CPSC 436C
Cloud Computing for Data Science

Distributed Machine Learning

Maryam R.Aliabadi

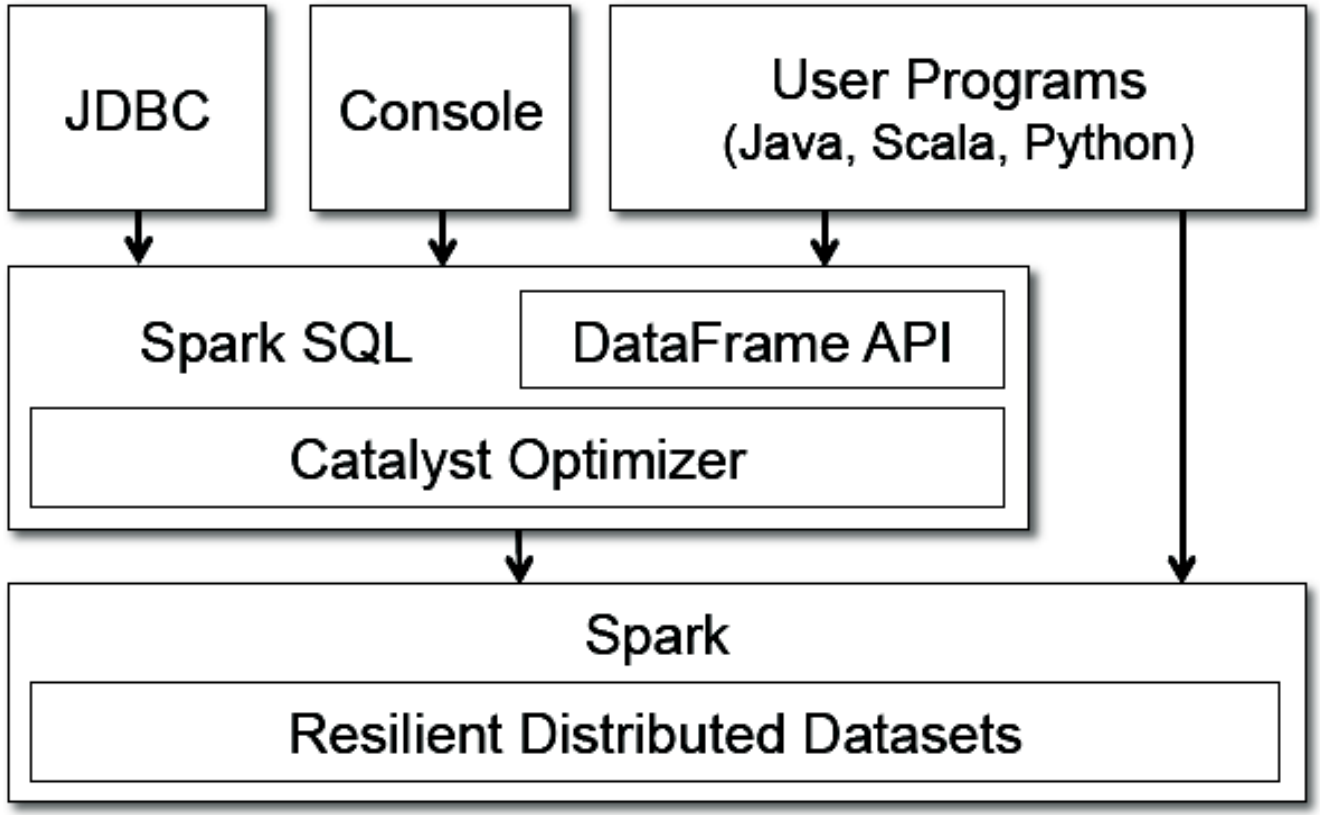
mraiyata@cs.ubc.ca

Spring 2024

Last Week‘s Review

- Structured Data processing using SparkSQL
- RDD vs. DataFrame vs. DataSet
- Logical and physical plans

Spark SQL

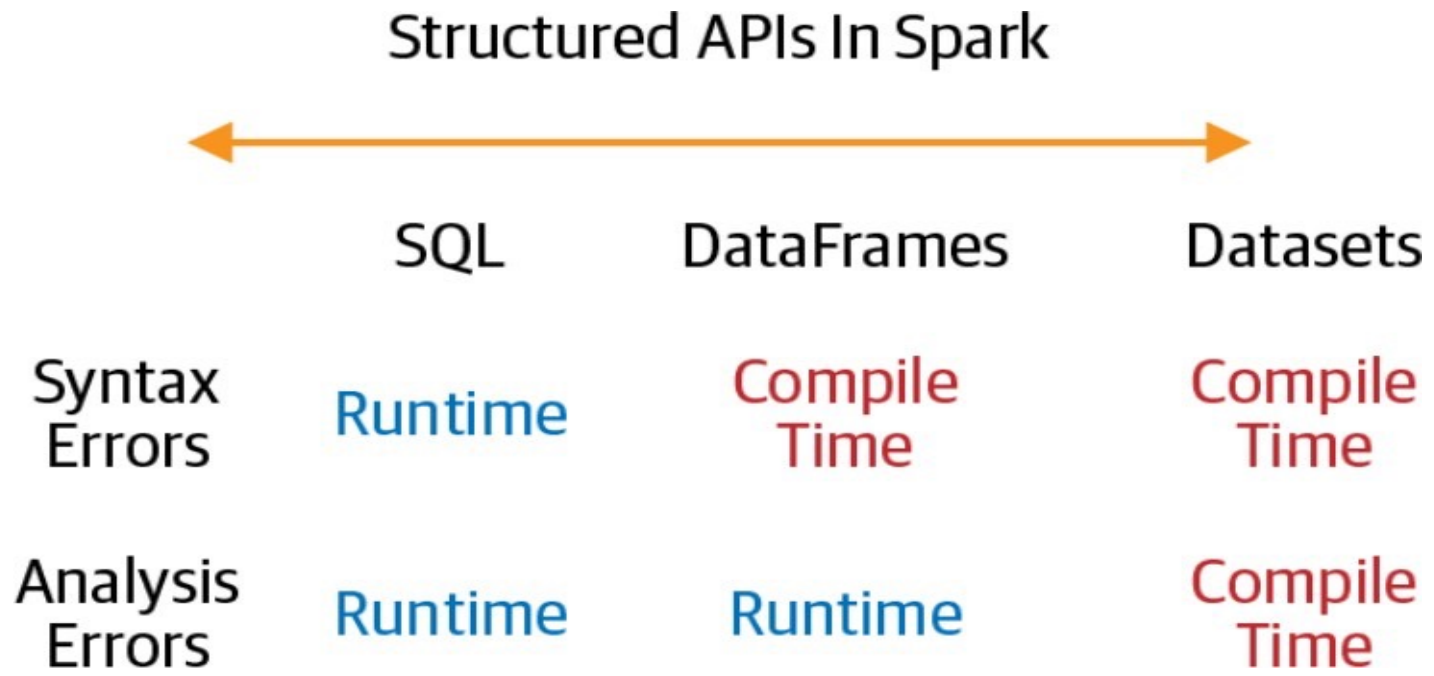

JDBC
Console
User Programs
(Java, Scala, Python)
Spark SQL
DataFrame API
Catalyst Optimizer
Spark
Resilient Distributed Datasets

DataFrames and Datasets

- Spark has two notions of structured collections:
  - DataFrames
  - Datasets

- They are distributed table-like collections with well-defined rows and columns.
- They represent immutable lazily evaluated plans.
- When an action is performed on them, Spark performs the actual transformations and return the result.

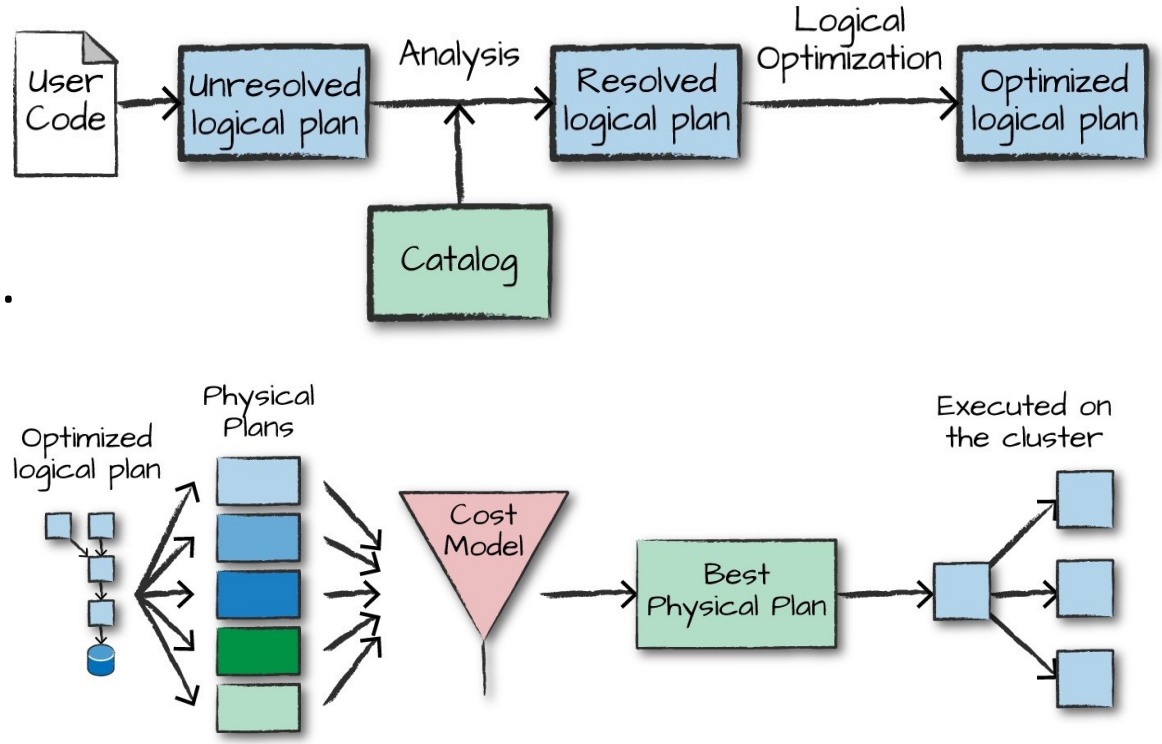
Structured API in Spark



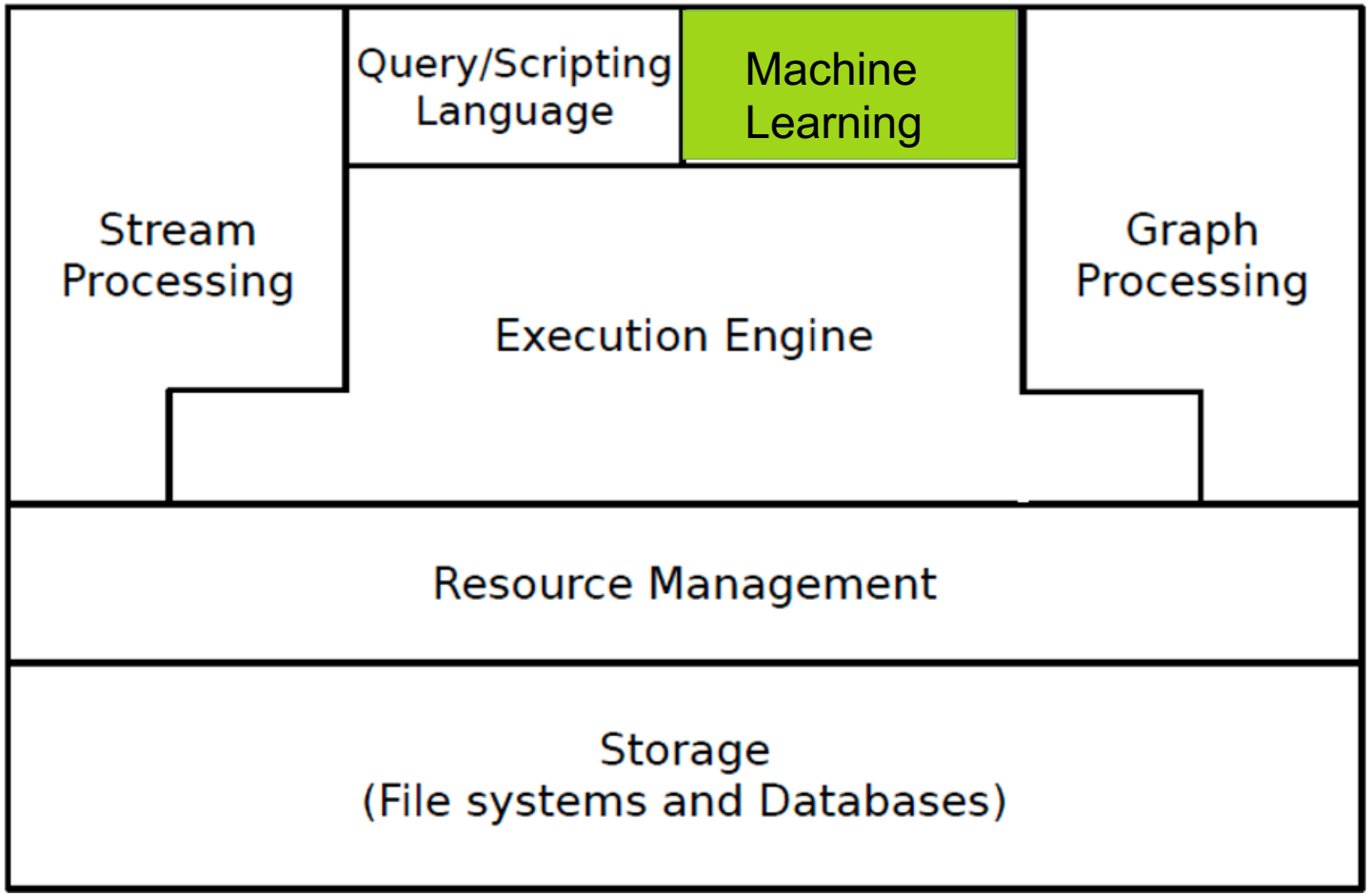
Spark SQL; The Whole Story

- Create and Run Spark Programs Faster:

1. Write less code.
2. Read less data.
3. Let the optimizer do the hard work.



Today's Topic



Machine Learning

What is Machine Learning?

