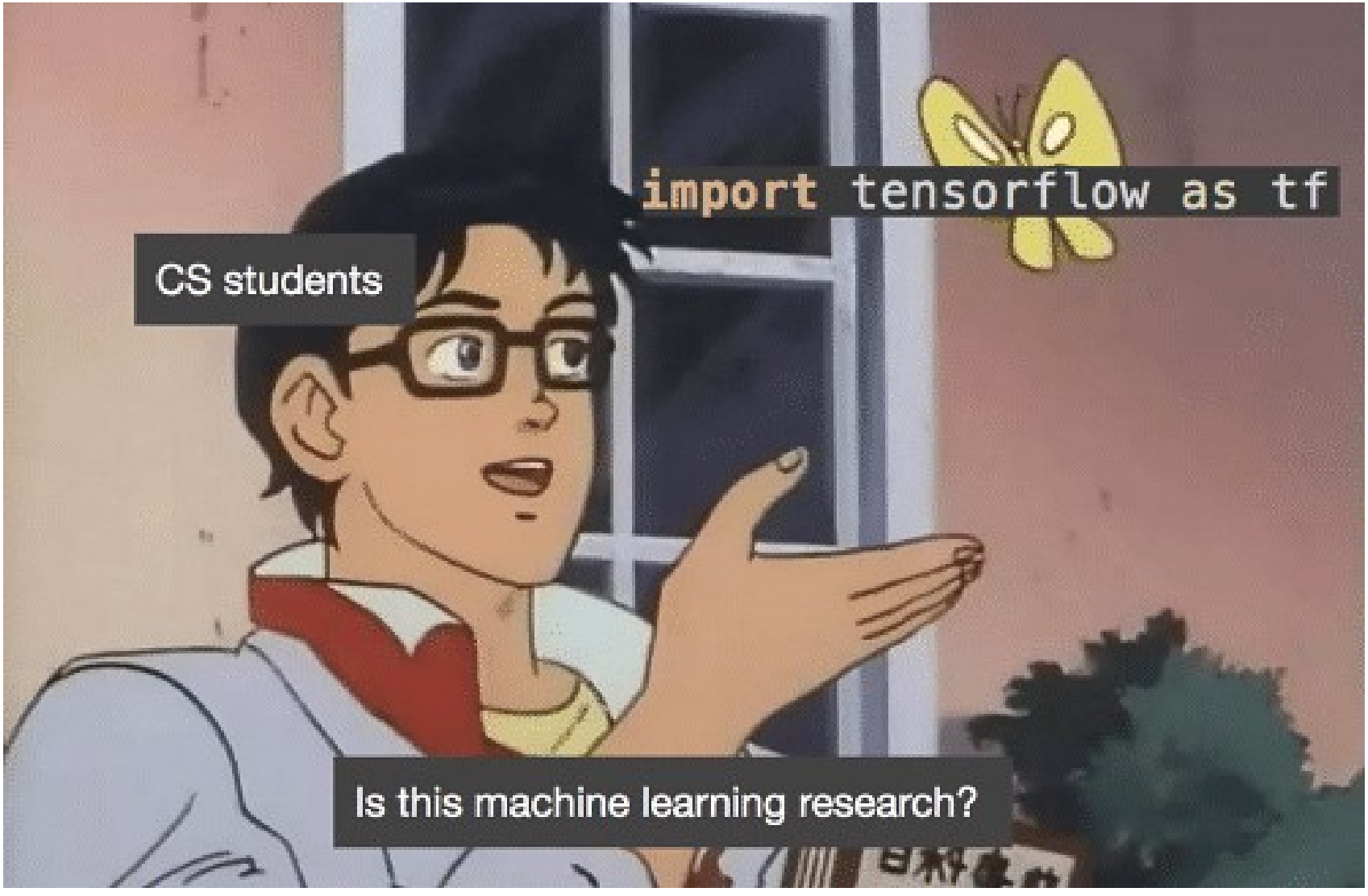
Distributed Machine Learning

Distributed Systems Spring 2019

Lecture 22

Obligatory ML is saving/destroying the world slide

Insert


import tensorflow as tf
CS students
Is this machine learning research?

