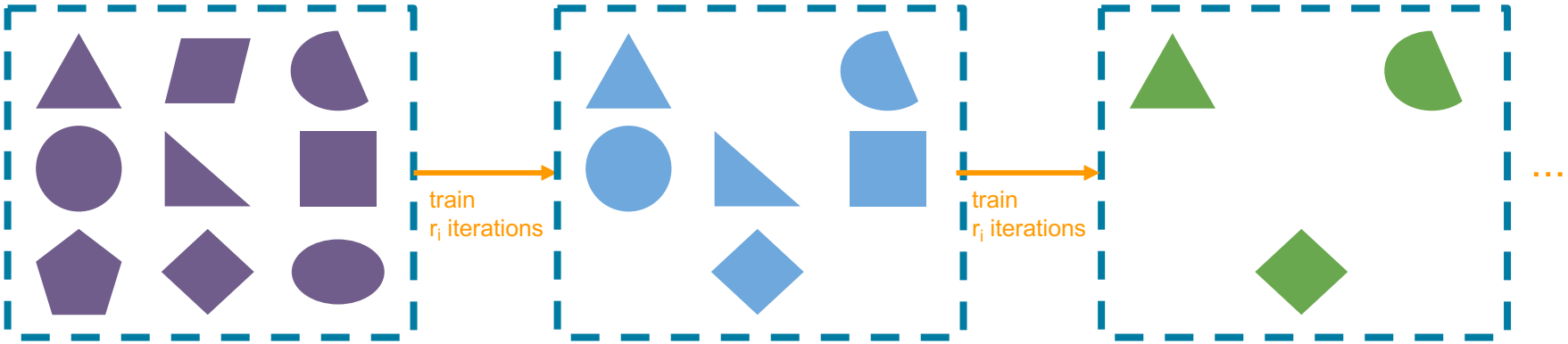
Model says:

dog	0.9532
deer	0.0258
bird	0.0093

4b. AutoML Example Using Model Hopper

Hyperband

Successive halving



Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization
https://arxiv.org/pdf/1603.06560.pdf

Hyperband

Inputs:

- R: maximum amount of resource that can be allocated to a single configuration
- η: controls the proportion of configurations discarded in each round of successive halving

bracket | num model configs | num iterations

R= 81
η = 3

	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

Model Configurations

Note ranges specified, not exact values

Model architecture

Model id	Type	Weights
1	CNN	1.2M
2	CNN	552K
3	CNN	500K

Optimizer

Type	Params
RMSprop	lr=[.0001, 0.001] decay=1e-6
Adam	lr=[.0001, 0.001]
SGD	lr=[.001, 0.01] momentum=[0.9, 0.95]