That is roughly the problem that Machine Learning addresses!

Data and Knowledge

- Is this email spam or no spam?
- Is there a face in this picture?
- Should I lend money to this customer given his spending behavior?

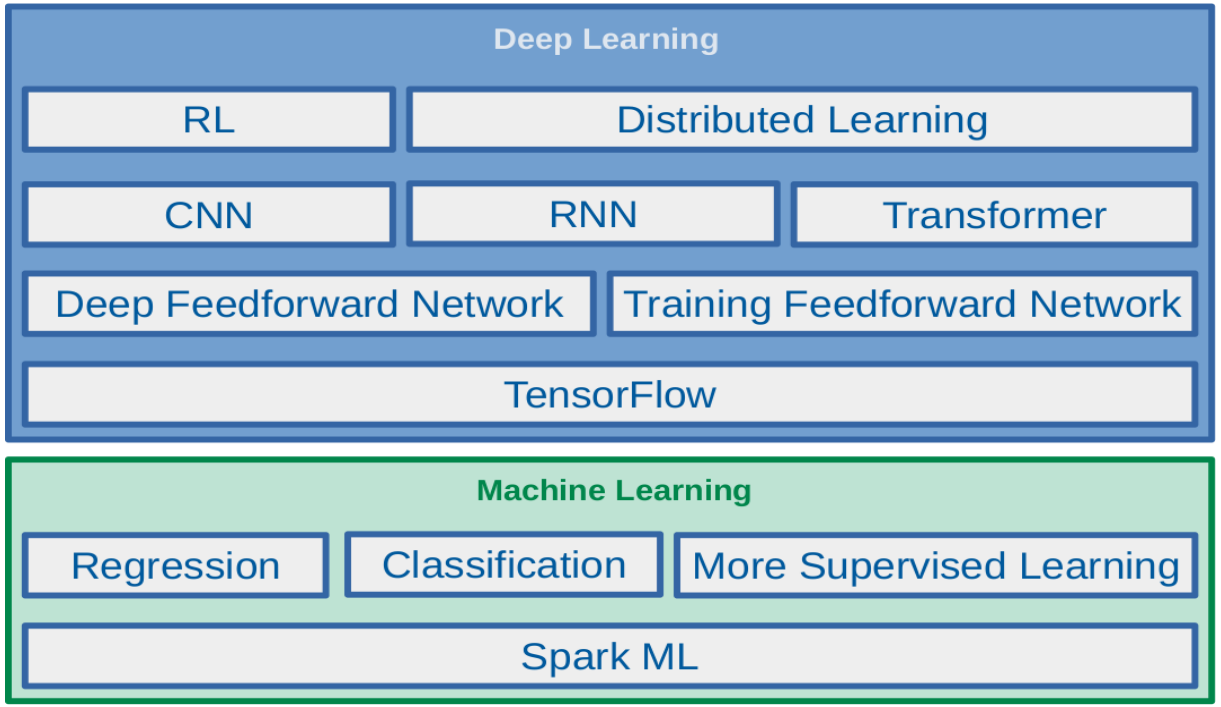
Data and Knowledge

- Knowledge is not concrete
- Spam is an abstraction
- Face is an abstraction
- Who to lend to is an abstraction

- You do not find spam, faces, and financial advice in datasets, you just find bits!

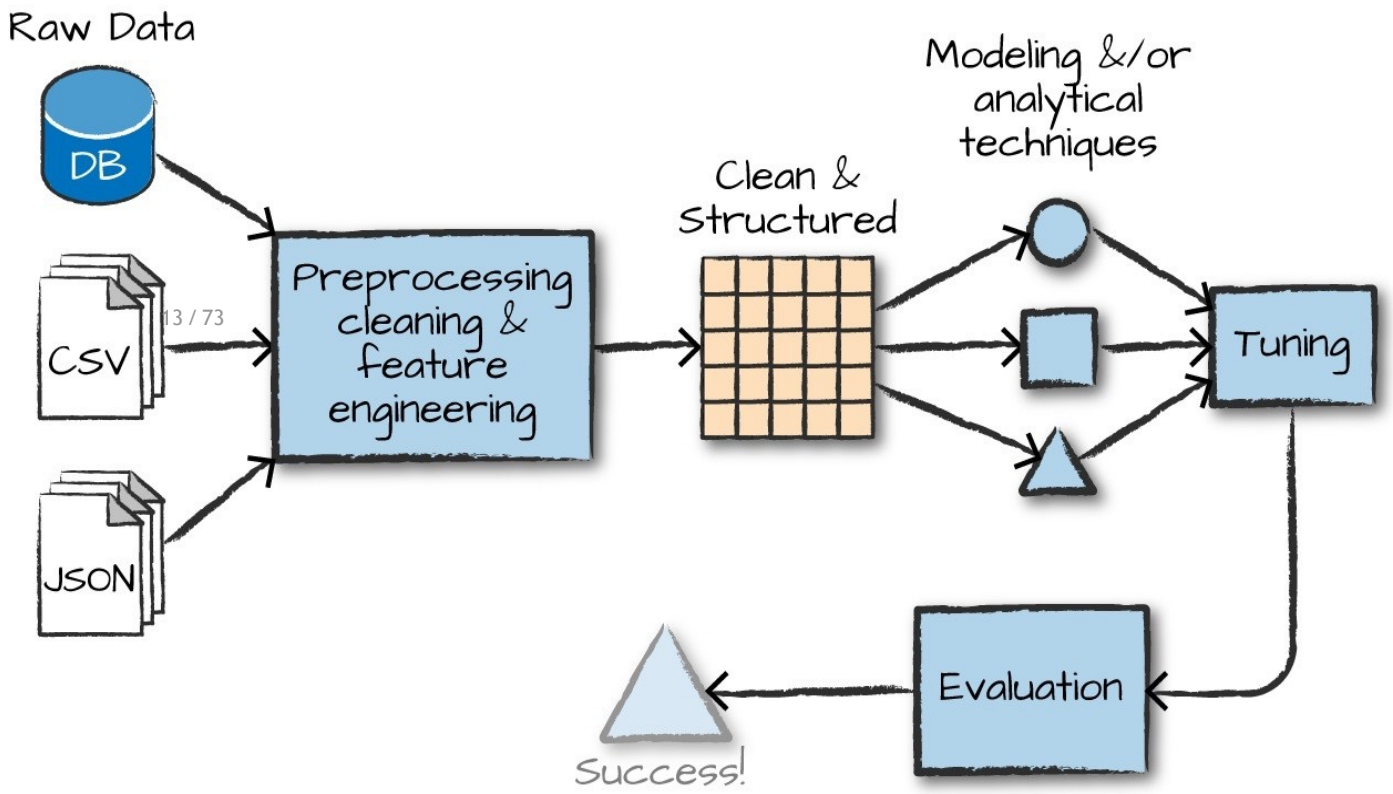
Machine Learning with Spark

- Spark provides support for statistics and machine learning.
  - Supervised learning
  - Unsupervised engines
  - Deep learning



The Advanced analytical process

- Data collection
- Data cleaning
- Feature engineering
- Training models
- Model tuning and evaluation



What is MLlib?

- MLlib is a package built on Spark.
- It provides interfaces for:
  - Gathering and cleaning data
  - Feature engineering and feature selection
  - Training and tuning large-scale supervised and unsupervised machine learning models
  - Using those models in production

What is MLlib?

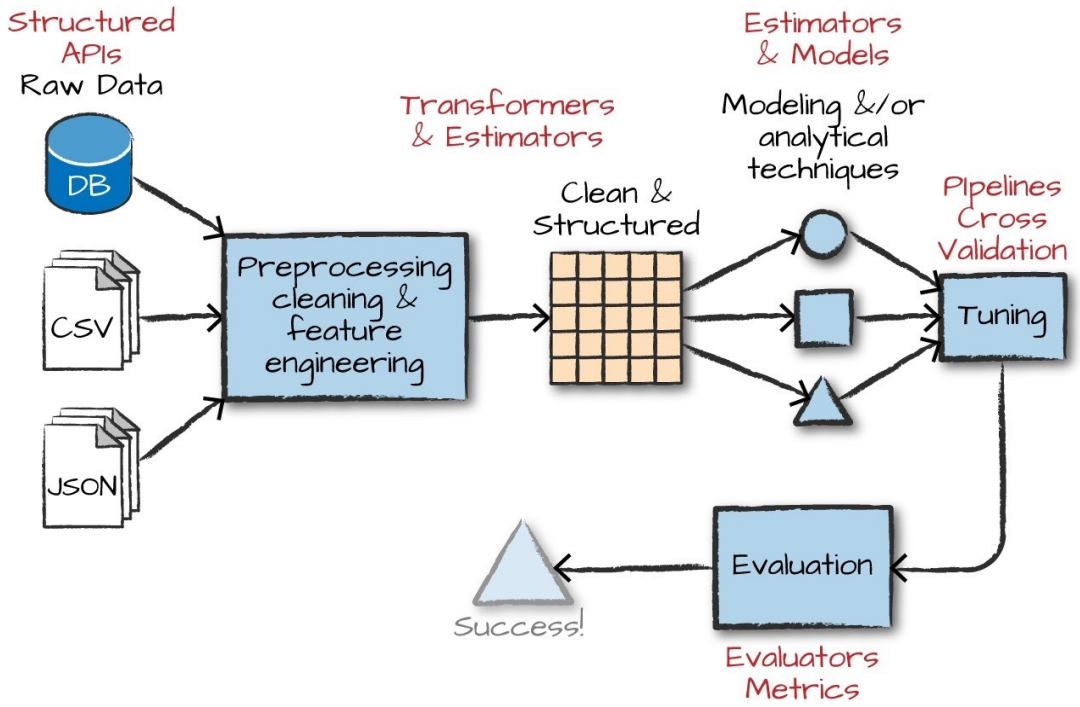
- MLlib consists of two packages.

- org.apache.spark.mllib
  - Uses RDDs
  - It is in maintenance mode (only receives bug fixes, not new features)

- org.apache.spark.ml
  - Uses DataFrames
  - Offers a high-level interface for building machine learning pipelines

High level MLlib concept

- ML pipelines (spark.ml) provide a uniform set of high-level APIs built on top of DataFrames to create machine learning pipelines.



Pipeline

- Pipeline is a sequence of algorithms to process and learn from data.

- E.g., a text document processing workflow might include several stages:

  - Split each document's text into words.
  - Convert each document's words into a numerical feature vector.
  - Learn a prediction model using the feature vectors and labels.

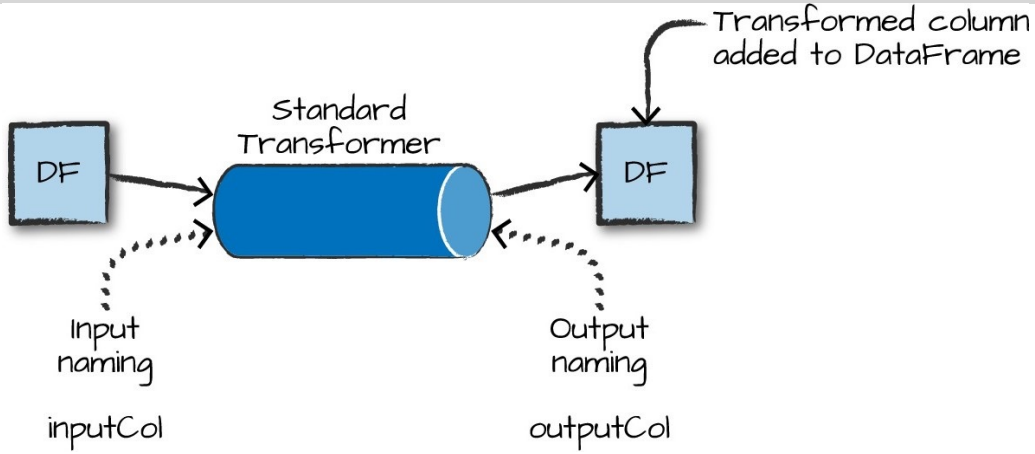
- Main pipeline components: transformers and estimators.