Supervised Machine Learning In One Slide

- Goal: Fit a function over some data (x_i, y_i)
- Functions are *parametrized* : $f(w, x)$
- Example: Linear function $f = ax + b$, $w = (a, b)$
- How to find the parameters/weights that fit best?
- Loss function, $L = \sum_i error(y_i, f(w, x_i)) = \sum_i |y_i - f(w, x_i)|^2$
- Computational goal: Find w to minimize the loss function
- Deep neural networks: millions of weights

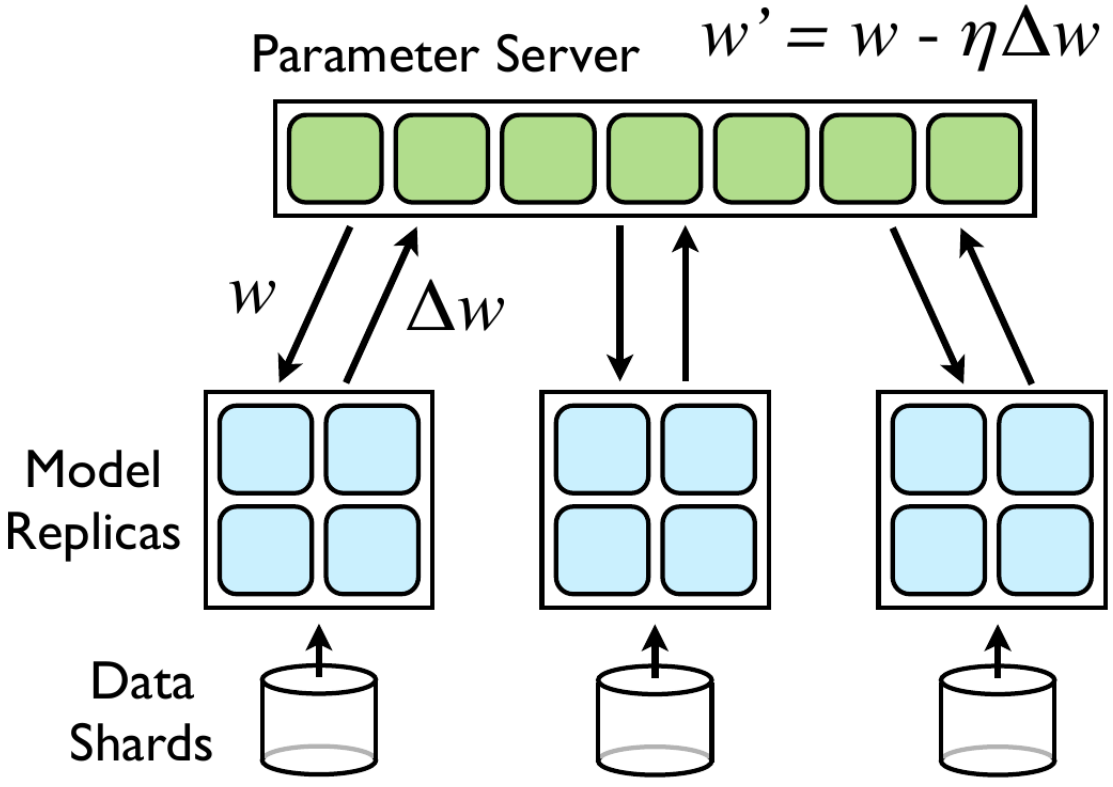
Distributed Machine Learning In One Slide

- What parallelism can we exploit?
- Basic idea: distribute the loss function computation
- Process 0 computes $min_w \sum_{i=0}^{N/2} |y_i - f(w, x_i)|^2$ and finds w_1
- Process 1 computes $min_w \sum_{i=N/2}^{N} |y_i - f(w, x_i)|^2$ and finds w_2
- This is **data parallelism**.
- Each process computes over a subset of the data (i.e., a map operation)
- How to reduce, i.e., combine w_1 and w_2?

