Distributed TensorFlow

CSE545 - Spring 2022

Stony Brook University

H. Andrew Schwartz

Big Data Analytics, The Class

Goal: Generalizations
A *model* or *summarization* of the data.

Data Frameworks

- Hadoop File System
- Spark
- Streaming
- MapReduce
- Tensorflow

Algorithms and Analyses

- Similarity Search
- Hypothesis Testing
- Graph Analysis
- Recommendation Systems
- Deep Learning

Limitations of Spark

Spark is fast for being so flexible

- Fast: RDDs in memory + Lazy evaluation: optimized chain of operations.
- Flexible: Many transformations -- can contain any custom code.

Limitations of Spark

Spark is fast for being so flexible

- Fast: RDDs in memory + Lazy evaluation: optimized chain of operations.
- Flexible: Many transformations -- can contain any custom code.

However:

- Hadoop MapReduce can still be better for extreme IO, data that will not fit in memory across cluster.

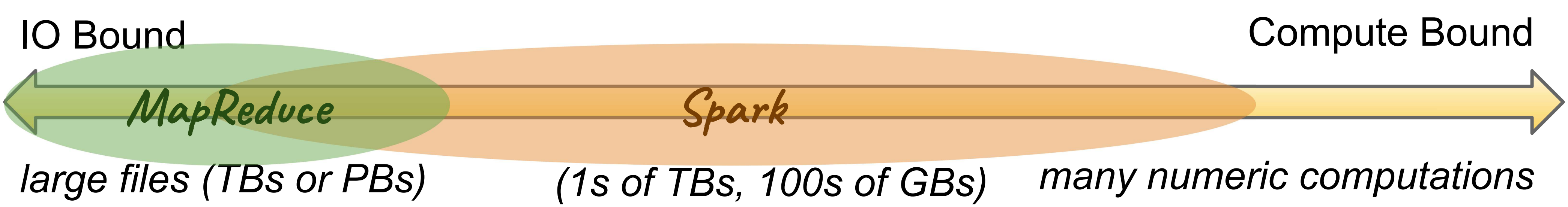
Limitations of Spark

Spark is fast for being so flexible

- Fast: RDDs in memory + Lazy evaluation: optimized chain of operations.
- Flexible: Many transformations -- can contain any custom code.

However:

- Hadoop MapReduce can still be better for extreme IO, data that will not fit in memory across cluster.



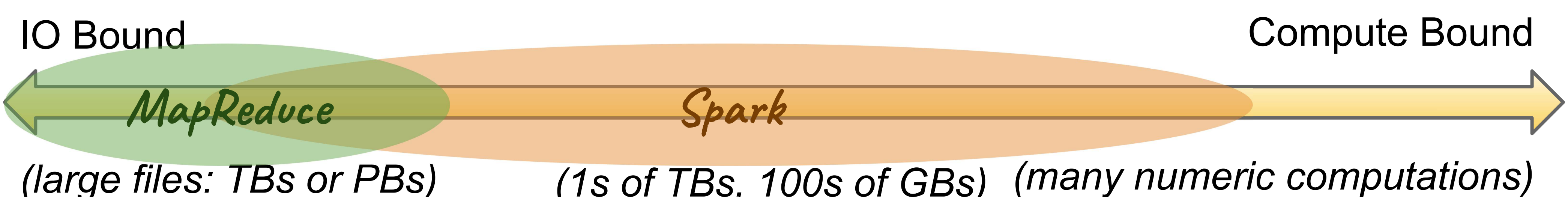
Limitations of Spark

Spark is fast for being so flexible

- Fast: RDDs in memory + Lazy evaluation: optimized chain of operations.
- Flexible: Many transformations -- can contain any custom code.

However:

- Hadoop MapReduce can still be better for extreme IO, data that will not fit in memory across cluster.
- Modern machine learning (esp. Deep learning), a common big data task, requires heavy numeric computation.



* this is the subjective approximation of the instructor as of February 2020. A lot of factors at play.

Limitations of Spark

Spark is fast for being so flexible

- Fast: RDDs in memory + Lazy evaluation: optimized chain of operations.
- Flexible: Many transformations -- can contain any custom code.