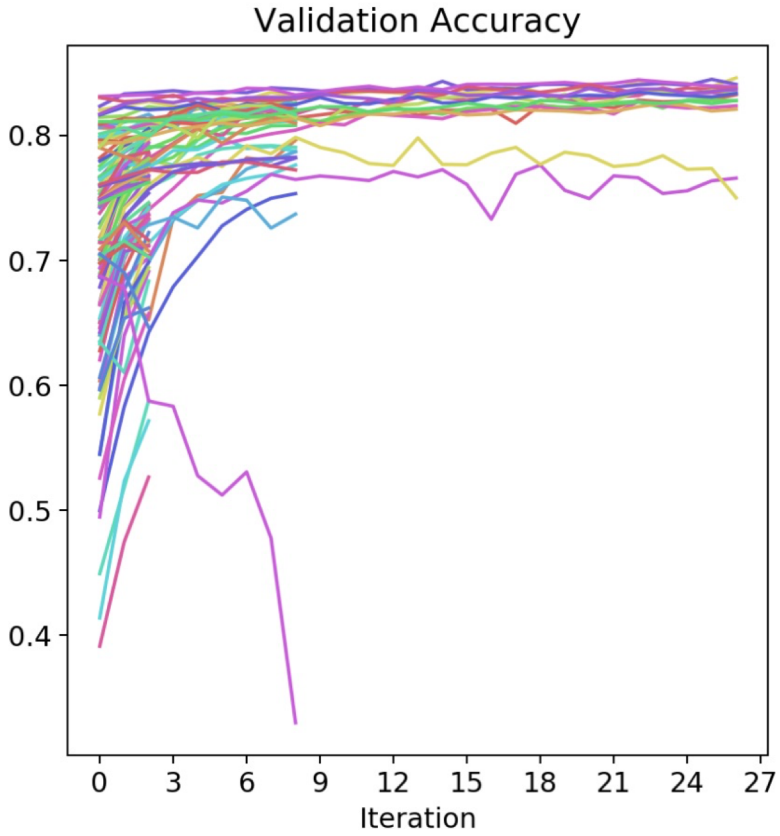
Batch size

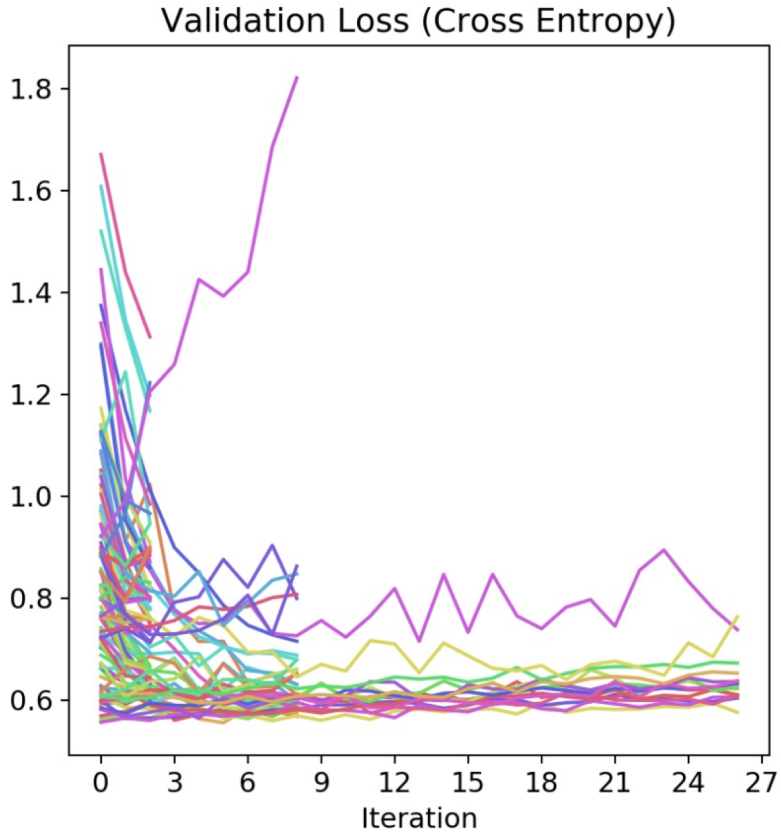
Model id
32
64
128
256

Hyperband schedule

	s=4		s=3		s=2		s=1		s=0	
params	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
	---		---		---		---		---	
max_iter = 81	81	1.0	27	3.0	9	9.0	6	27.0		
eta = 3	27.0	3.0	9.0	9.0	3.0	27.0				
skip_last = 1	9.0	9.0	3.0	27.0						
	3.0	27.0								

Again, a lot of information!





Run time: 3:21:04 on 16 workers/16 GPUS

Refine Model Configurations

Adjust ranges

Model architecture

Model id	Type	Weights
1	CNN	1.2M ✖
2	CNN	552K
3	CNN	500K

Optimizer

Type	Params
RMSprop	lr=[.0001, 0.001] decay=1e-6 ✖
Adam	lr=[.0001, 0.0005]
SGD	~~lr=[.001, 0.01]~~ lr=[.001, 0.005] momentum=[0.9, 0.95]