Model Fitting With SGD

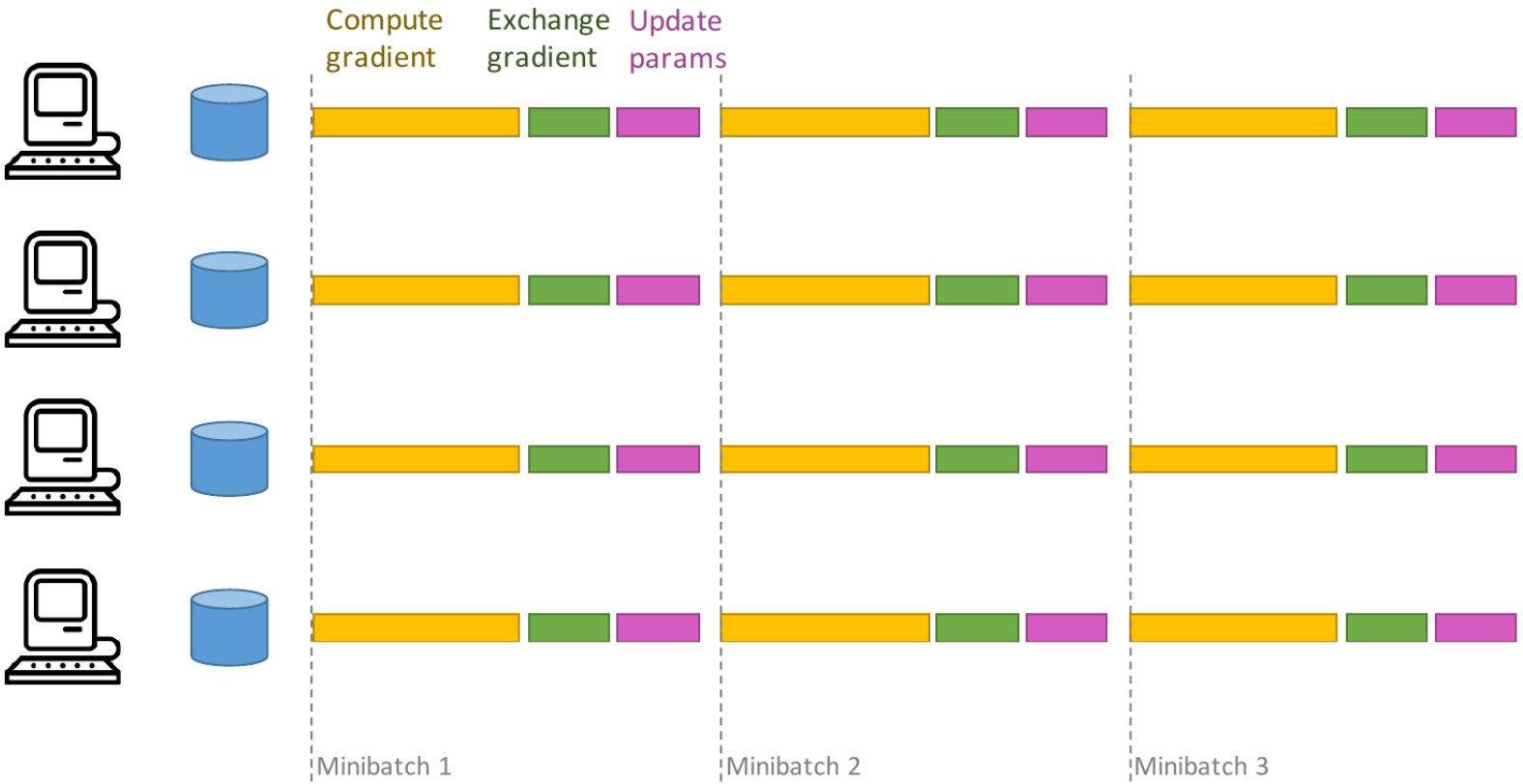
- Loss function, $L = \sum_i error(y_i, f(w, x_i)) = \sum_i |y_i - f(w, x_i)|^2$
- Central question: How to find w ?
- Most common technique: Gradient Descent
- $w_{t+1} = w_t - \eta \Delta L(w_t)$
- $\Delta L(w_t)$ is the gradient found using partial derivatives
- Stochastic gradient descent (SGD): Evaluate gradient on only subset of data points
- Also called a "mini-batch"