Transformers

- Transformers take a DataFrame as input and produce a new DataFrame as output.
- The class `Transformer` implements a method `transform()` that converts one DataFrame into another.

```
// transformer: DataFrame =[transform]=> DataFrame

transform(dataset: DataFrame): DataFrame
```

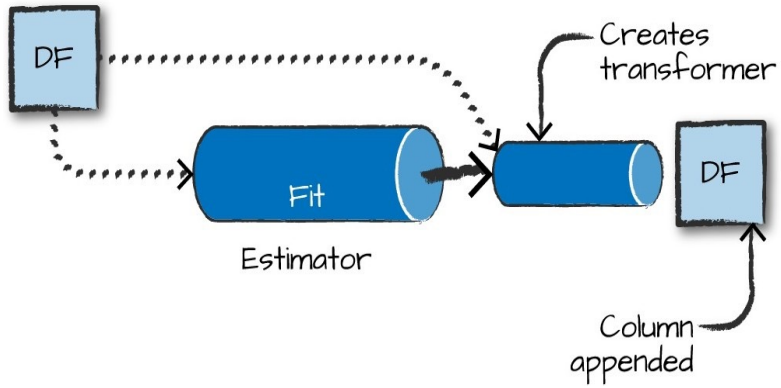


Estimators

- Estimator is an abstraction of a learning algorithm that fits a model on a dataset.
- The class `Estimator` implements a method `fit()`, which accepts a DataFrame and produces a Model (Transformer).

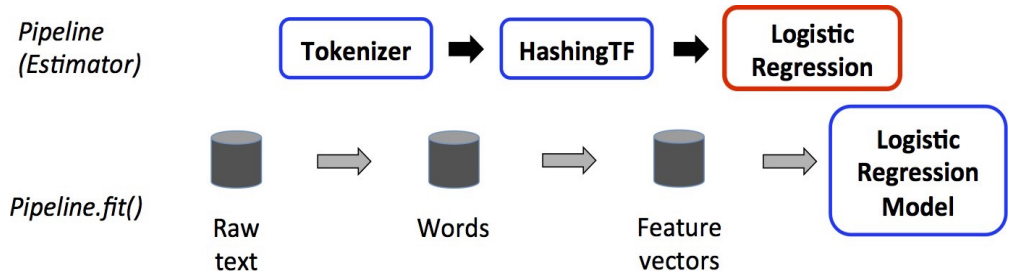
```
// estimator: DataFrame =[fit]=> Model

fit(dataset: DataFrame): M
```



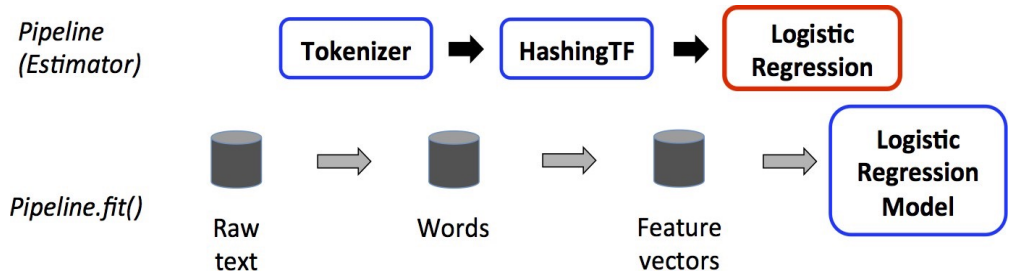
How does pipeline work?

- A pipeline is a sequence of stages.
- Stages of a pipeline run in order.
- The input DataFrame is transformed as it passes through each stage.
- Each stage is either a Transformer or an Estimator.
- E.g., a Pipeline with three stages: Tokenizer and HashingTF are Transformers, and
- LogisticRegression is an Estimator.



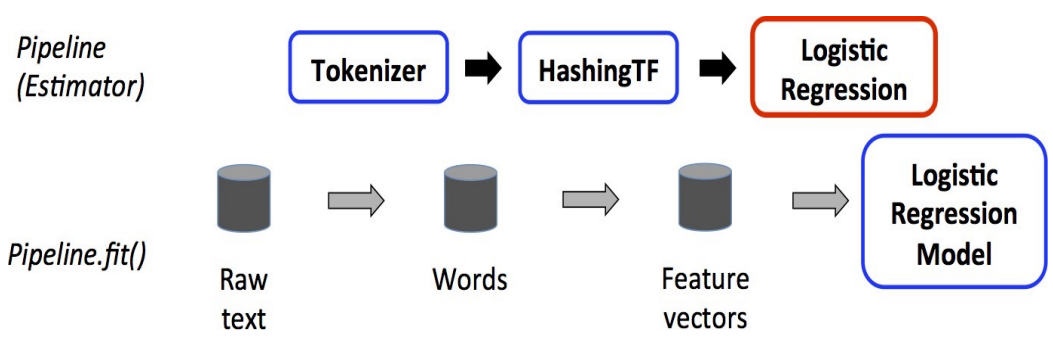
How does pipeline work?

- `Pipeline.fit()`: is called on the original DataFrame
  - DataFrame with raw text documents and labels
- `Tokenizer.transform()`: splits the raw text documents into words
  - Adds a new column with words to the DataFrame
- `HashingTF.transform()`: converts the words column into feature vectors
  - Adds new column with those vectors to the DataFrame
- `LogisticRegression.fit()`: produces a model (LogisticRegressionModel).

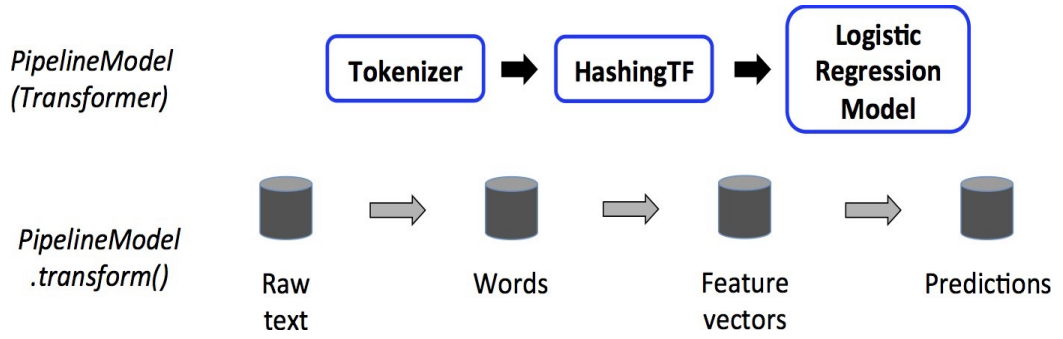


How does pipeline work?

- A Pipeline is an Estimator (DataFrame =[fit]=> Model).
- After a Pipeline's fit() runs, it produces a PipelineModel.
- PipelineModel is a Transformer (DataFrame =[transform]=> DataFrame).
- The PipelineModel is used at test time.