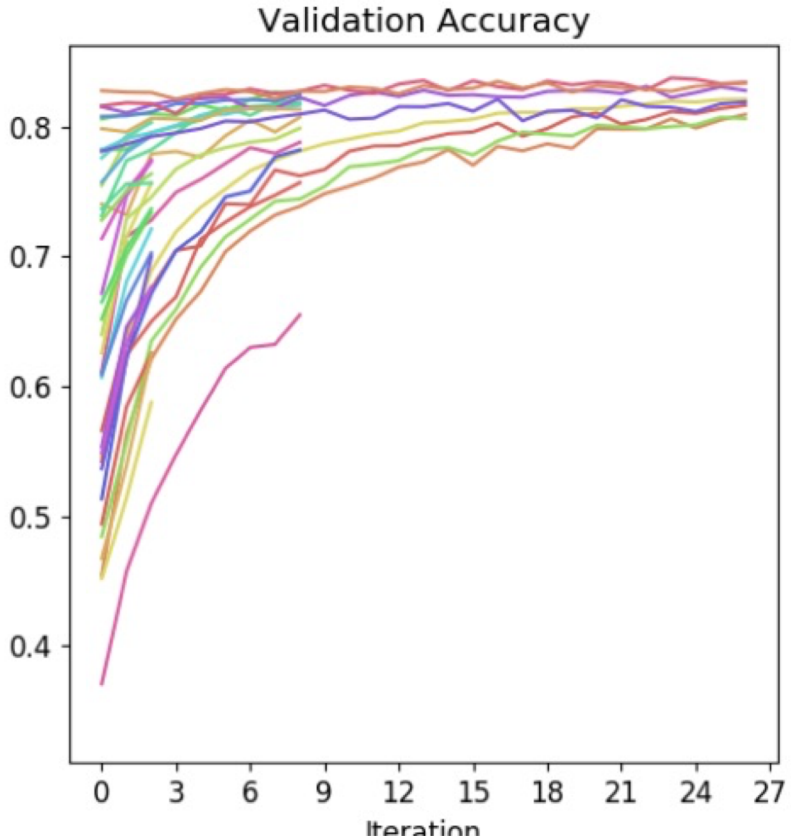
Batch size

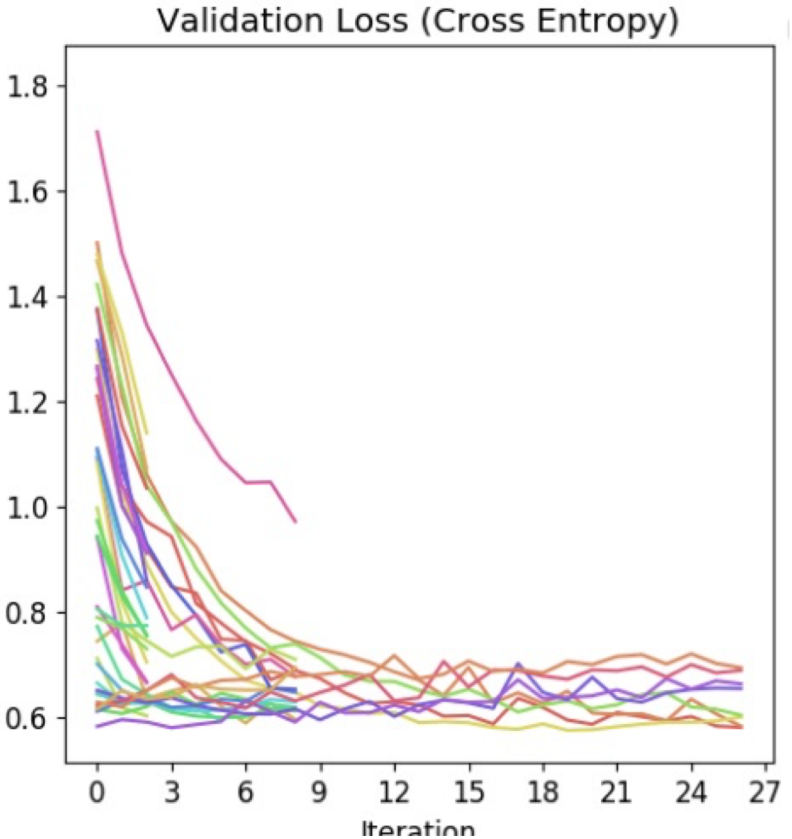
Model id
32 ✖
64 ✖
128
256

Adjust Hyperband schedule

	s=3		s=2		s=1		s=0	
params	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
	---	---	---	---	---	---	---	---
max_iter = 27	27	1.0	9	3.0	6	9.0	4	27
eta = 3	9.0	3.0	3.0	9.0	2.0	27.0		
skip_last = 0	3.0	9.0	1.0	27.0				
	1.0	27.0						