Parameter Server Architecture

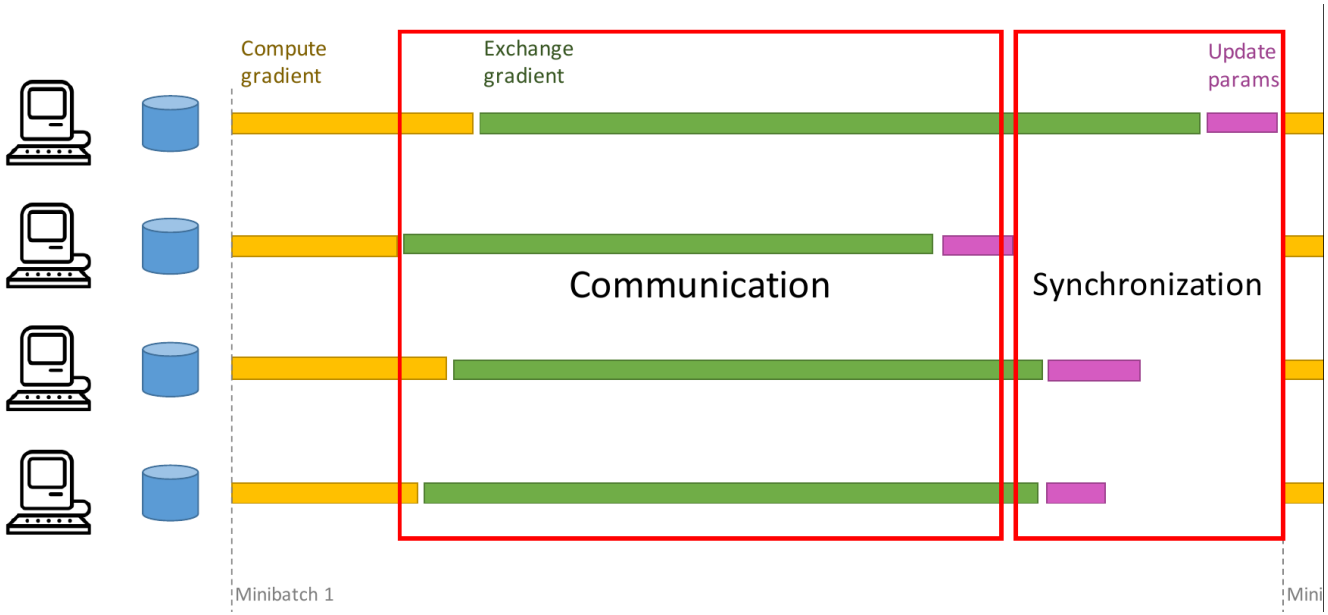


- Workers compute gradients using SGD on small data batches
- New gradients shared with other workers via parameter server
- Other ways of communication: all-to-all reduce,...

Data Parallel SGD Timeline



Synchronous SGD



- Synchronous SGD uses a barrier.

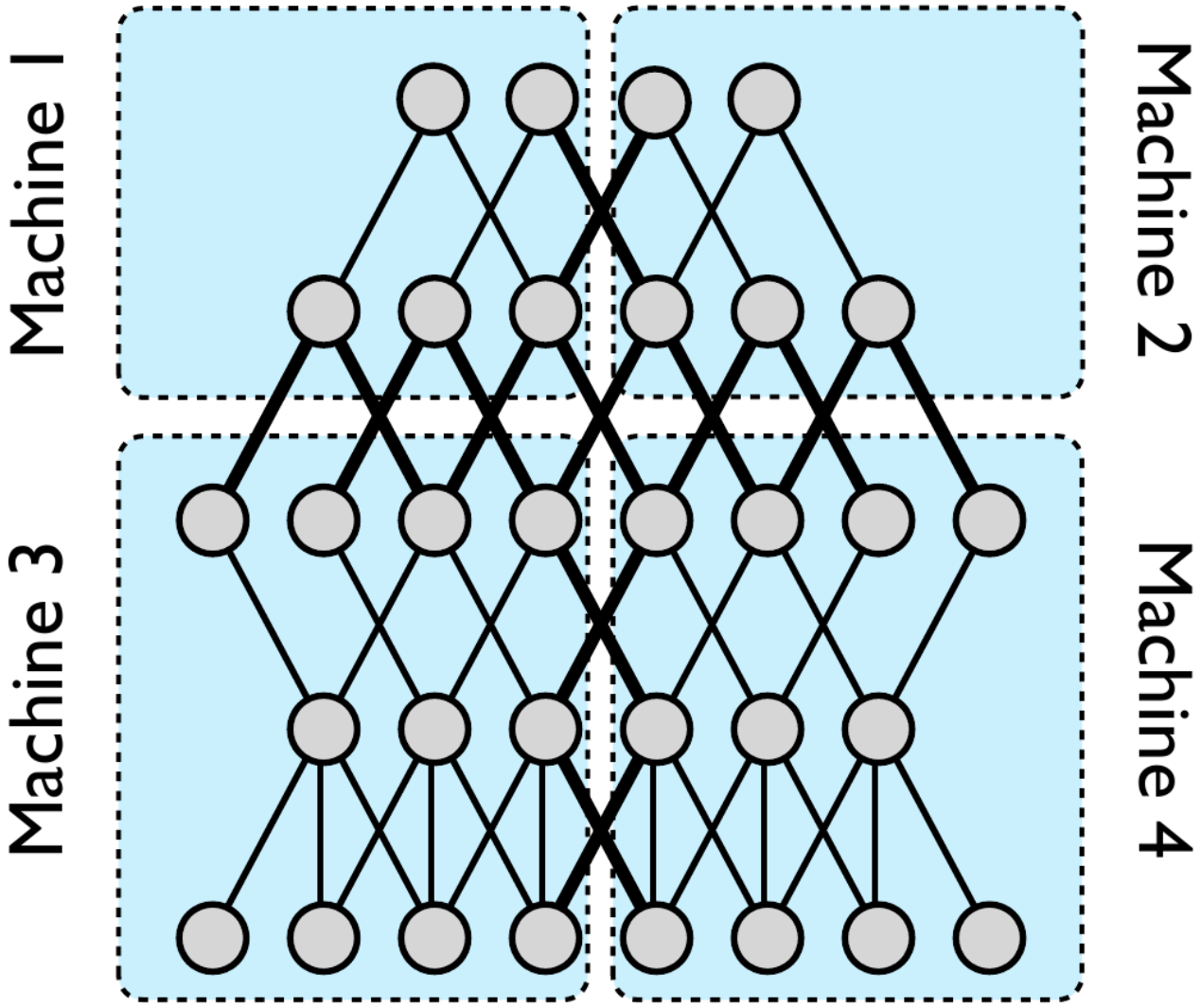
HogWild!

