Distributed Deep Learning Training

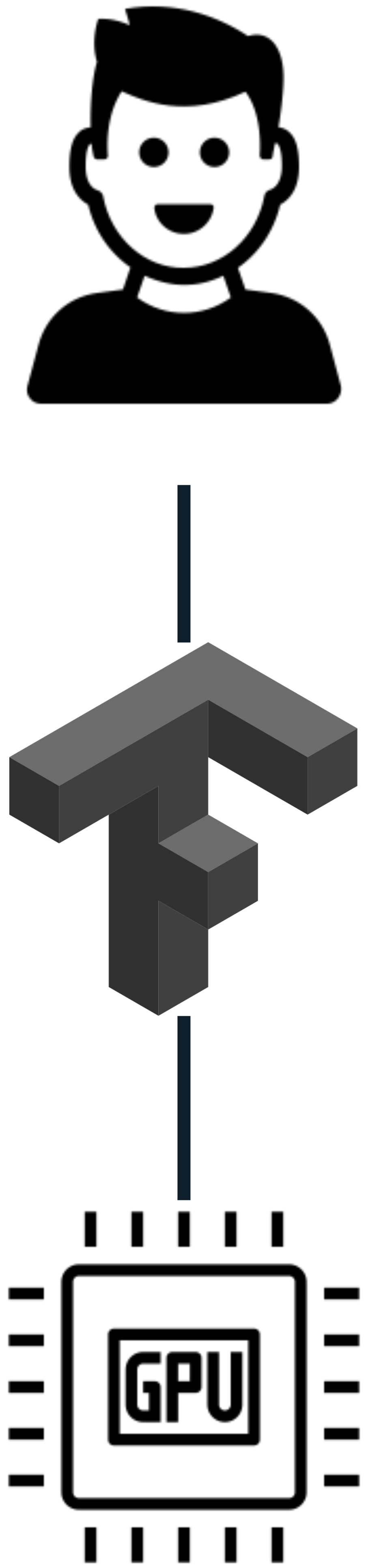
Aaron Harlap
aaron@determined.ai

Determined AI

Talk Outline

- Current Deep Learning tool chain.
- Why do we need distributed multi-machine training?
- How does distributed training work?
- Challenges and effective solutions.

Team of one



Small and simple

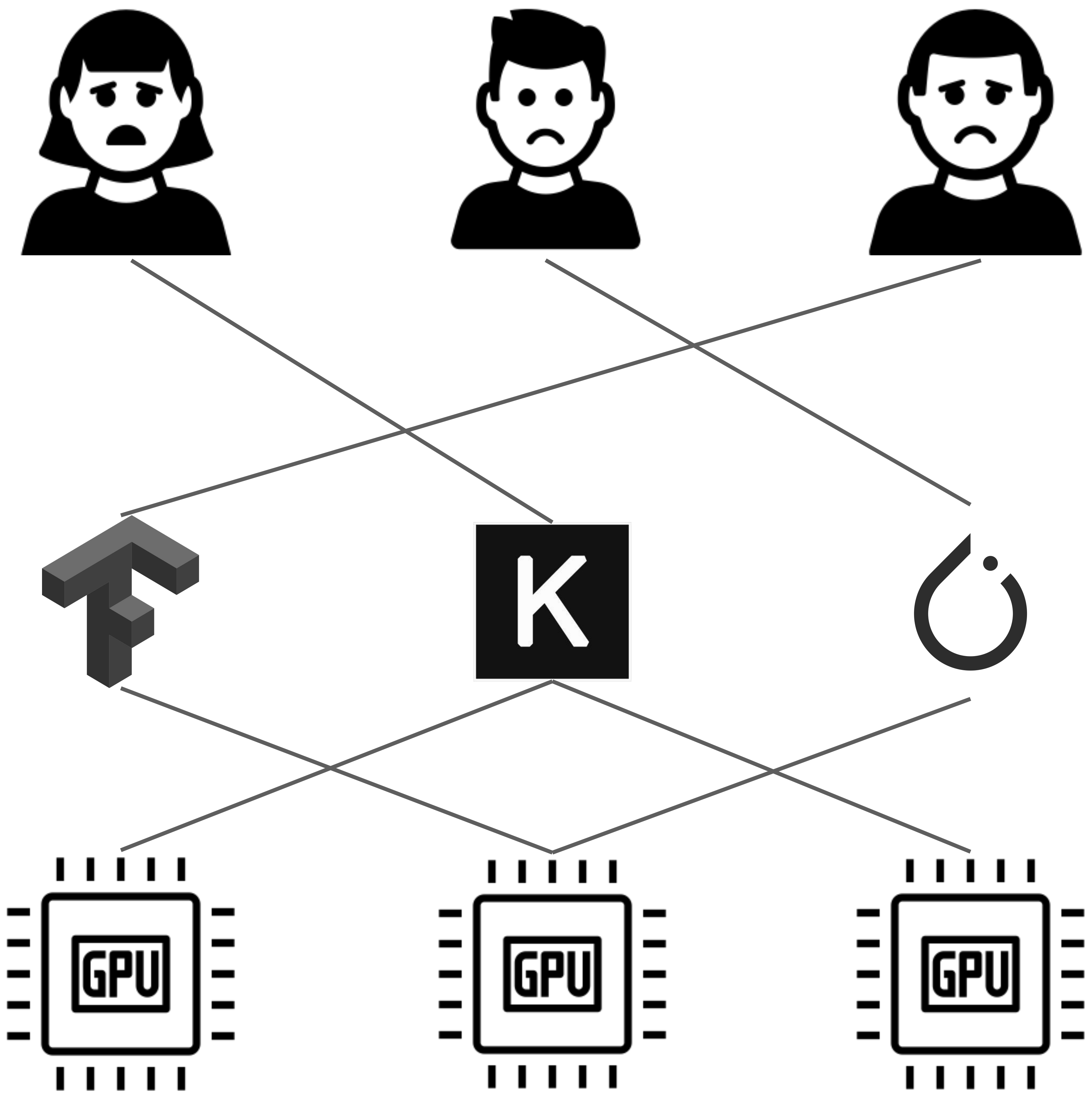
Relatively small experiments on single machine

No need to share resources or productionize results

Relatively simple infrastructure to manage

Simple experiment tracking is sufficient, e.g., `results/lstm.dataset.batchsize-16.epochs-500.opt-adam.log`