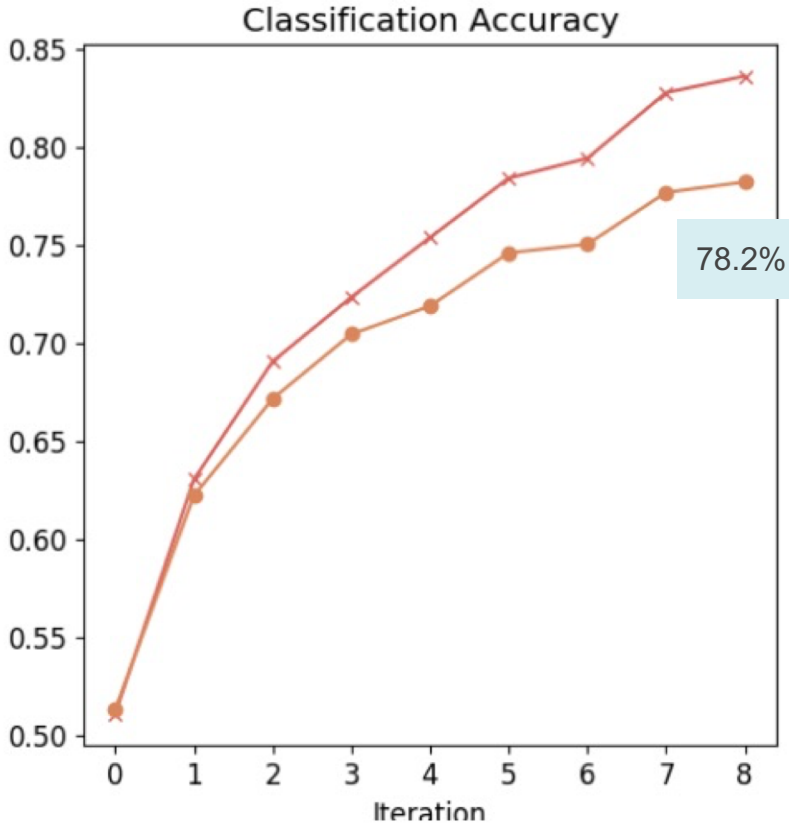
Re-run the best with Hyperband again



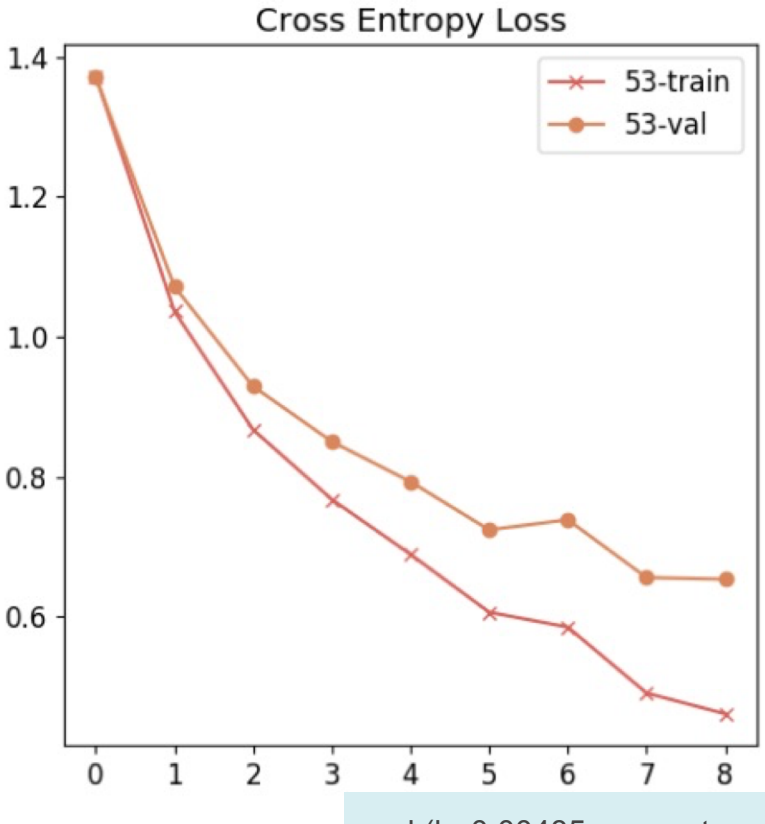


Run time: 55:49 on 16 workers/16 GPUS

Example good result



78.2%



sgd (lr=0.00485,momentum=0.90919)
batch size=128

Summary

- Model hopper can efficiently train many deep nets at a time on an MPP database
- Can add autoML methods on top of model hopper
- Areas of future work:
  - Improve GPU efficiency
  - Add more autoML methods

References

- Model hopper

  - Supun Nakandala, Yuhao Zhang, and Arun Kumar, "Cerebro: Efficient and Reproducible Model Selection on Deep Learning Systems," DEEM'30, June 30, 2019, Amsterdam, Netherlands
    https://adalabucsd.github.io/papers/2019_Cerebro_DEEM.pdf

- Greenplum
  - https://github.com/greenplum-db/gpdb
  - https://greenplum.org/

- Apache MADlib
  - https://github.com/apache/madlib
  - http://madlib.apache.org/

- Jupyter notebooks
  - https://github.com/apache/madlib-site/tree/asf-site/community-artifacts

Thank you!