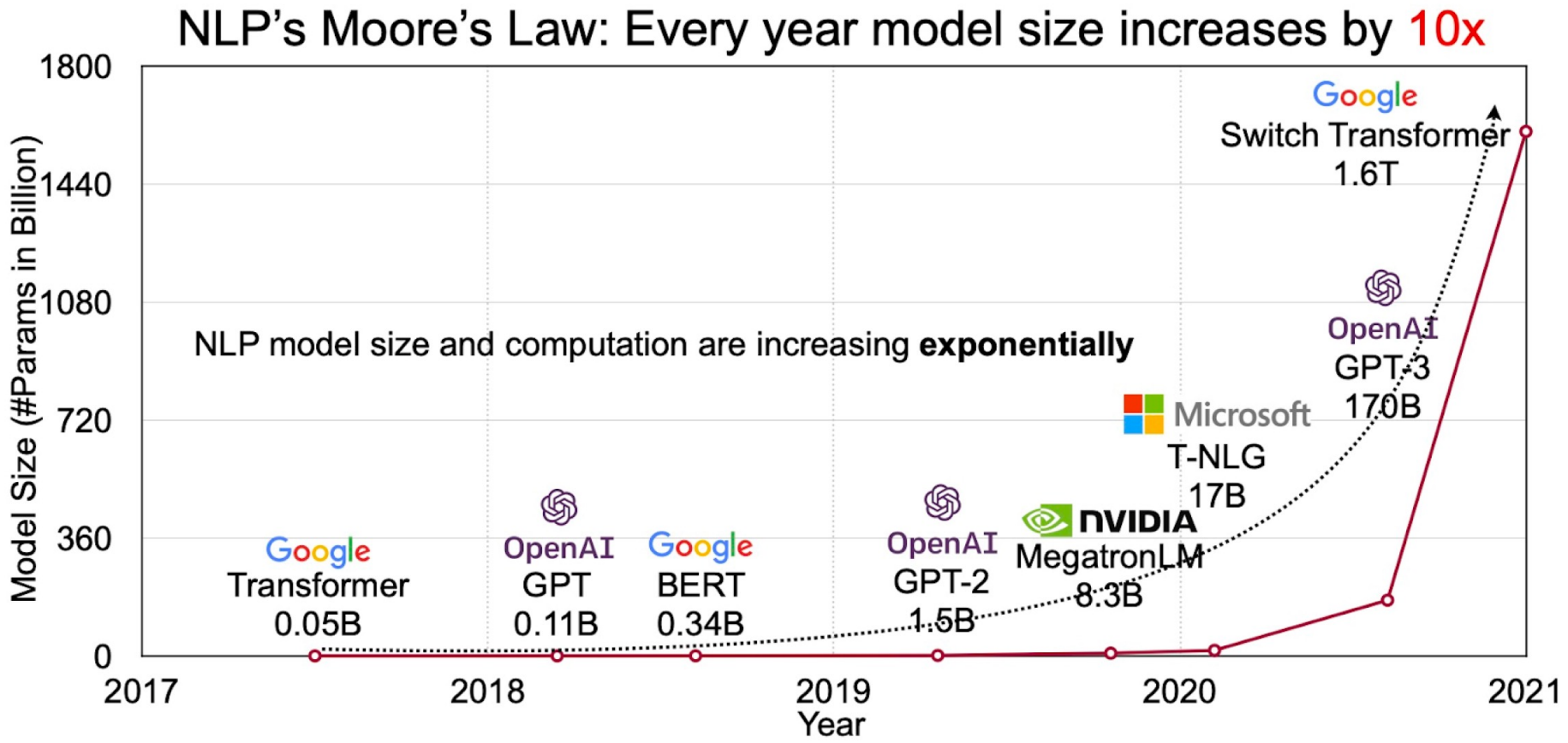
Pipeline



PipelineModel

Distributed Machine Learning

NLP Model Size Growth



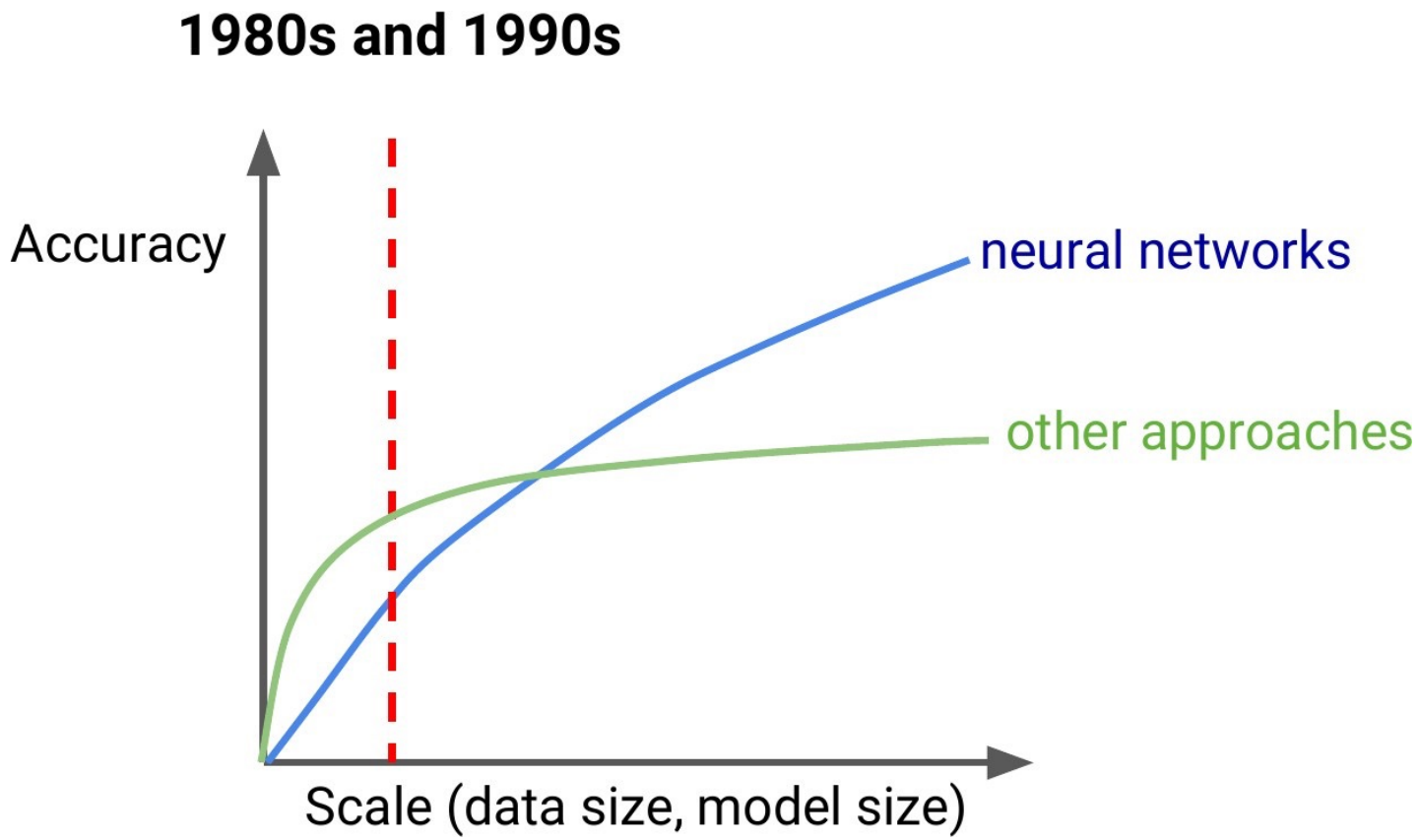
https://indiaai.gov.in/article/the-future-of-large-language-models-llms-strategy-opportunities-and-challenges

Challenge

- Training deep neural networks are:
  - Computationally intensive
  - Time consuming

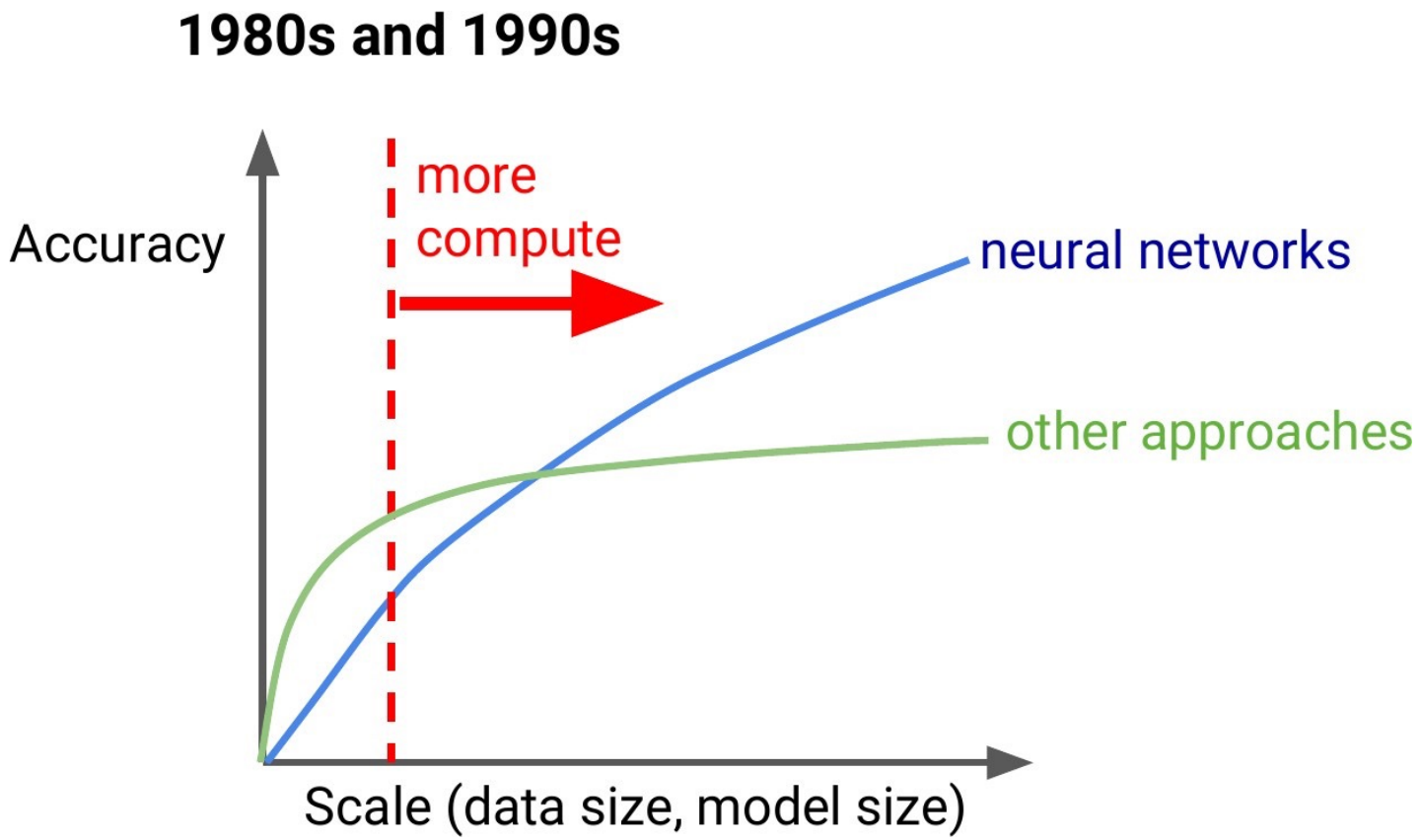
- Why ?
  - Massive amount of training dataset
  - Large number of parameters

Accuracy Vs. Data/Model Size



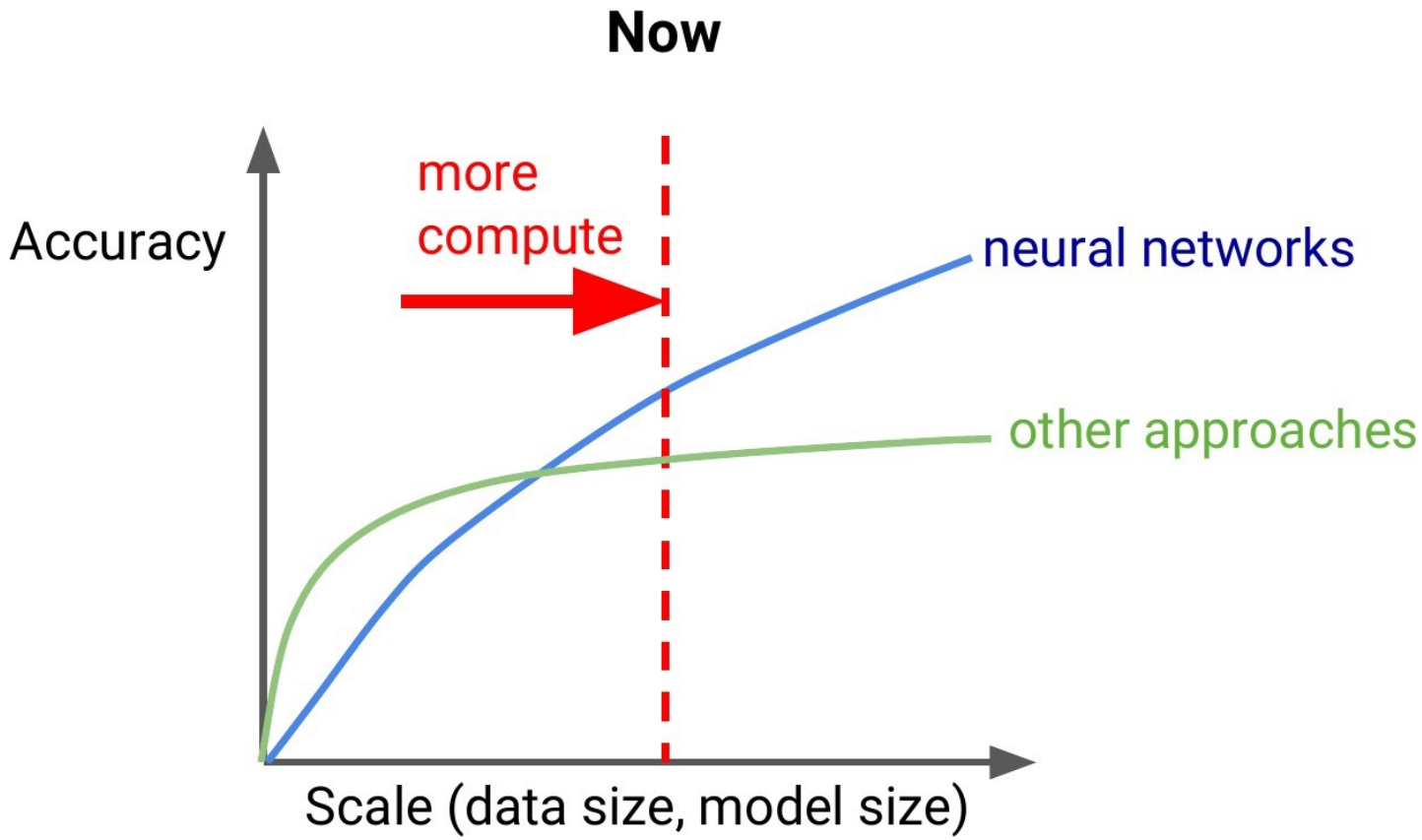
[Jeff Dean at AI Frontiers: Trends and Developments in Deep Learning Research]

Accuracy Vs. Data/Model Size



[Jeff Dean at AI Frontiers: Trends and Developments in Deep Learning Research]

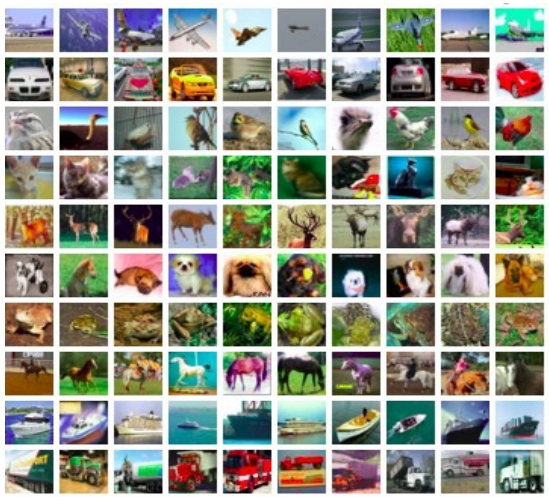
Accuracy Vs. Data/Model Size



[Jeff Dean at AI Frontiers: Trends and Developments in Deep Learning Research]

Fundamentals of Machine Learning

Training Data Sets



Entities

Society and Culture | Science and Mathematics | Health | Education and Reference | Computers and Internet | Sports | Business and Finance | Entertainment and Music | Family and Relationships | Politics and Government

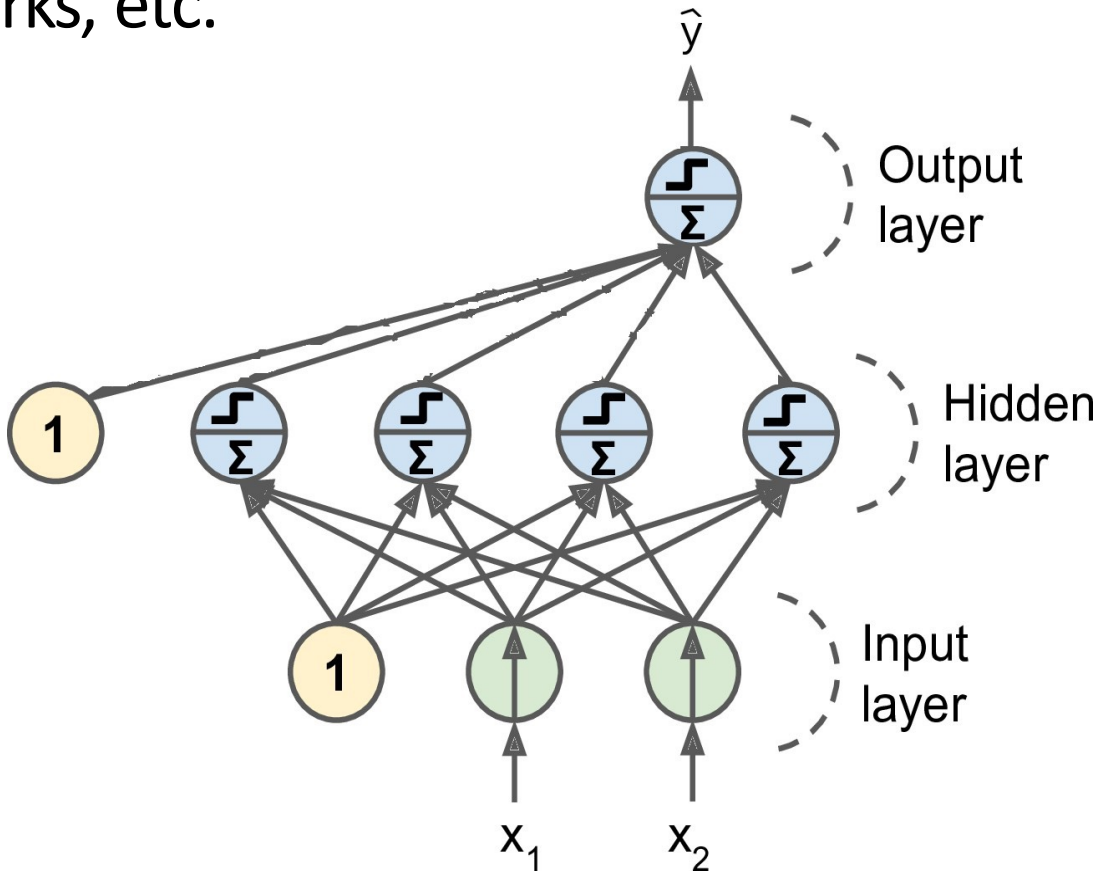
does anyone here play habbohotel and want 2 be friends? Answer: No on the first part and maybe on the second part. I got to think it over first.