As scalability kicks in



Challenges surface

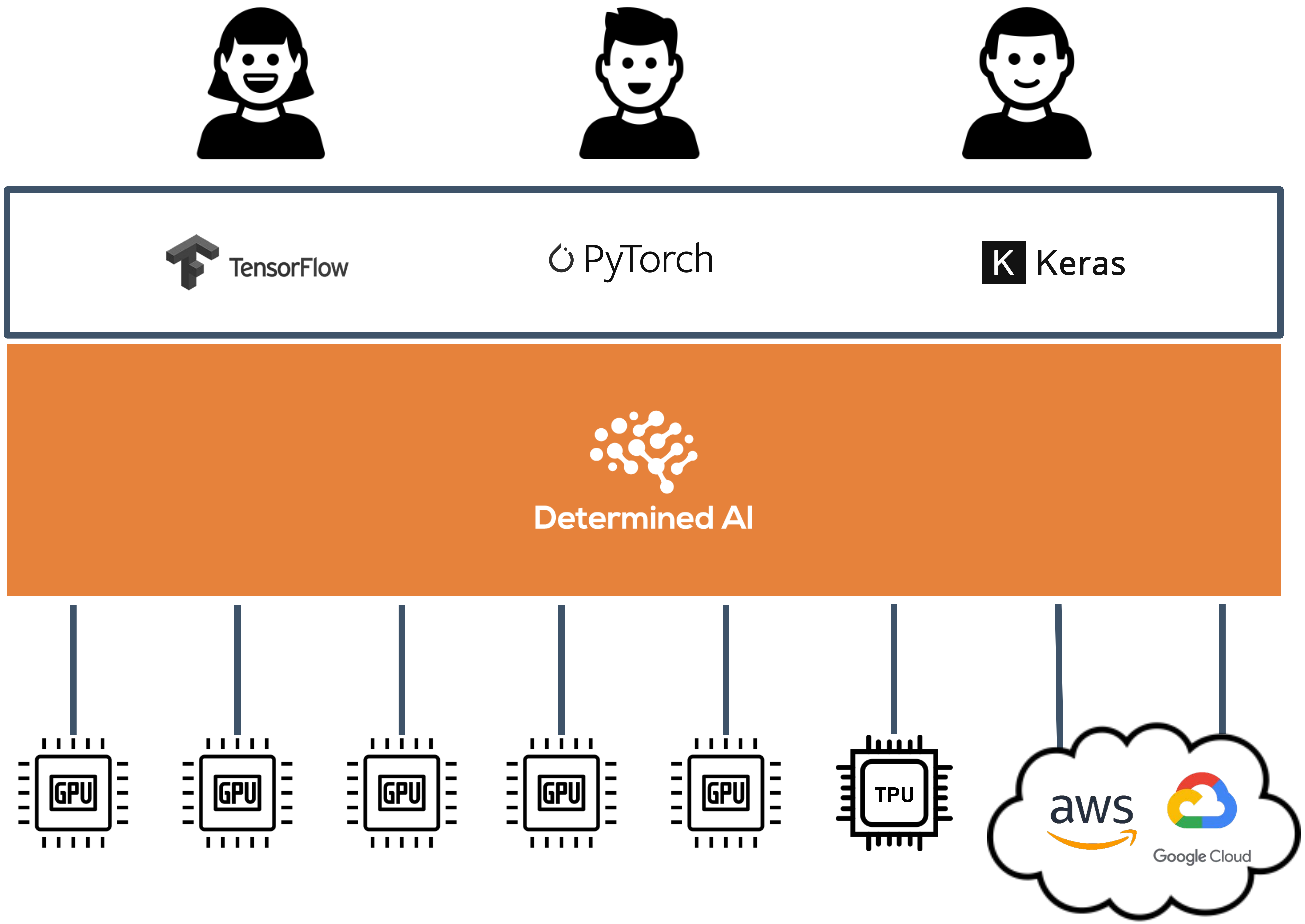
Resource/GPU sharing and infrastructure management

Experiment tracking for reproducibility and collaboration

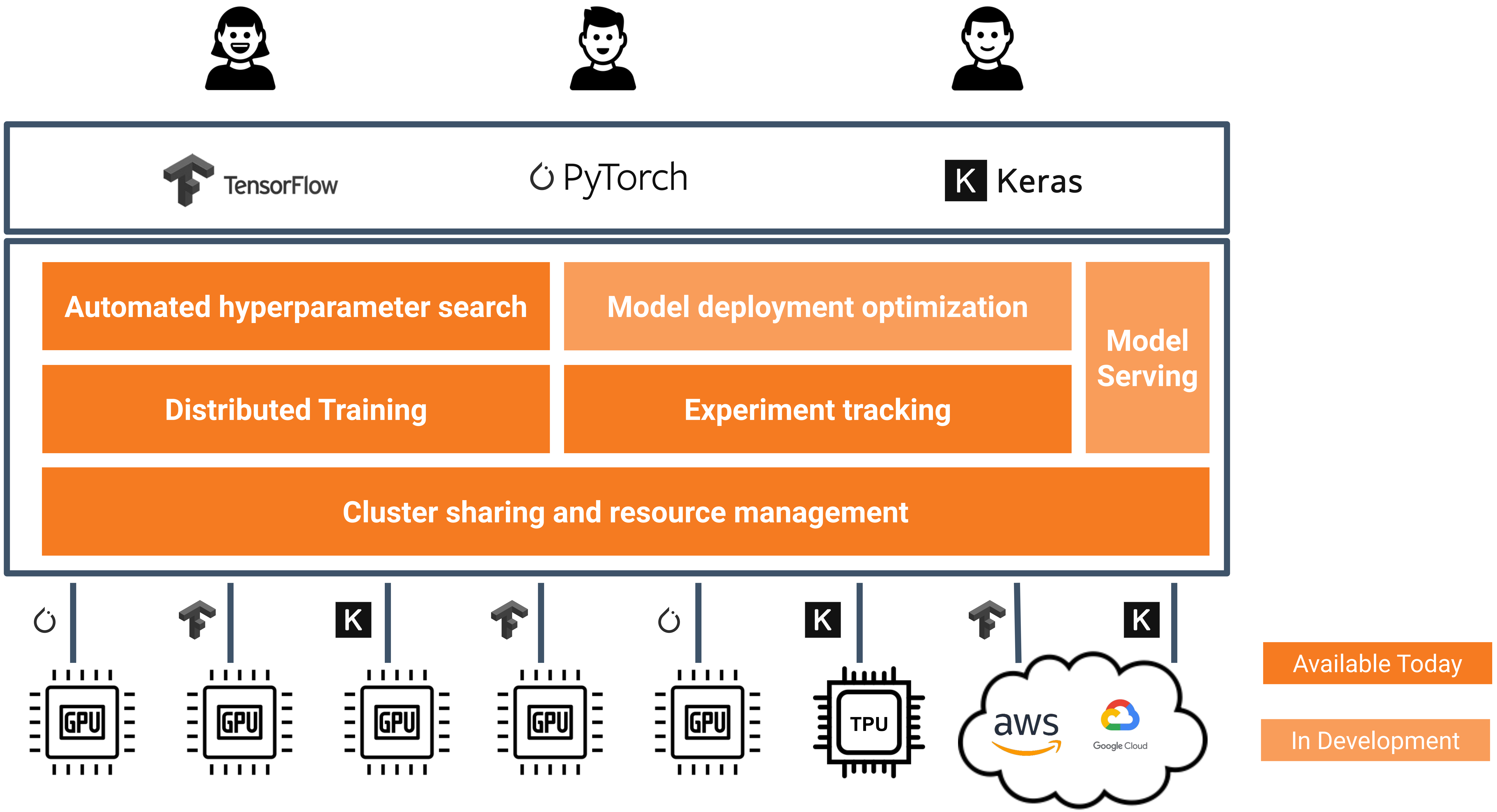
Repetitive and time-consuming work in training, e.g. hyperparameter search