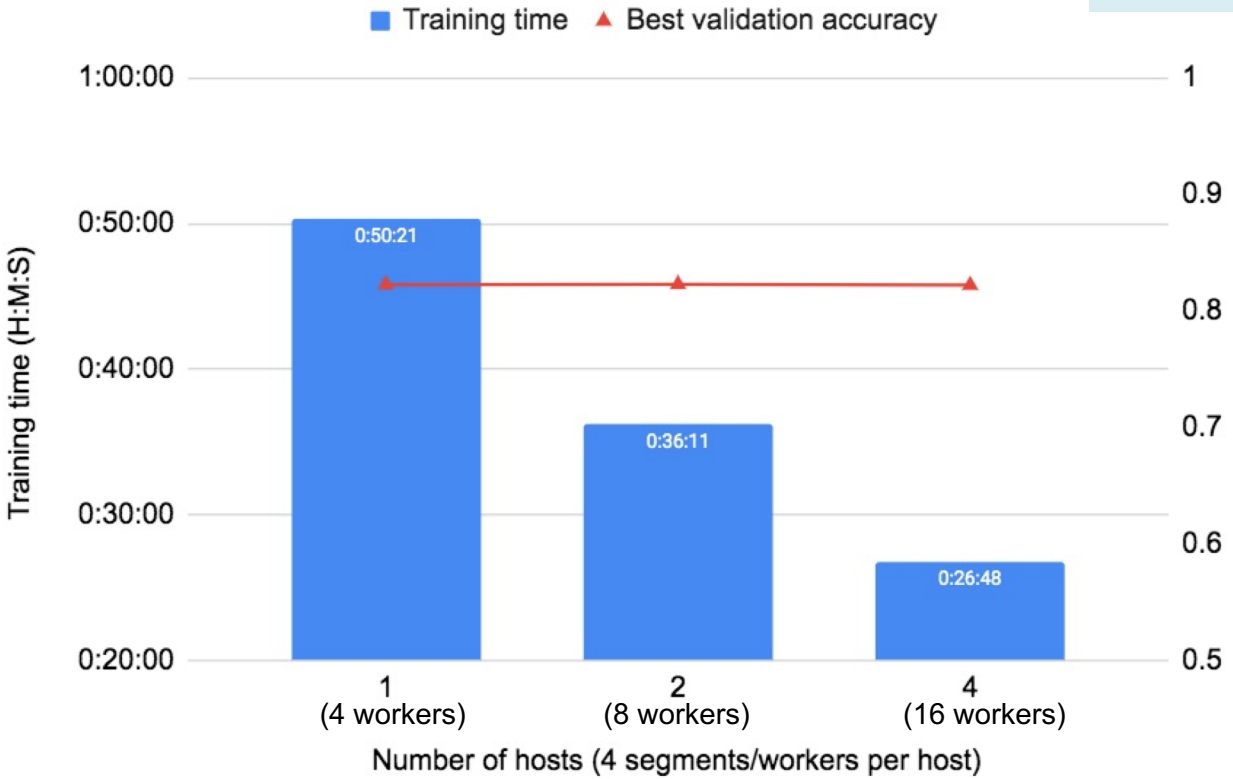
Backup Slides

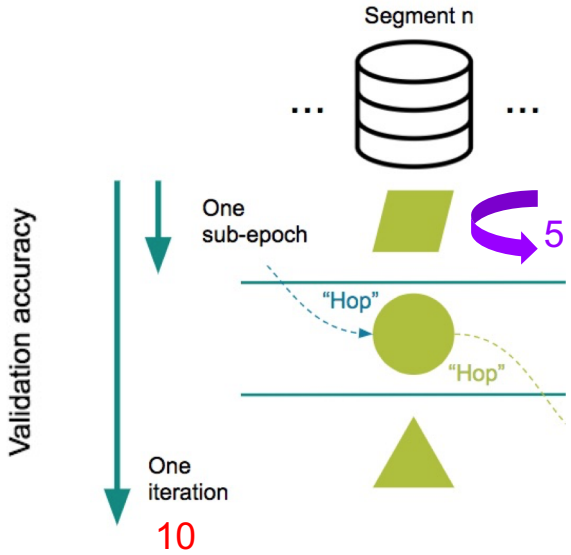
Tuning Epochs on a Distributed System

Effect of cluster size

CIFAR-10, 2 CNNs with ~500k and ~1M weights
16 model configurations

5 passes per sub-epoch X 10 iterations =
50 epochs