Family and Relationships

Date	Cost	Actions	Offsite conversions	Impressions	Clicks
2017-04-04	29.44	461	4	5655	477
2017-04-03	74.08	1331	16	18170	1340
2017-04-02	76.09	1349	12	16877	1357
2017-04-01	76.79	1382	8	19757	1378
2017-03-31	77.28	1141	21	18598	1116
2017-03-30	68.62	1065	18	14847	1046
2017-03-29	64.9	1111	25	13994	1094
2017-03-28	65.12	1137	12	15952	1145
2017-03-27	66.98	1185	7	17970	1190
2017-03-26	64.94	1118	5	14410	1116
2017-03-25	66.3	1208	6	15123	1204
2017-03-24	67.38	1143		15298	1159
2017-03-23	65.59	1147	13	14972	1143
2017-03-22	68.19	1129	4	17959	1116
2017-03-21	64.78	1081		25810	1059

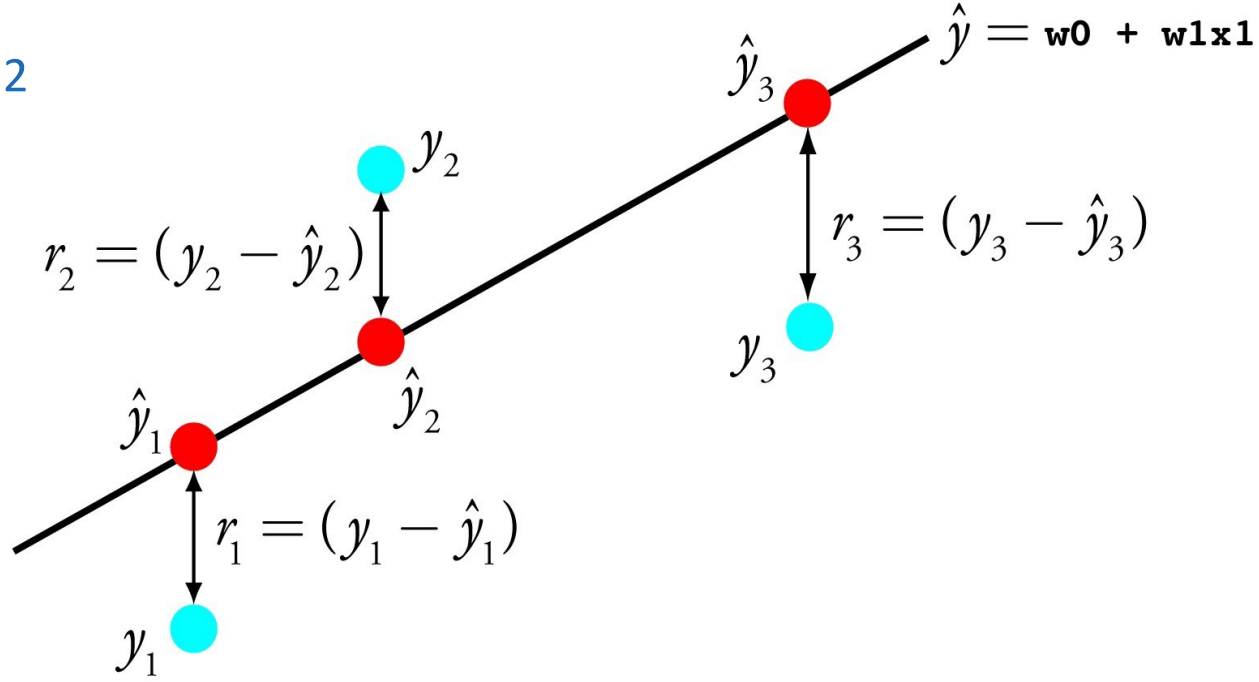
ML Model

- E.g., linear models, neural networks, etc.
- $\hat{y} = f_w(x)$



Loss Function

- How good $\hat{y}$ is able to predict the expected outcome y.
- $J(w) = \sum_{i=1}^{m} l(y_i, \hat{y}_i)$
- $J(w) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$

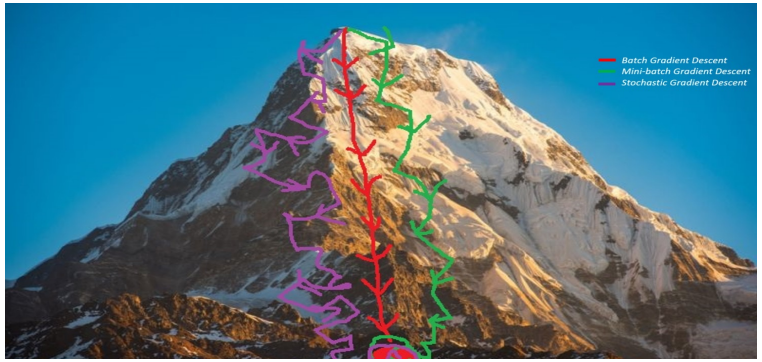


Objective

- Minimize the loss function
- $\arg\min_w J(w)$
- $J(w) = \sum_{i=1}^{m} l(y_i, \hat{y}_i)$

Training

- $J(w) = \sum_{i=1}^{m} l(y_i, \hat{y}_i)$
- Gradient descent, i.e., $w := w - \eta \nabla J(w)$
- Stochastic gradient descent, i.e., $w := w - \eta \tilde{g} J(w)$
  - $\tilde{g}$: gradient at a randomly chosen point.
- Mini-barch gradient descent, i.e., $w := w - \eta \tilde{g}_B J(w)$
  - $\tilde{g}$: gradient with respect to a set of B randomly chosen points.



Let’s Scale/Distribute Training

Distributed Training

- Using the right hardware configuration can dramatically reduce training time.
- A shorter training time makes for faster iteration to reach your modeling goals.

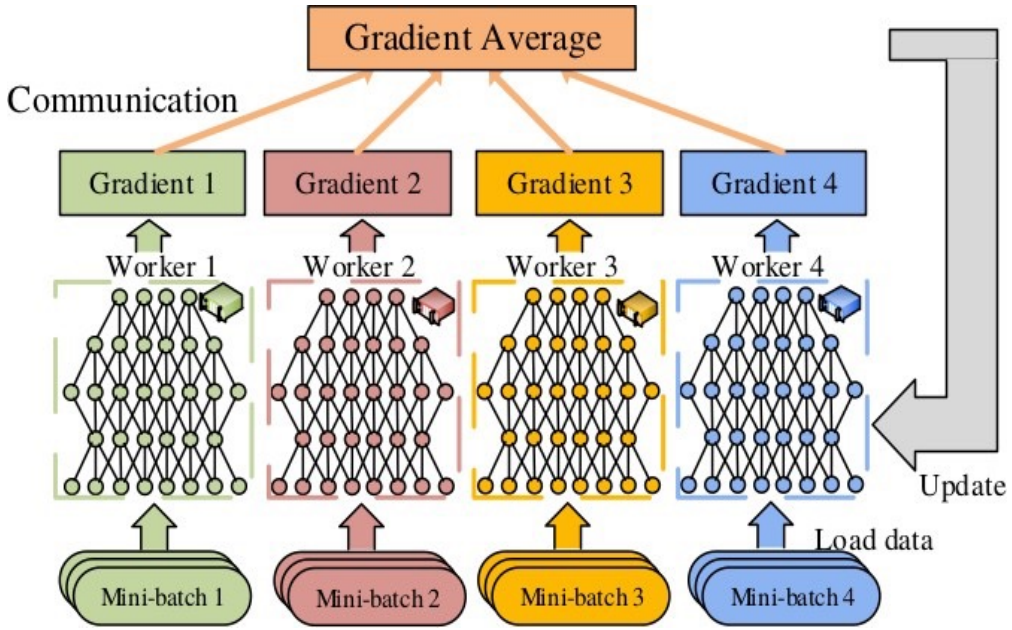
Distributed training

Data Parallelism

Model Parallelism

Data Parallelism

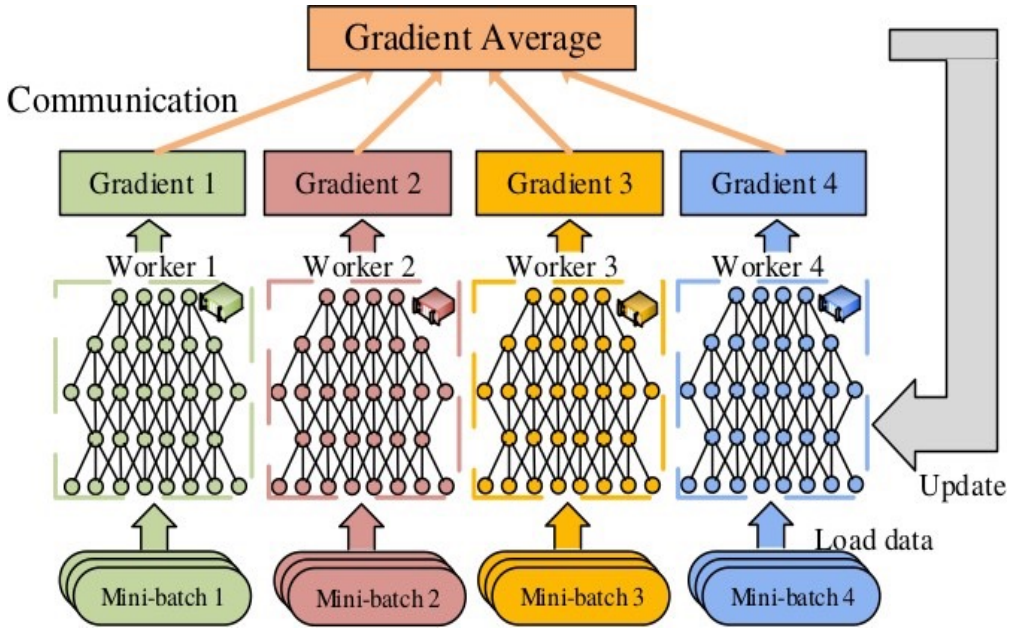
- Replicate a whole program on every device.
- Train all replicas simultaneously, using a different mini-batch for each.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Data Parallelism

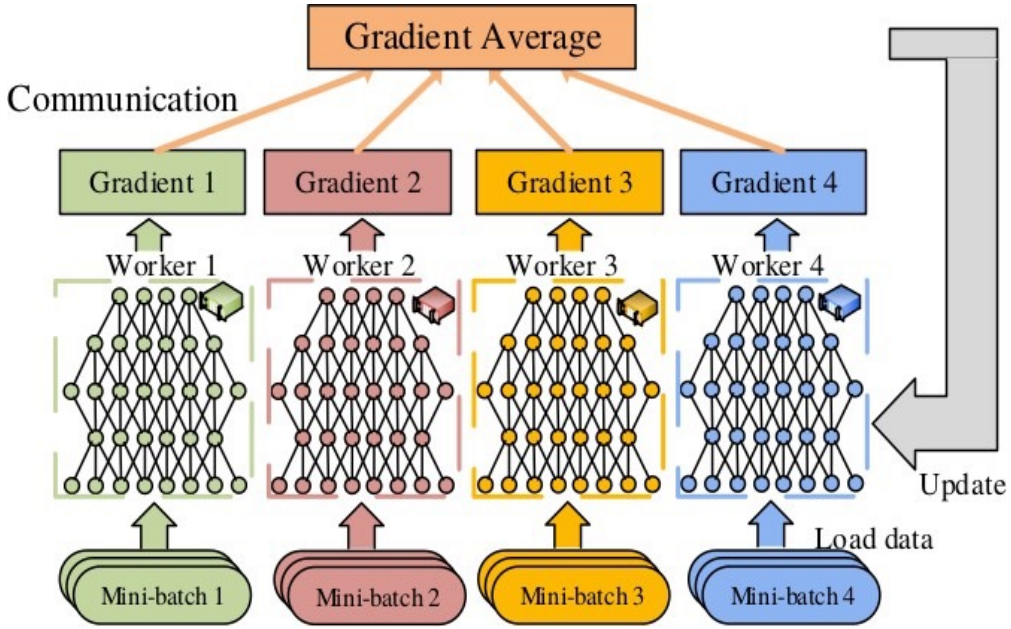
- Split data to K mini-batches on K devices
- $J_j(w) = \sum_{i=1}^{b_j} l(y_i, \hat{y}_i), \forall j = 1, 2, \cdots, k$
- $\tilde{g}_B J_j(w)$: gradient of $J_j(w)$ with respect to a set of B randomly chosen points at device j.
- Compute $\tilde{g}_B J_j(w)$ on each device j.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Data Parallelism