Model deployment in production e.g., on edge devices

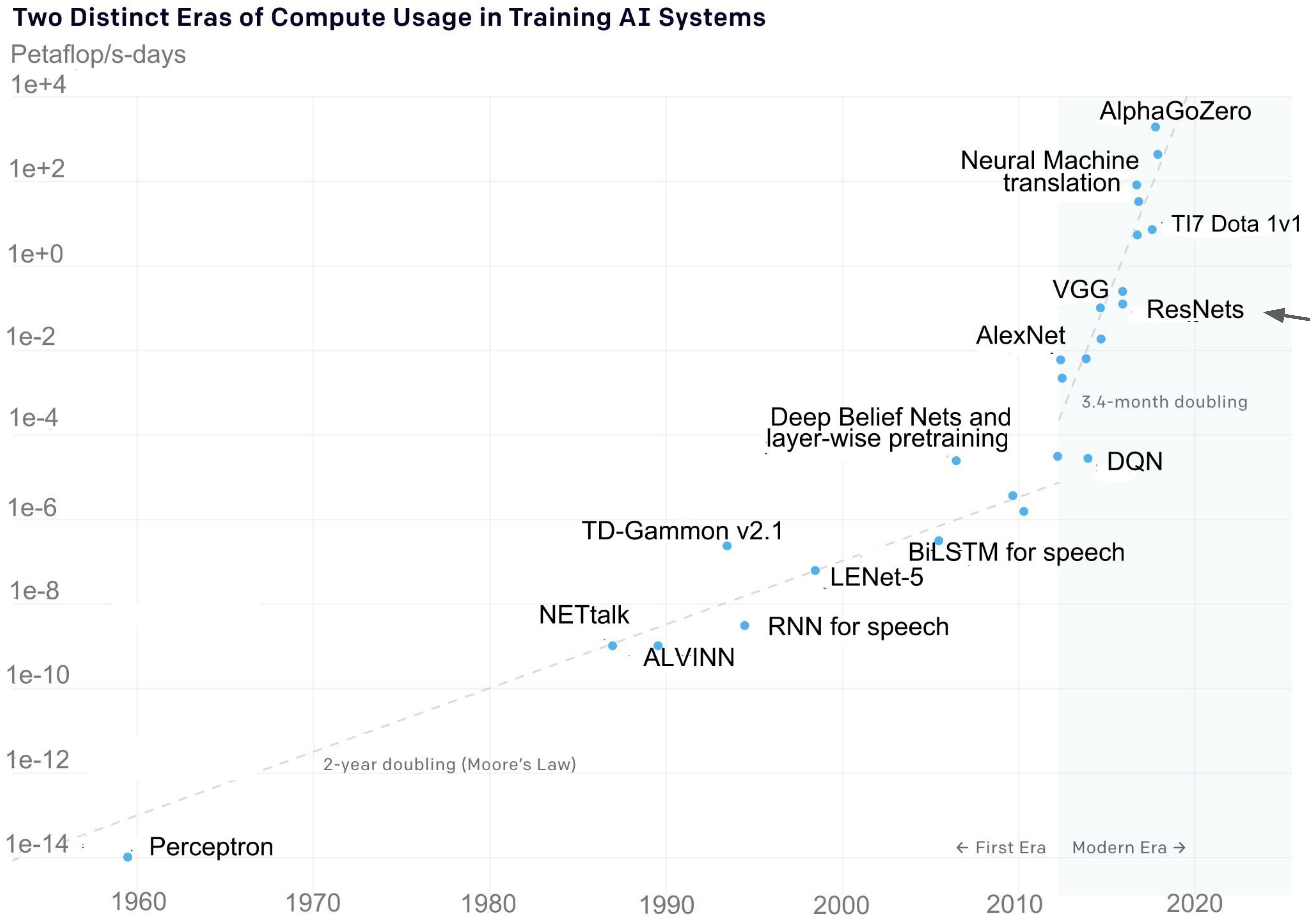
What kind of AI infrastructure is needed to address scalability?



Determined AI provides scalable deep learn infrastructure



Deep Learning is Computationally Expensive

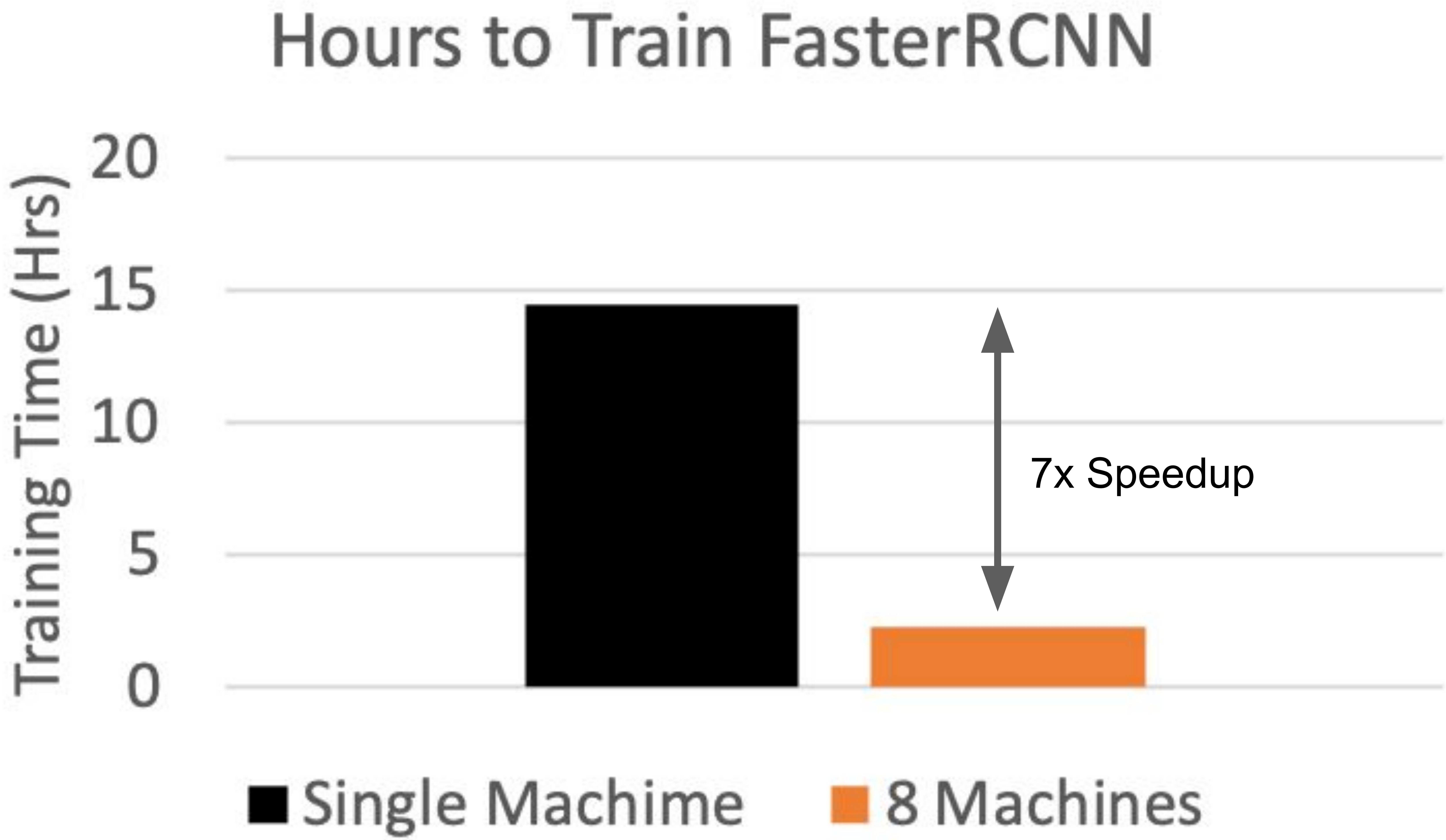


Need 1000+ GPUs to train in under an hour.