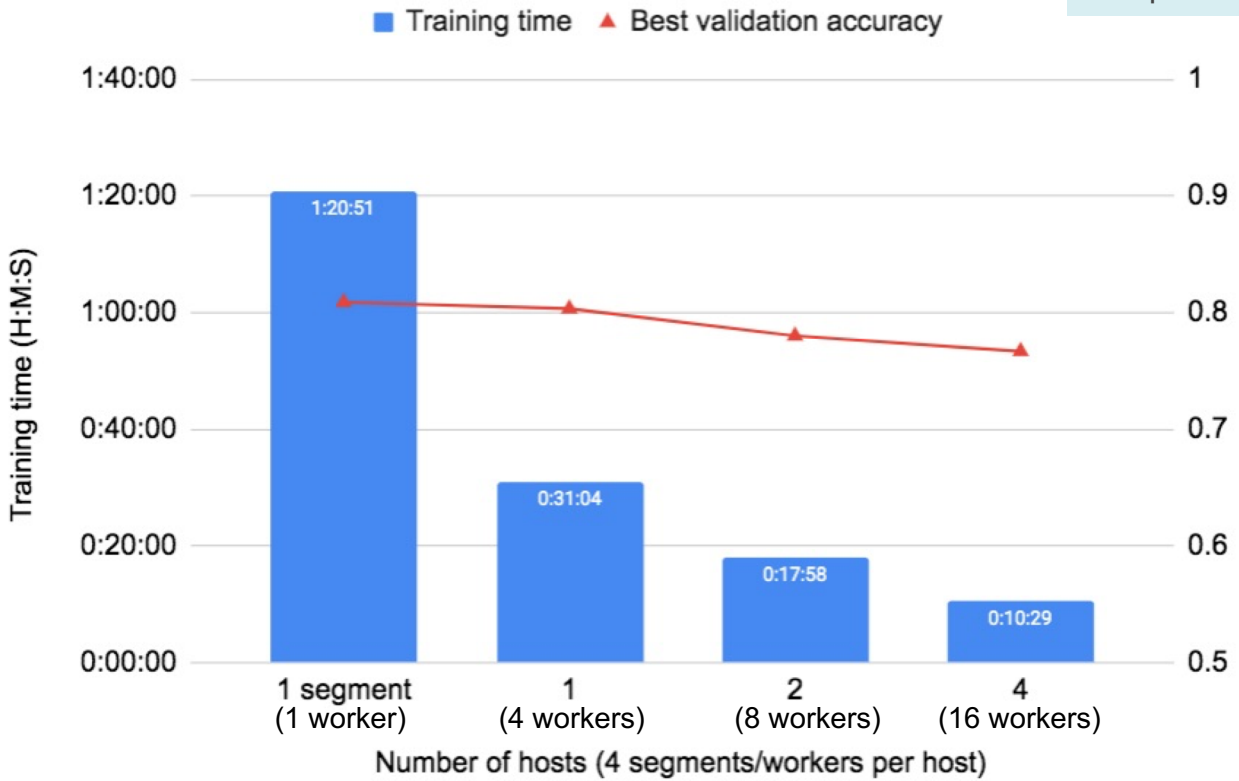


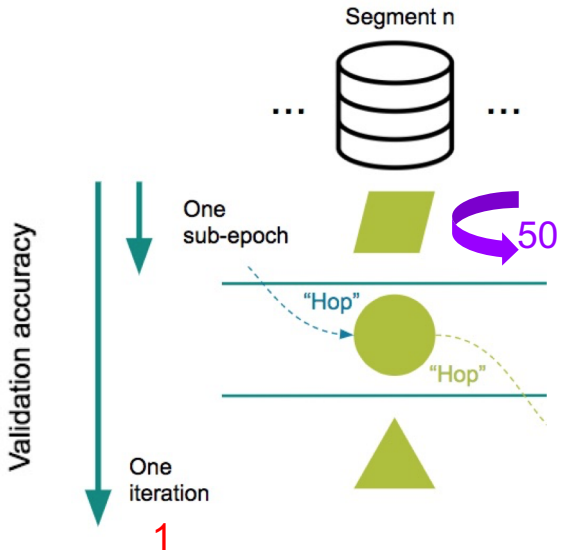


Effect of cluster size

CIFAR-10, 2 CNNs with ~500k and ~1M weights
16 model configurations

50 passes per sub-epoch X 1 iteration =
50 epochs





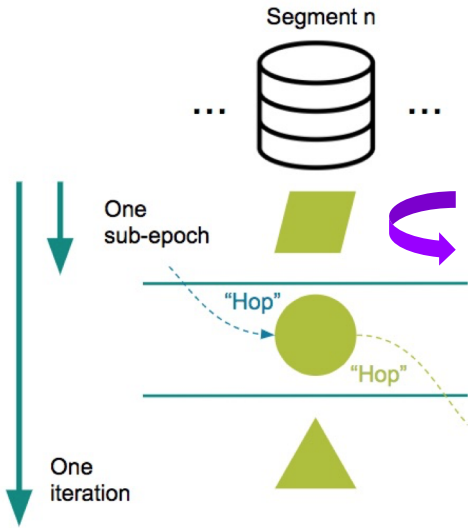
Effect of passes per sub-epoch