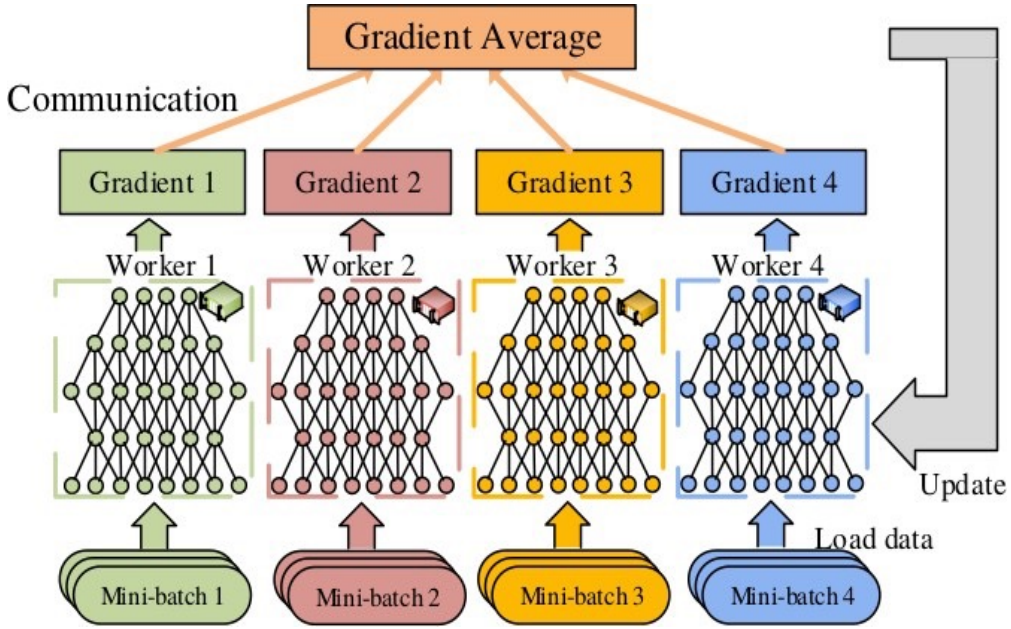
- Compute the mean of gradients
  - $g_B J(W) = \frac{1}{k} \sum_{j=1}^{k} \sim g_B J_j(W)$



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Data Parallelism

- Update the model
  - $w := w - \eta g_B J(w)$



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

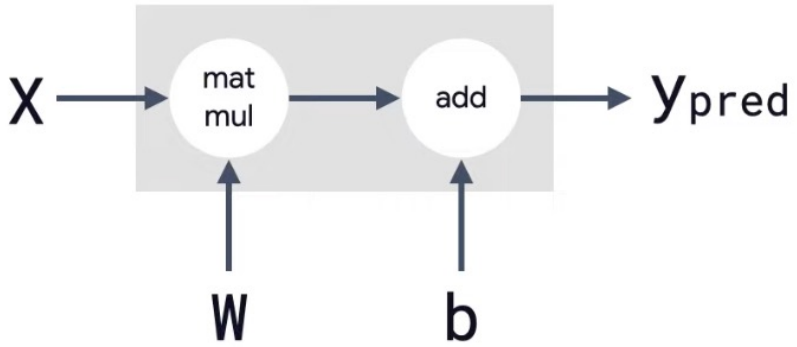
Batch size

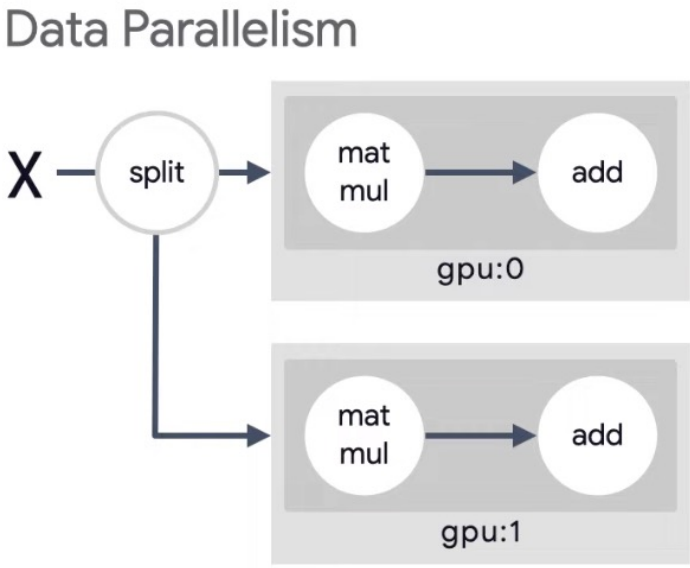
```
model.fit(x, y, batch_size=32)
```

```
model.fit(x, y, batch_size=(32 * NUM_GPUS))
```

Linear Model with Data Parallelism

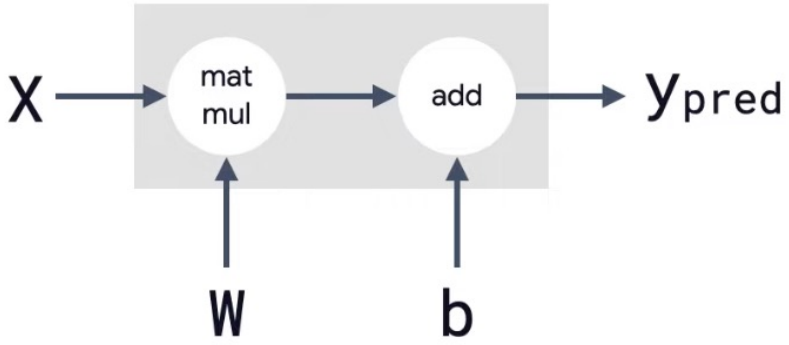
$$y_{pred} = WX + b$$



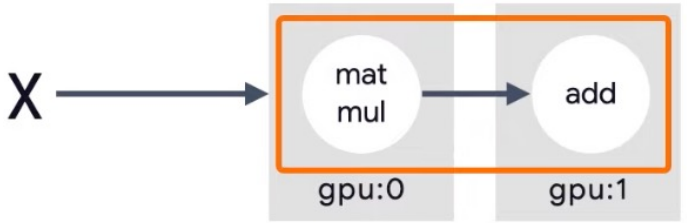


Linear Model with Model Parallelism

$$y_{pred} = WX + b$$

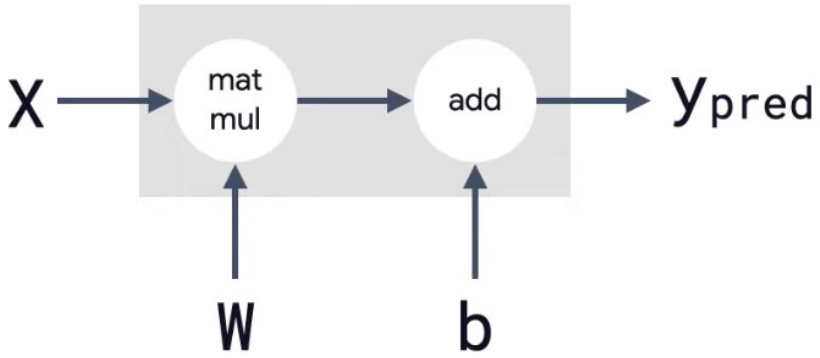


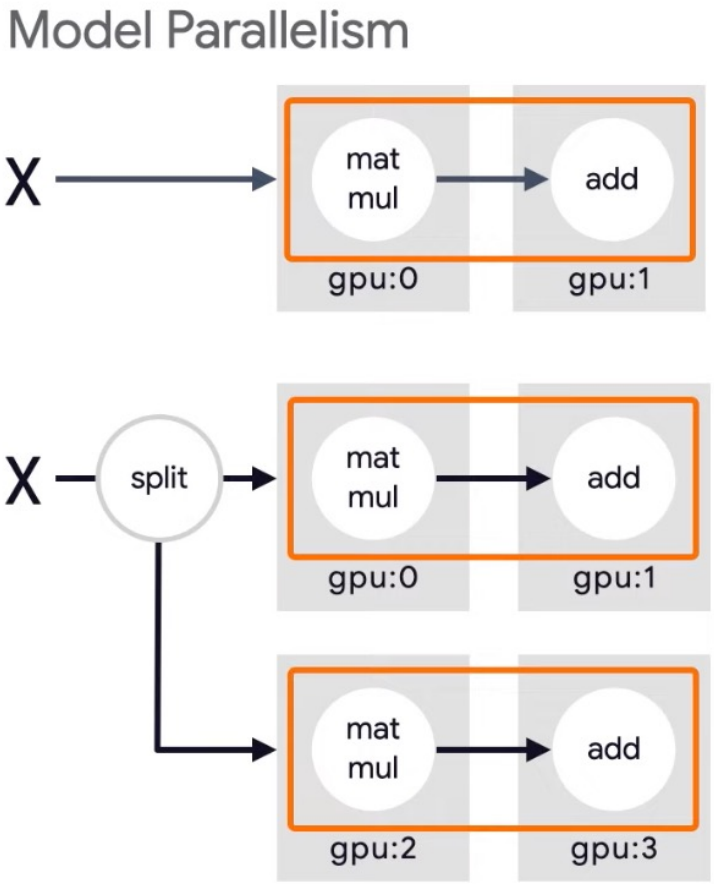
Model Parallelism



Linear Model with Data/Model Parallelism

$$y_{pred} = WX + b$$





Data Parallelism Design Issues

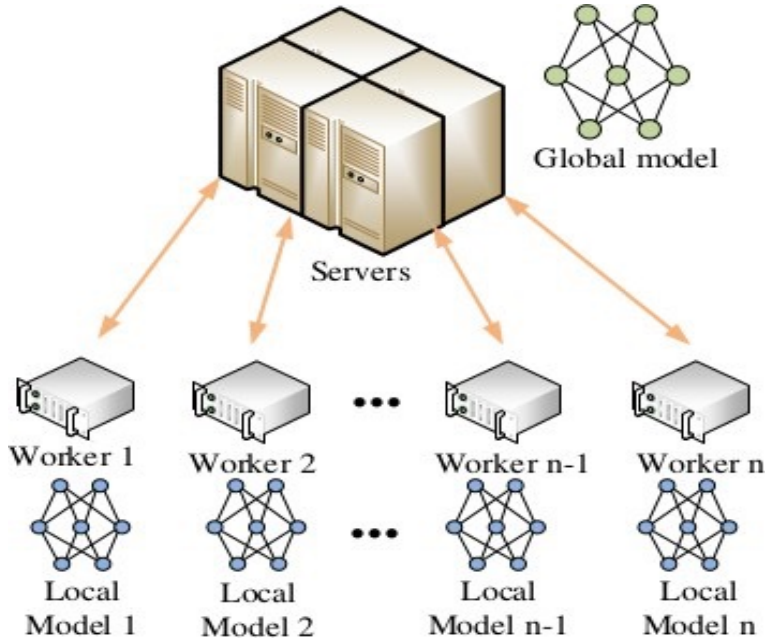
- The aggregation algorithm
- Communication synchronization and frequency
- Communication compression

The Aggregation Algorithm

- How to aggregate gradients (compute the mean of the gradients)?
- Centralized - Parameter Server
- Decentralized - AllReduce
- Decentralized - Gossip

Aggregation: Centralized - Parameter Server

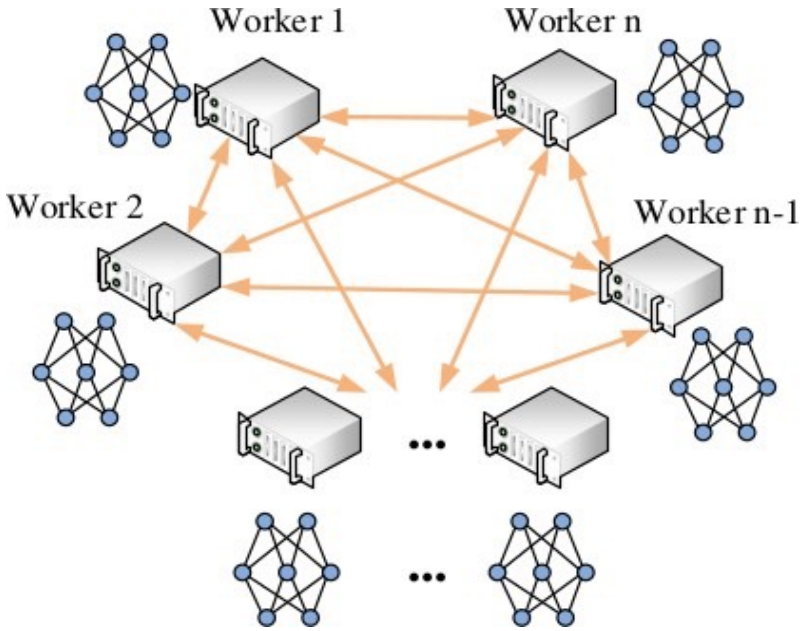
- Store the model parameters outside of the workers.
- Workers periodically report their computed parameters or parameter updates to a (set of) parameter server(s) (PSs).