Training Time Impacts Productivity

- Experimenting with new datasets and/or models is bottlenecked by training time.
    - Imagine waiting for 2 days every time you compile your program.
- Limit on how fast a single machine can go.
    - E.g., Fine-tuning on a large dataset takes 7+ days on a single machine.

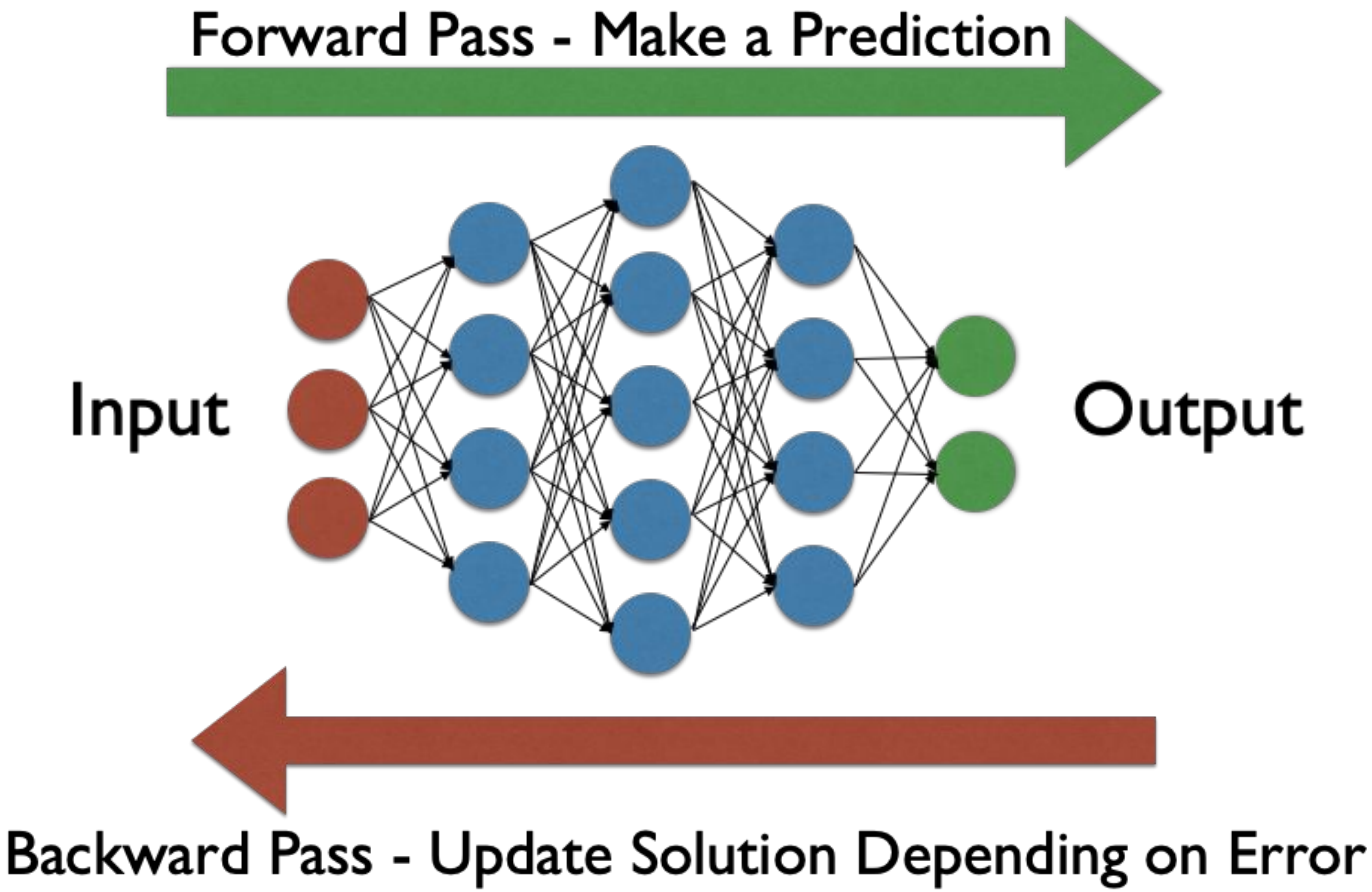
Distributed Training Reduces Training Time



- Machines: GCP w/ 8 V100 GPUs
- Dataset: Coco (2014)
- Target accuracy: 37.8% mAP

How does Distributed Training Work?