CIFAR-10, 2 CNNs with ~500k and ~1M weights

16 model configurations

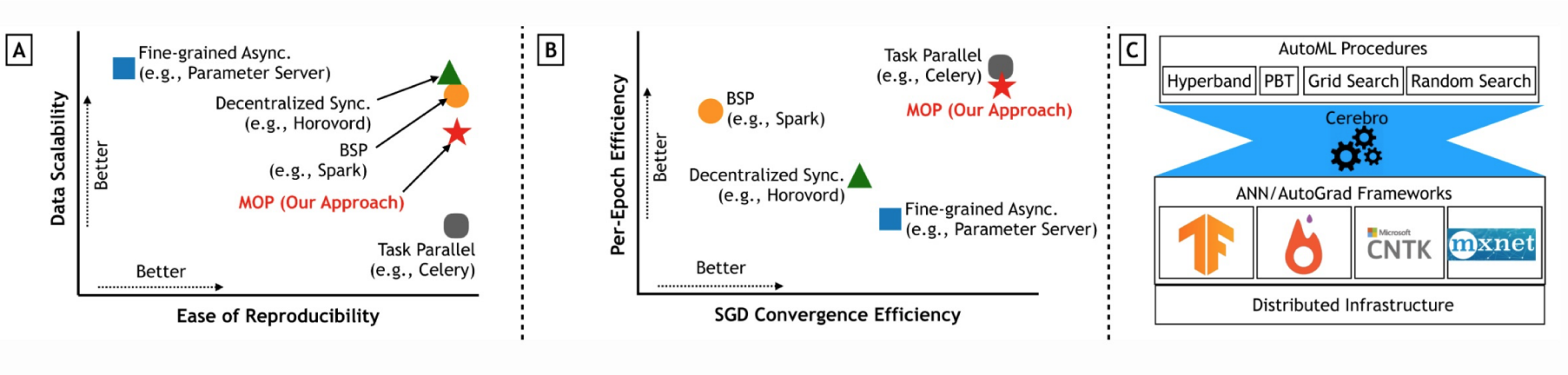
4 hosts X 4 workers per host = 16 workers

4 hosts X 4 GPUs per host = 16 GPUs



Other MOP Slides

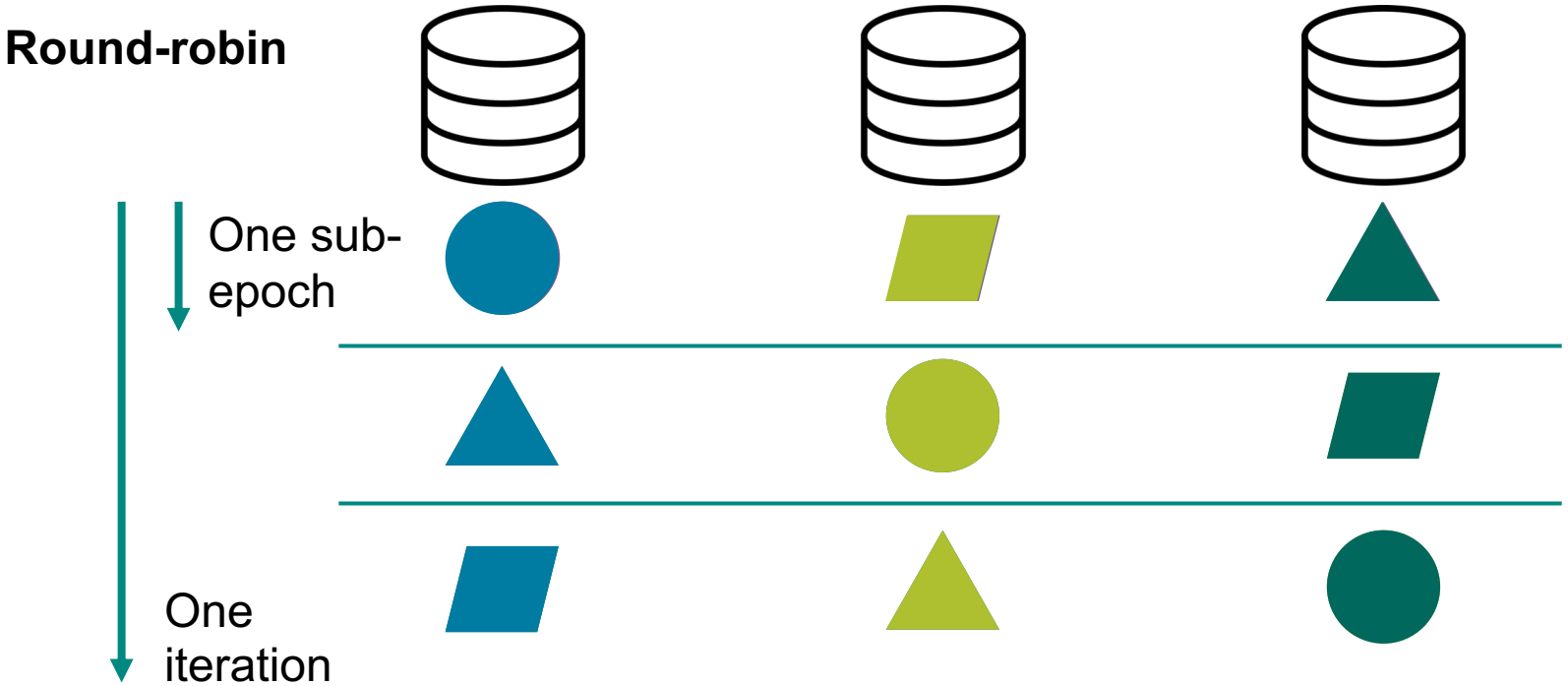
Comparison of Parallel Training Approaches