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Aggregation: Decentralized - AllReduce

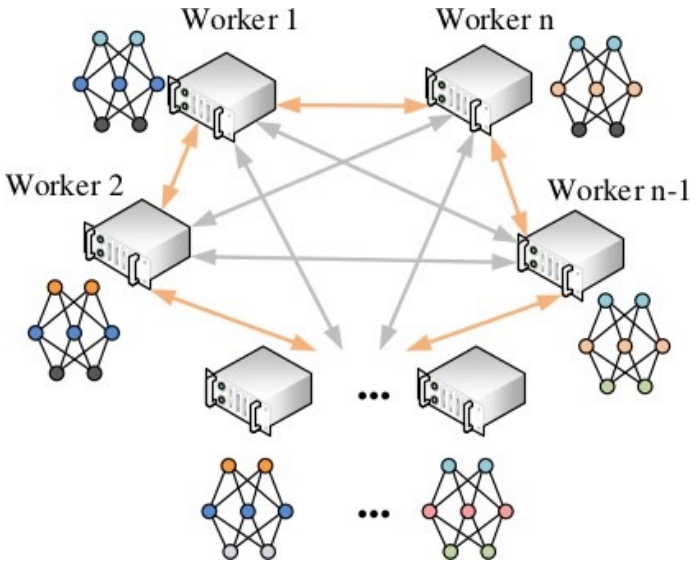
- Mirror all the model parameters across all workers (no PS).
- Workers exchange parameter updates directly via an allreduce operation.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Aggregation: Distributed - Gossip

- No PS, and no global model.
- Every worker communicates and updates with their neighbors.
- The consistency of parameters across all workers only at the end of the algorithm.

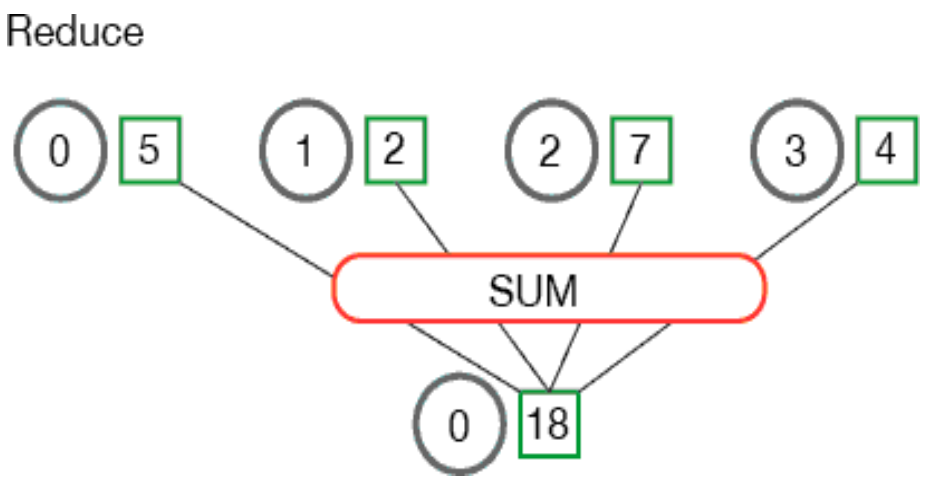


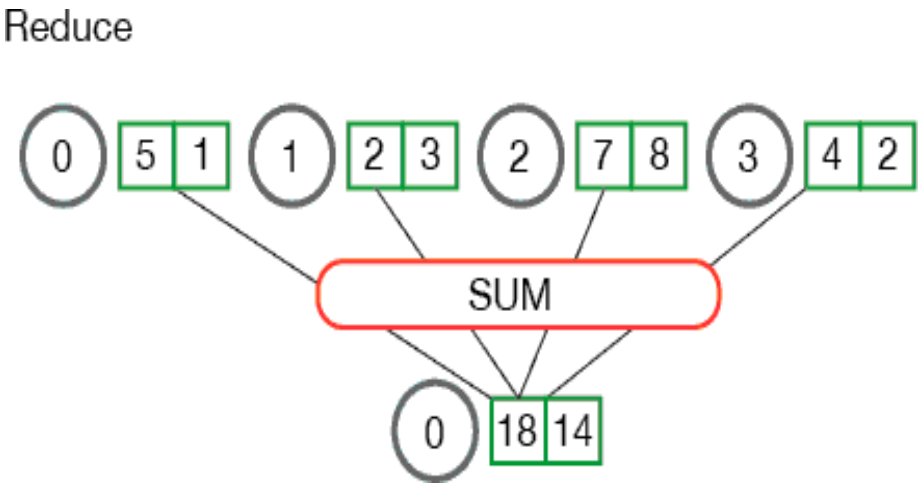
[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Centralized Vs. Decentralized

Reduce

- Reduce: reducing a set of numbers into a smaller set of numbers via a function.
- E.g., `sum([1, 2, 3, 4, 5]) = 15`
- Reduce takes an array of input elements on each process and returns an array of output elements to the root process.

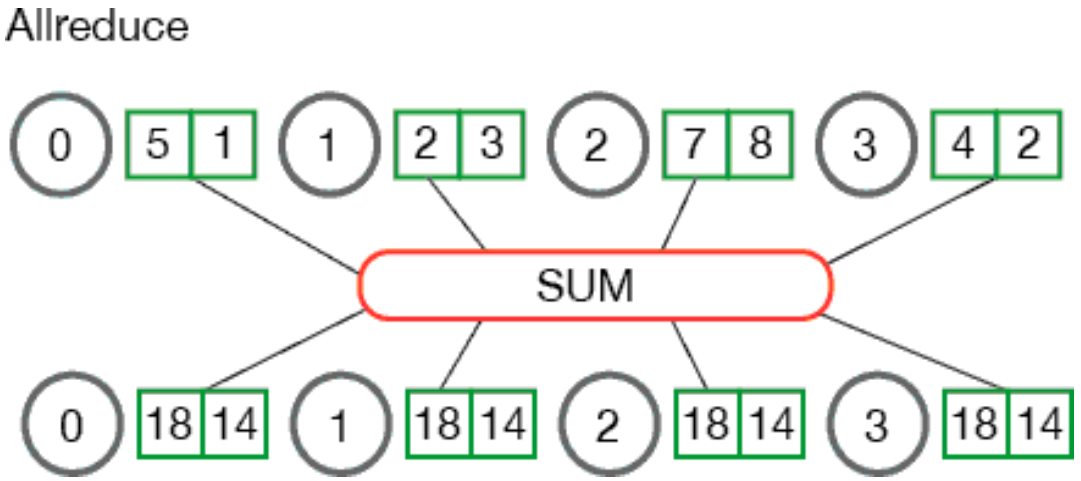




[https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce]

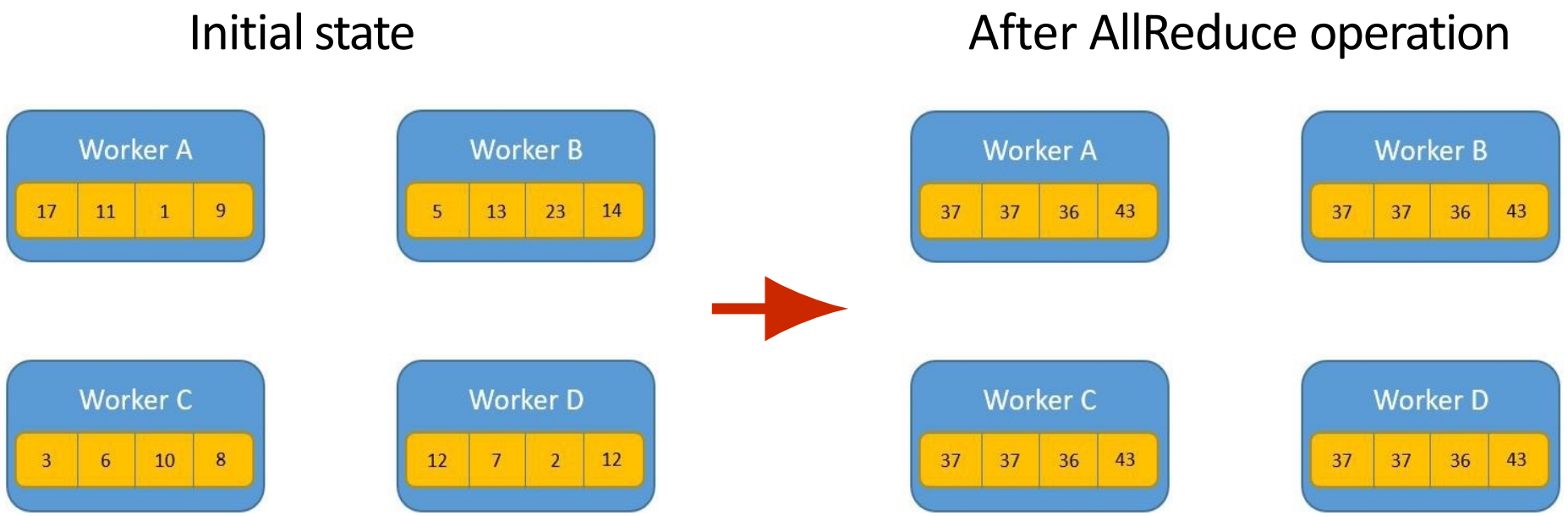
AllReduce

- AllReduce stores reduced results across all processes rather than the root process.



[https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce]

AllReduce Example



[https://towardsdatascience.com/visual-intuition-on-ring-allreduce-for-distributed-deep-learning-d1f34b4911da]

Synchronization

- When to synchronize the parameters among the parallel workers?

Communication Synchronization

- Synchronizing the model replicas in data-parallel training requires communication
  - between workers, in AllReduce
  - between workers and parameter servers, in the centralized architecture
- The communication synchronization decides how frequently all local models are synchronized with others.
- It will influence:
  - The communication traffic
  - The performance
  - The convergence of model training
- There is a trade-off between the communication traffic and the convergence.

Reducing Synchronization Overhead

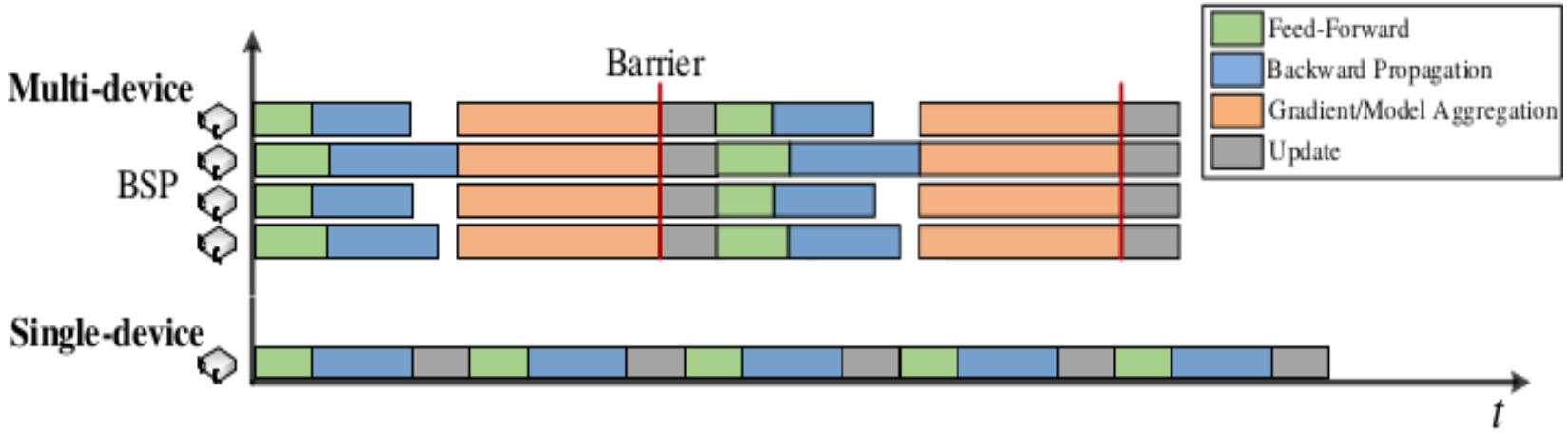
- Two directions for improvement:
  - To relax the synchronization among all workers.
  - The frequency of communication can be reduced by more computation in one iteration.