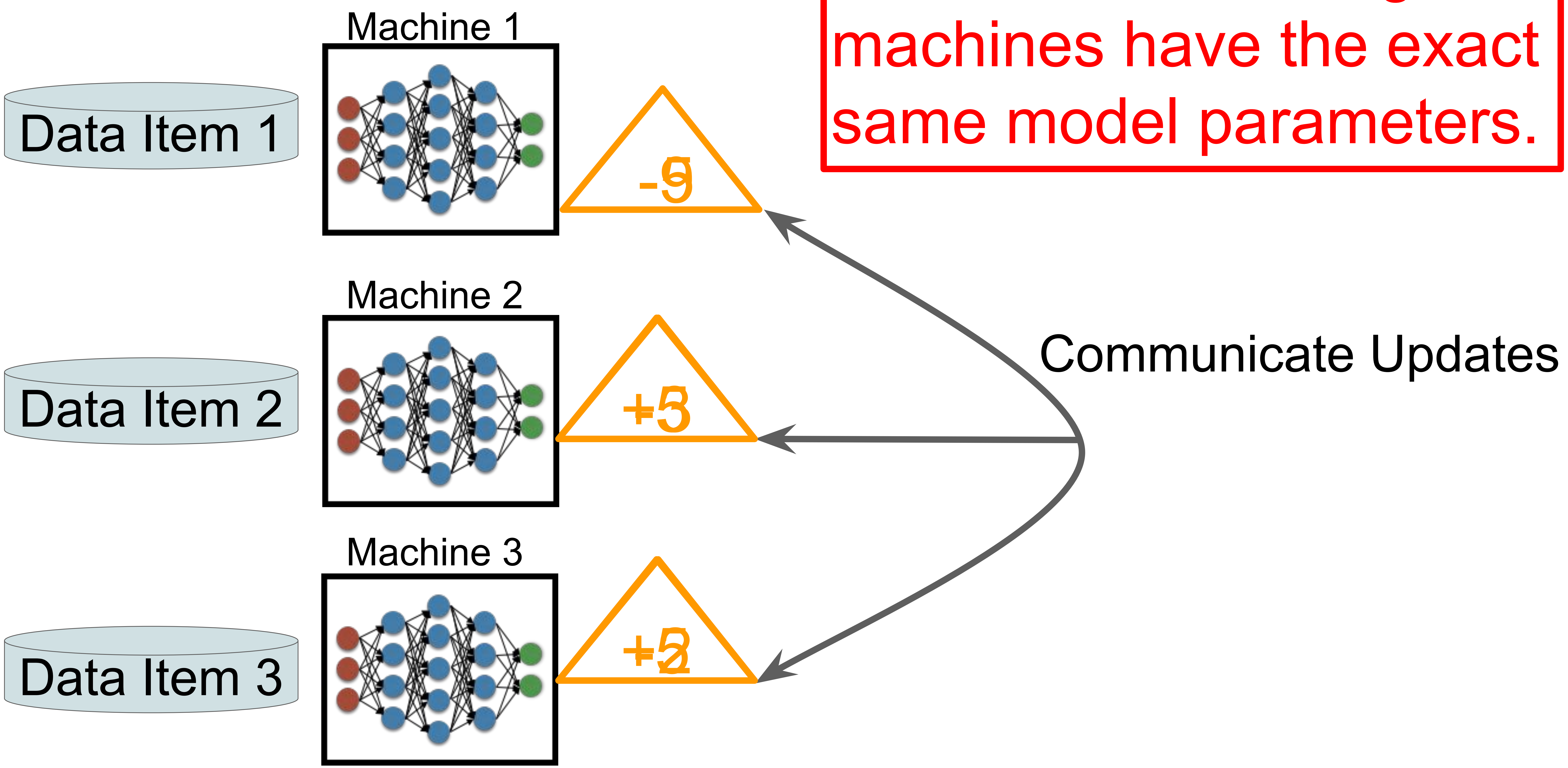
How Deep Learning Models Learn



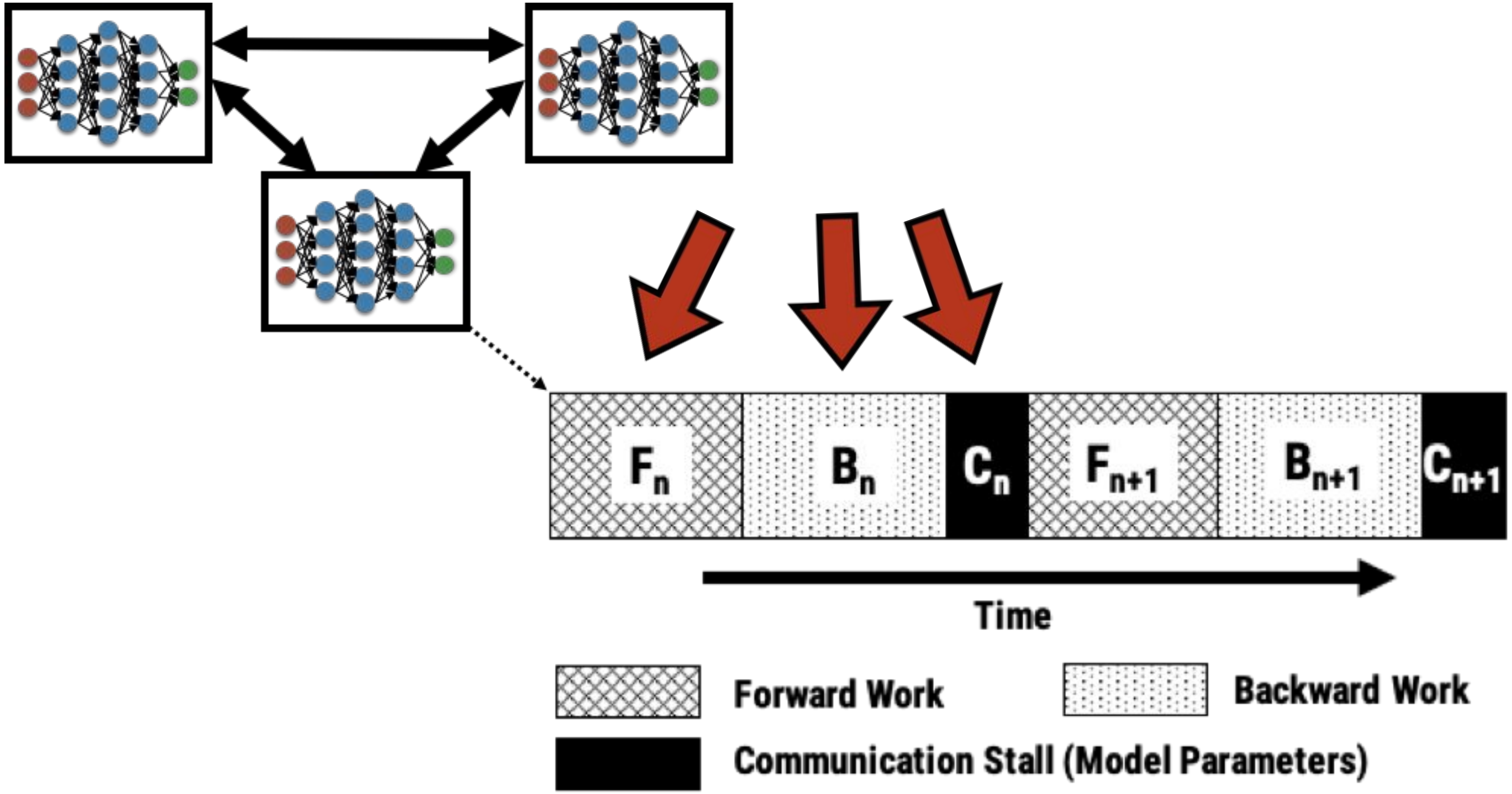
Distributed Training: Data Parallelism

- Data Parallelism is the most common technique for distributed training.
  - Separate copy of model on each machine

After communicating all machines have the exact same model parameters.