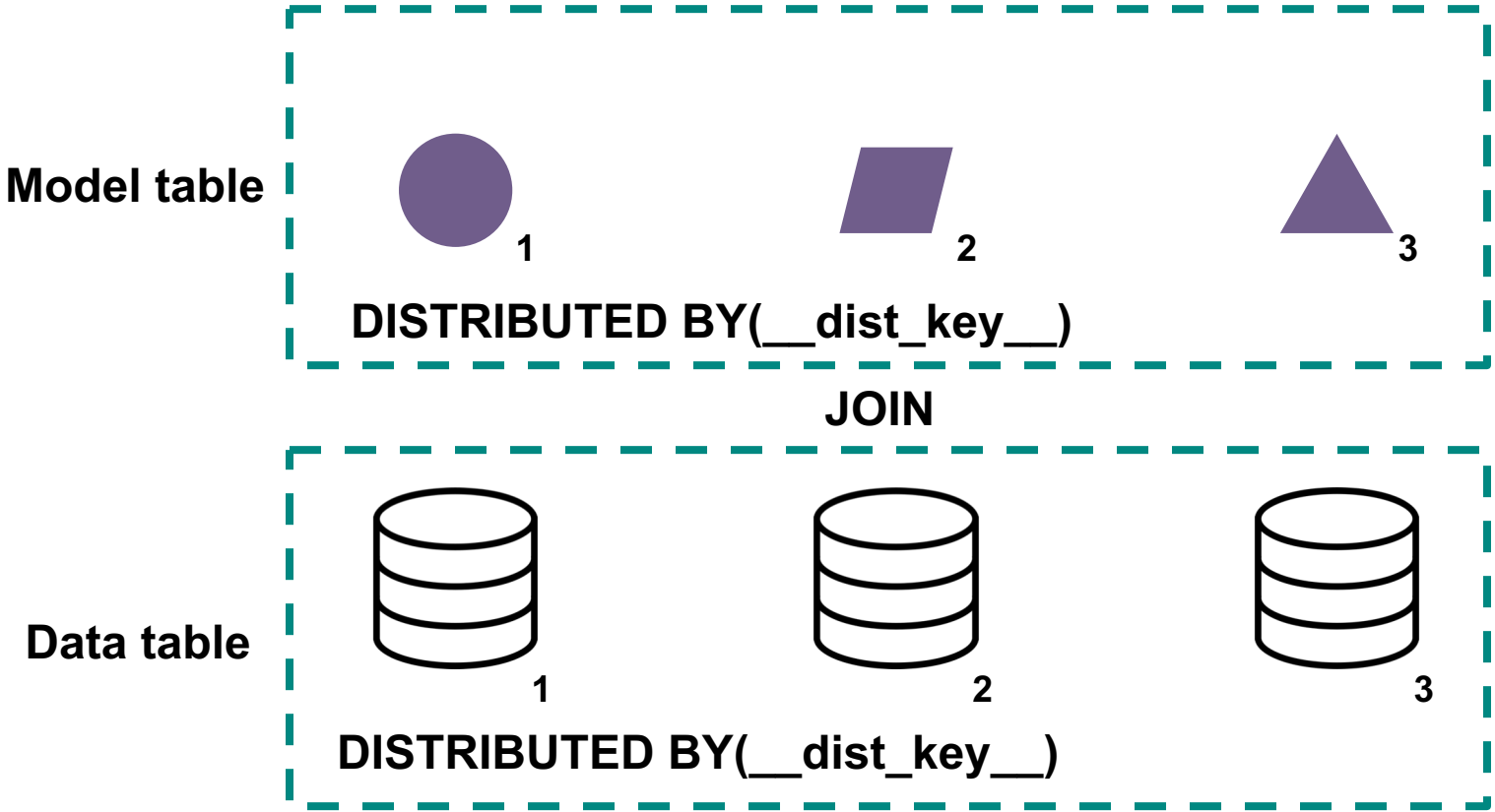
Implementation on GPDB

1. Scheduling
2. Dispatch model to specific segment (worker)
3. Run training
4. No data motion

Implementation on GPDB -- Scheduling



Implementation on GPDB -- Dispatching Models



Implementation on GPDB -- Training

SELECT  **UDA**

FROM

GROUP BY **__dist_key__**

Other Hyperband Slides

Hyperband Implementation on Greenplum

Running 1 bracket at a time will result in idle machines

81 model configurations
1 iteration each

1 model configuration
81 iterations

R= 81
η = 3

	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

Hyperband Implementation on Greenplum

Run across brackets “on the diagonal” to keep machines busy

1+1+1+2+5 = 10 model configurations
81 iteration eachs

R= 81
η = 3

	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								