Communication Synchronization Models

- Synchronous
- Stale-synchronous
- Asynchronous
- Local SGD

Synchronous Gradient Update

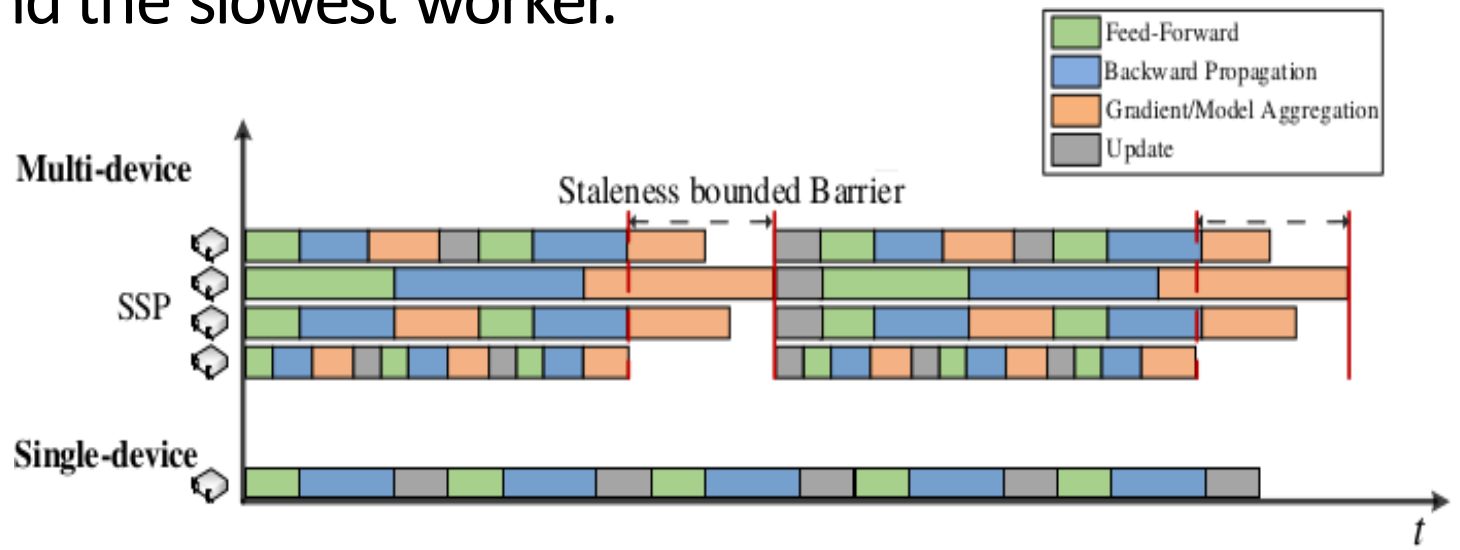
- After each iteration, the workers synchronize their parameter updates.
- Every worker must wait for all workers to finish the transmission of all parameters in the current iteration, before the next training.
- Stragglers can influence the overall system throughput.
- High communication cost that limits the system scalability.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Stale-Synchronous Update

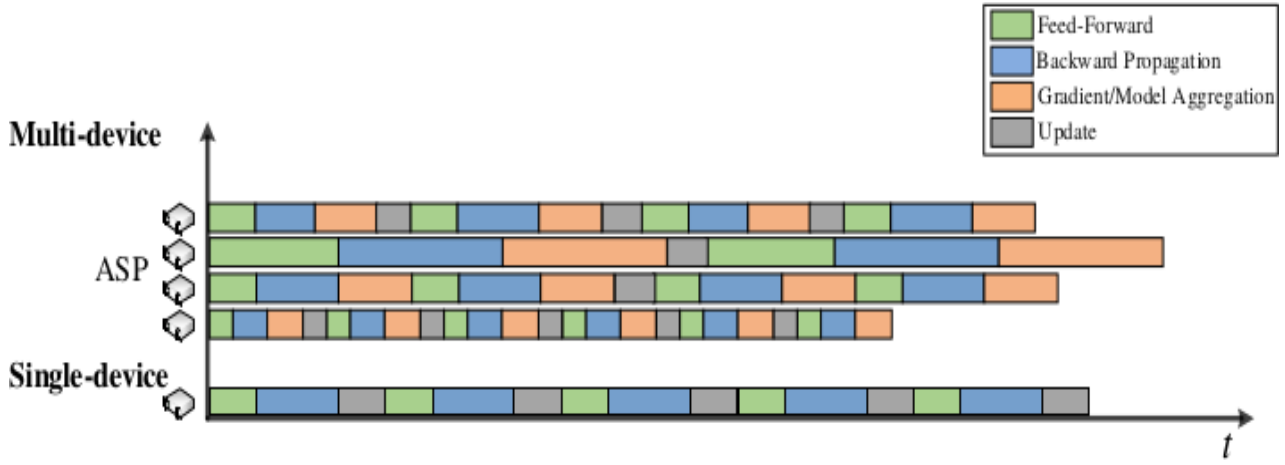
- Alleviate the straggler problem without losing synchronization.
- The faster workers to do more updates than the slower workers to reduce the waiting time of the faster workers.
- Staleness bounded barrier to limit the iteration gap between the fastest worker and the slowest worker.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Asynchronous Update

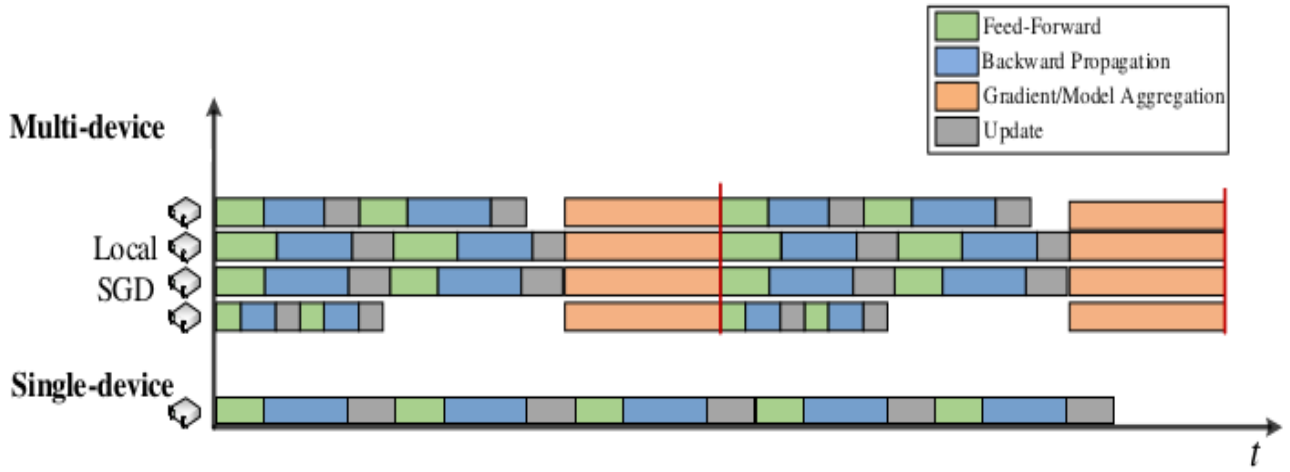
- It completely eliminates the synchronization.
- Each work transmits its gradients to the PS after it calculates the gradients.
- The PS updates the global model without waiting for the other workers.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Local SGD

- All workers run several iterations, and then averages all local models into the newest global model.



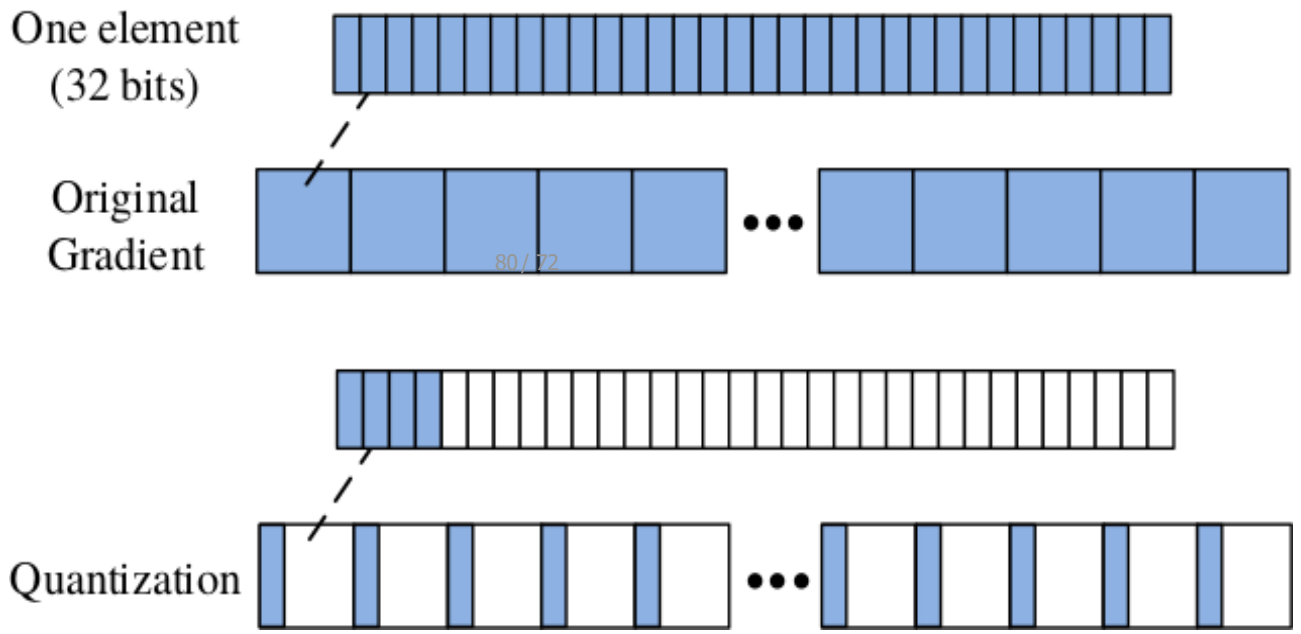
[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Communication Compression

- Reduce the communication traffic with little impact on the model convergence.
- Compress the exchanged gradients or models before transmitting across the network.
  - Quantization
  - Sparsification

Communication Compression - Quantization

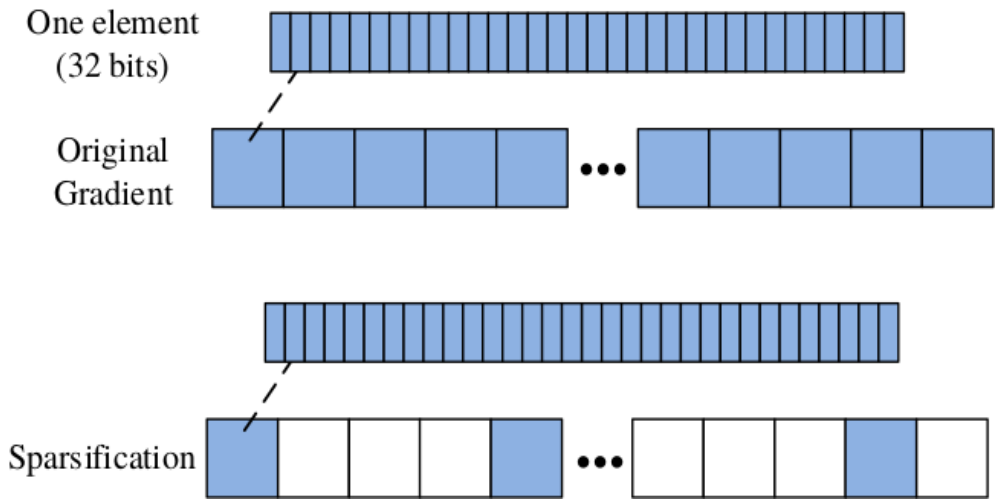
- Using lower bits to represent the data.
- The gradients are of low precision.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Communication Compression - Sparcification

- Reducing the number of elements that are transmitted at each iteration.
- Only significant gradients are required to update the model parameter to guarantee the convergence of the training.
- E.g., the zero-valued elements are no need to transmit.



[Tang et al., Communication-Efficient Distributed Deep Learning: A Comprehensive Survey, 2020]

Recap

- Scalability matters
- Parallelization
- Data Parallelization
  - Parameter server vs. AllReduce
  - Synchronized vs. asynchronized

Next class:

Stream Processing