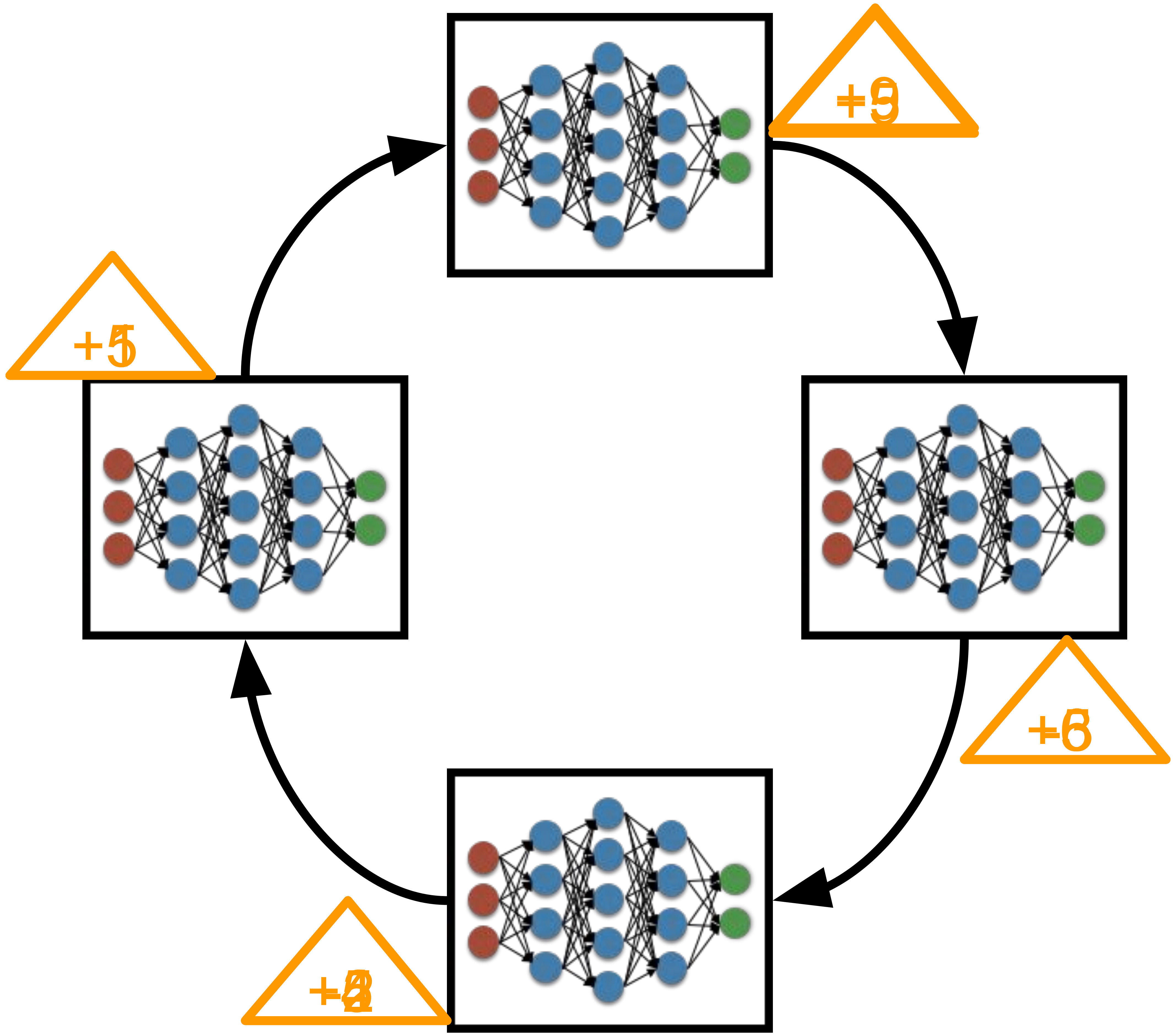
Overhead is from Communication



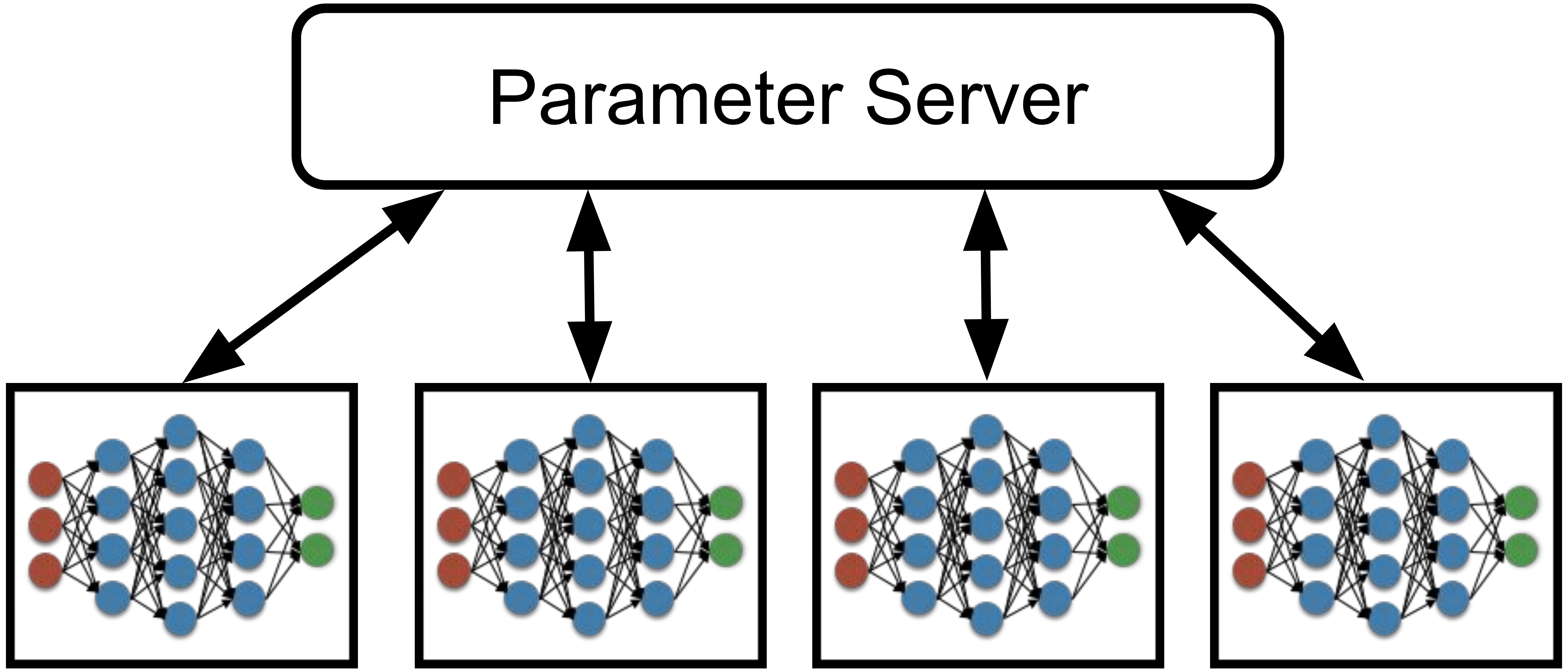
How is the Communication Performed?

Two Popular Approaches for Communicating Updates

Ring All-Reduce



Parameter Server



Ring All-Reduce vs. Parameter Server

- Ring All-Reduce
  - Most common approach (easier to implement today).
  - Optimal bandwidth usage.
- Parameter Server
  - More efficient for sparse updates (e.g., classical ML).
  - Better suited for supporting non-traditional network topologies.
  - Efficient support for a variety of synchronization schemes.

Two Categories of Challenges in Distributed Training

- Configuration Issues
  - Setting up machine connections.
  - Fault Tolerance.
  - Distributed data.

Current tools (TF, PyTorch) make you do this yourself.

- Performance Issues
  - Efficiently utilizing interconnects.
  - Reducing communication overhead.

Current tools (TF, PyTorch) don’t solve many of these.