- Asynchronous parallel SGD
- Typical way: processes compute gradients, and update shared gradient array.
- Gradient array protected using a lock
- HogWild! is a lock-free technique
- Key idea: Let processes update their gradients *without* locking
- Works well if updates are *sparse*

Consistency vs. Performance

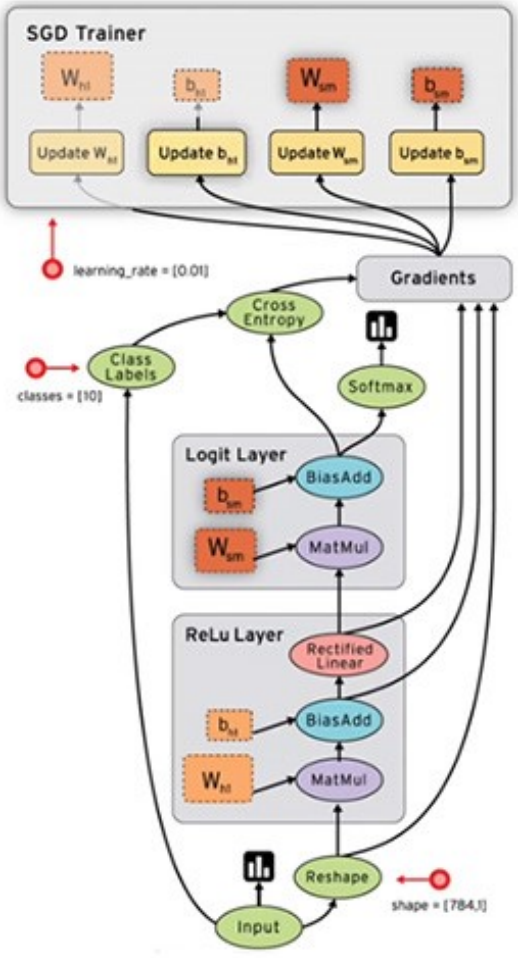
- Synchronous SGD has synchronization overheads
- But, Async SGD suffers from poor *statistical efficiency*
- Because of updating models with stale weights
- Synchronous: Per-iteration time is longer
- Async: Per-iteration time is shorter, but more iterations required

Model Parallelism



- Alternative is to replicate data on all machines, but split model

TensorFlow



- Computation structured as a dataflow graph
- TF programs create graphs (ala Spark)
- TF engine evaluates graph
- Many inbuilt operation kernels
- Dataflow graphs can be partitioned in many ways
- Supports model and data parallelism, and hybrid schemes

Some Open Questions

- Sync vs. Async
- How many parameter servers vs. workers?
- Role of GPUs
- Geo-distributed training: how to minimize updates?
- Accelerate inference (evaluate $f(w, x)$

References

Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. Tal Ben-Nun, Torsten Hoefler

END