Distributed Training Made Easy @ Determined AI

- Solve Configuration Issues
  - Automatic cluster configuration
  - Fault tolerance management
  - Data automatically partitioned
- Performance optimization
  - Reducing the communication overhead

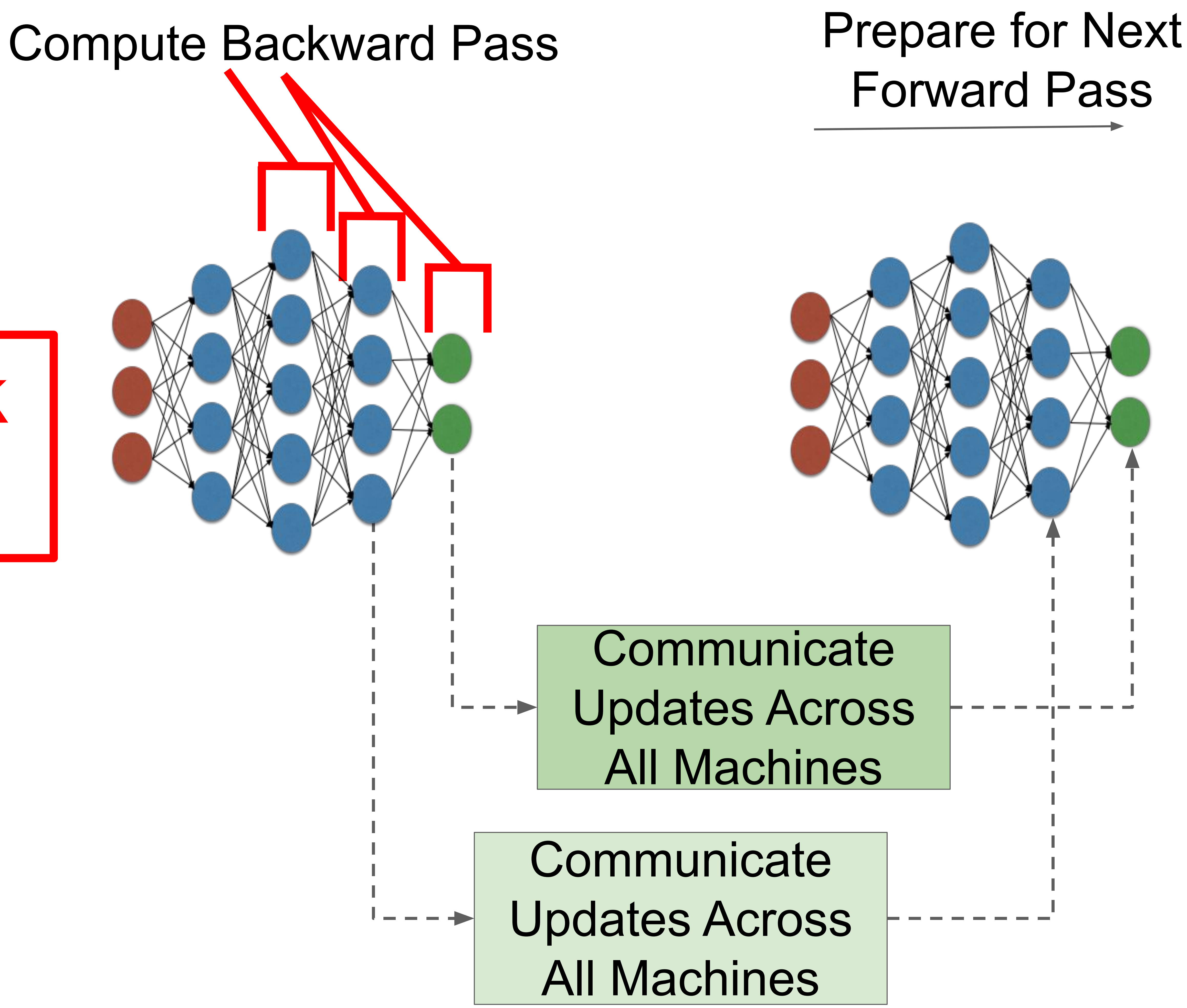
```
resources:
  gpus: 64
```

Leveraging Characteristics of Deep Learning

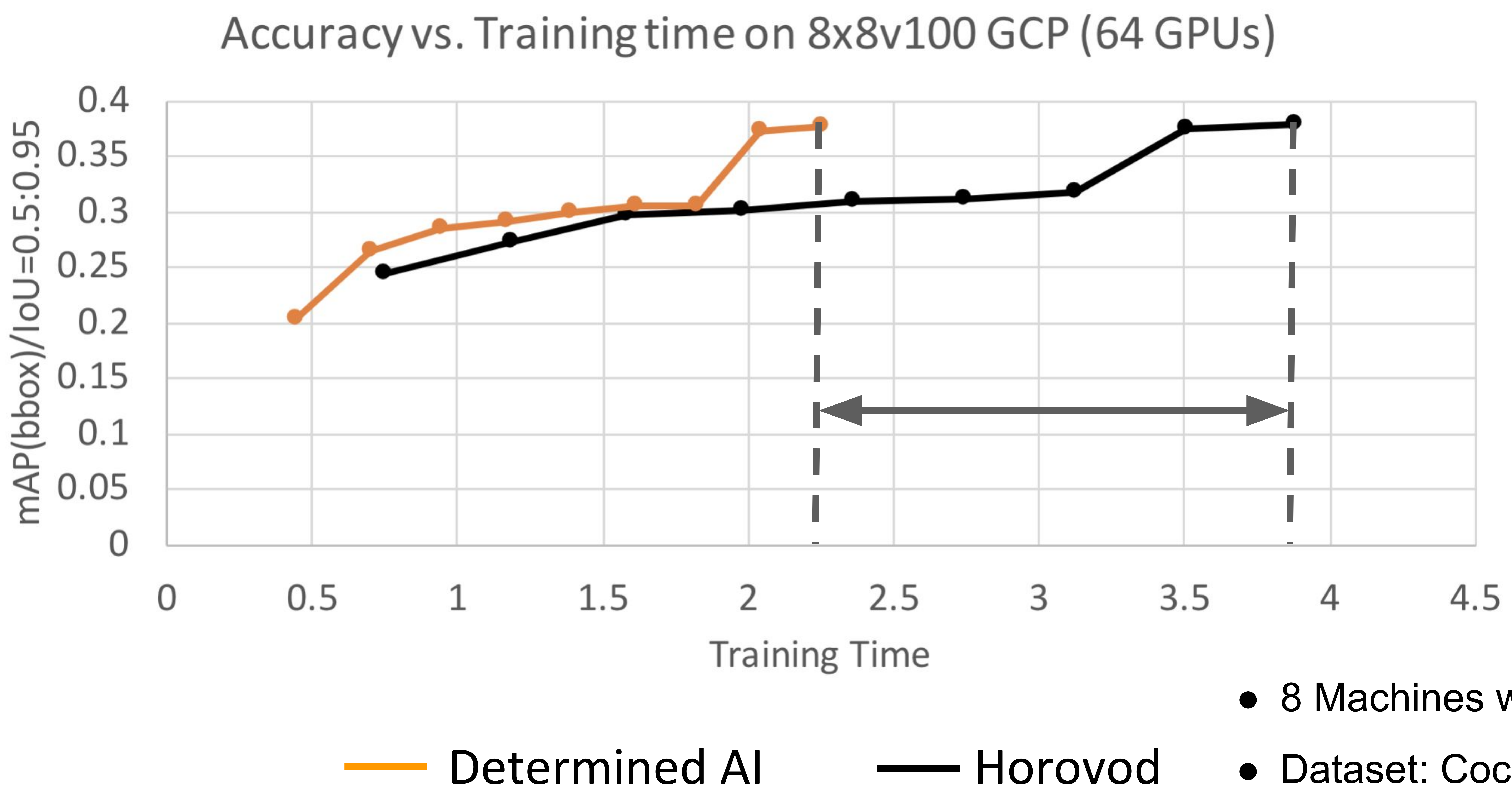
- Reducing communication is common problem in distributed systems.
- Deep Learning is a known workload, we can take advantage of this!
- Characteristics of deep learning workloads:
    - Backwards pass is computationally intensive and generates communication.
    - Iterative in nature.
    - Output of communication is reducible.

Overlapping Computation and Communication

Wait Free Back Propagation

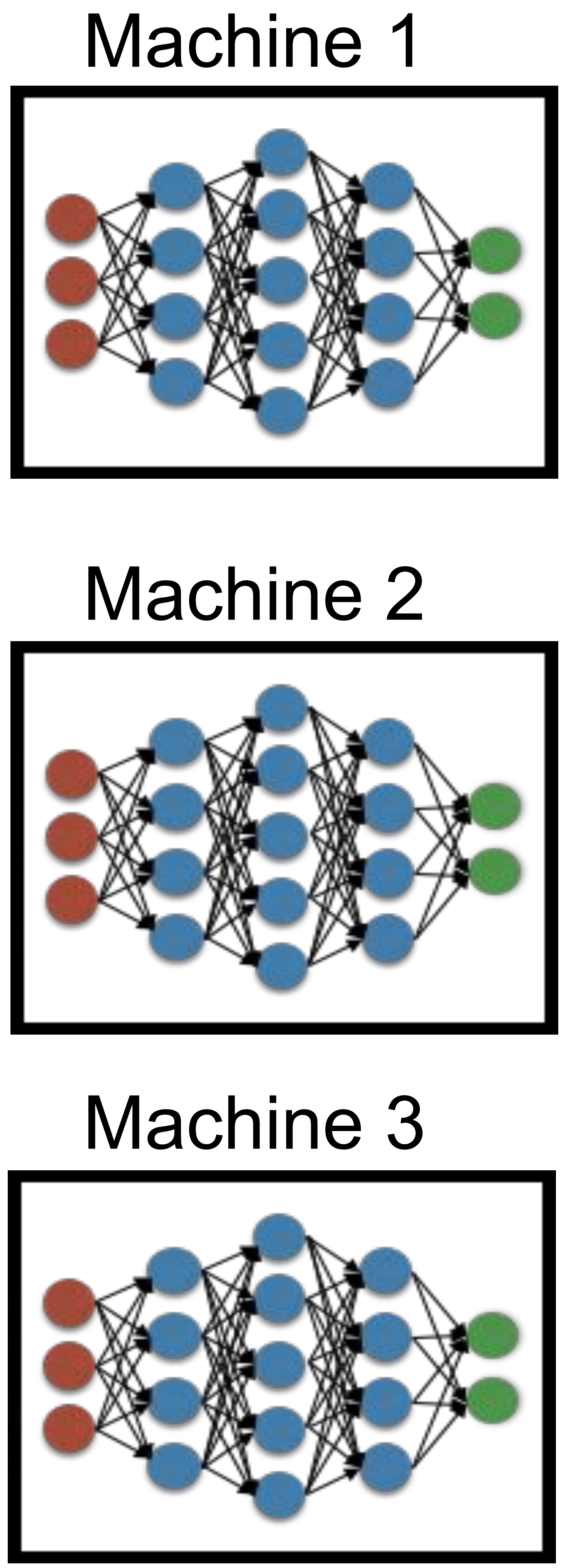


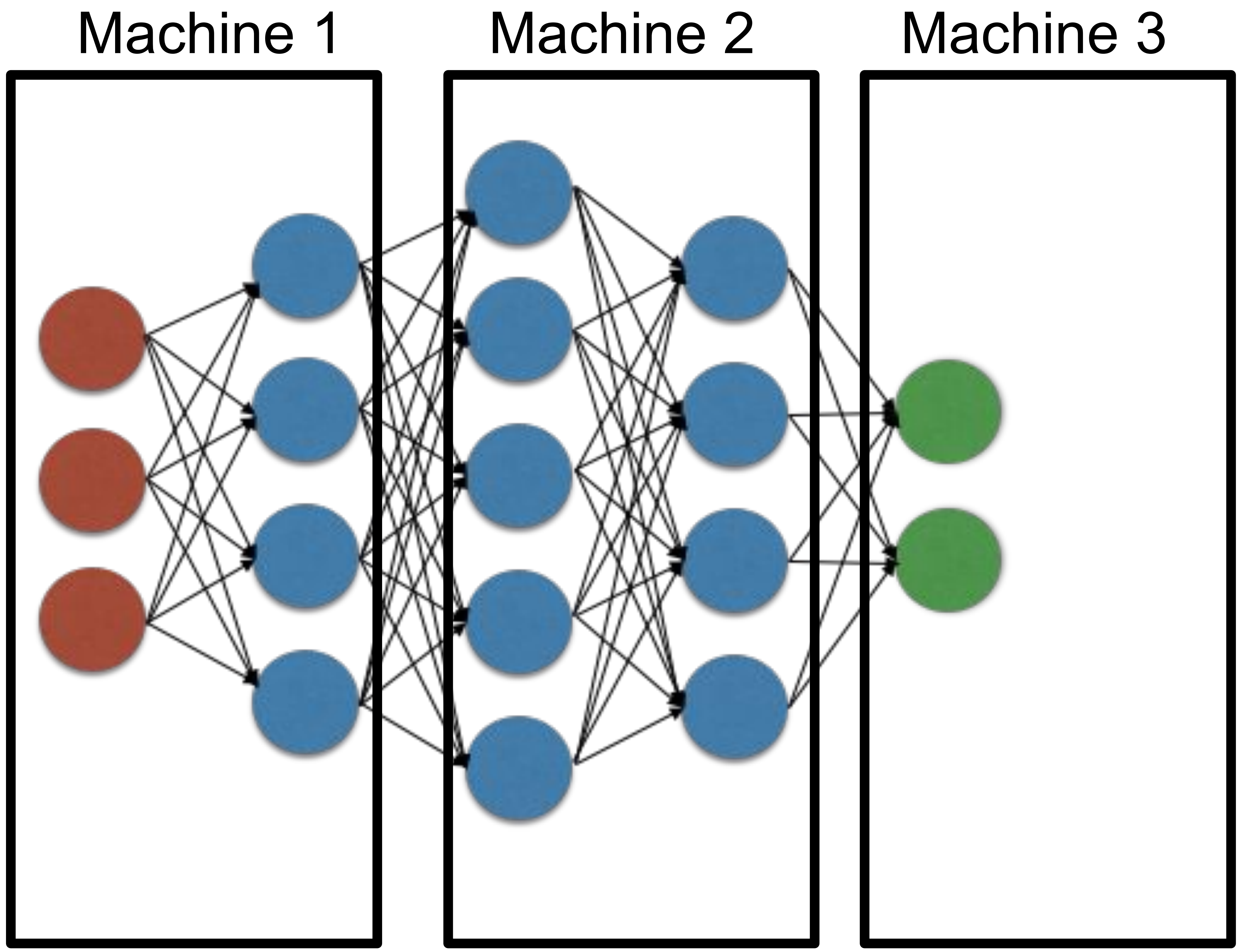
Up to 2X Faster than Horovod



- 8 Machines w/ 8 V100 GPUs each
- Dataset: Coco (2014)
- Target accuracy: 37.8% mAP

Wait there is more: Model Parallelism





Thank you

Learn more at determined.ai

chat with us at hello@determined.ai

Determined AI