However:

- Hadoop MapReduce can still be better for extreme IO, data that will not fit in memory across cluster.
- Modern machine learning (esp. Deep learning), a common big data task, requires heavy numeric computation.

* this is the subjective approximation of the instructor as of February 2020. A lot of factors at play.

Learning Objectives

- Understand TensorFlow as a data workflow system.
    - Know the key components of TensorFlow.
    - Understand the key concepts of *distributed* TensorFlow.
- Execute basic distributed tensorflow program.
- Establish a foundation to distribute deep learning models:
    - Convolutional Neural Networks
    - Recurrent Neural Network (or LSTM, GRU)

What is TensorFlow?

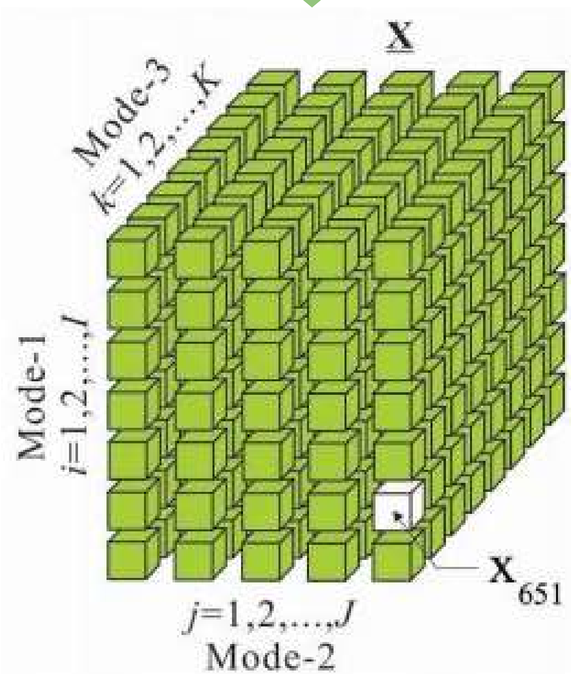
A workflow system catered to numerical computation.

One view: Like Spark, but uses *tensors* instead of *RDDs*.

What is TensorFlow?

A workflow system catered to numerical computation.

One view: Like Spark, but uses *tensors* instead of *RDDs*.



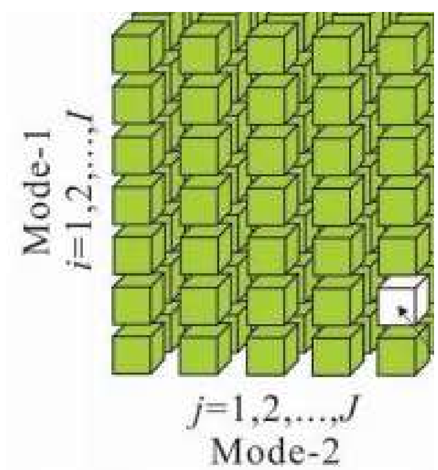
A multi-dimensional matrix

(i.stack.imgur.com)

What is **Tensor**Flow?

A workflow system catered to numerical computation.

One view: Like Spark, but uses *tensors* instead of *RDDs*.



(i.stack.imgur.com)

A 2-d tensor is just a matrix.

- 1-d: vector
- 0-d: a constant / scalar

Note: Linguistic ambiguity:
Dimensions of a Tensor =/=
Dimensions of a Matrix

What is **Tensor**Flow?

A workflow system catered to numerical computation.

One view: Like Spark, but uses *tensors* instead of *RDDs*.

Examples > 2-d :

Image definitions in terms of RGB per pixel

Image[*row*][*column*][*rgb*]

Subject, Verb, Object representation of language:

Counts[verb][subject][object]

What is **Tensor**Flow?

A workflow system catered to numerical computation.

One view: Like Spark, but uses *tensors* instead of *RDDs*.

Technically, less abstract than *RDDs* which could hold tensors as well as many other data structures (dictionaries/HashMaps, Trees, ...etc…).

Then, why TensorFlow?

What is TensorFlow?

Efficient, high-level built-in **linear algebra** and **machine learning optimization** *operations* (i.e. transformations).

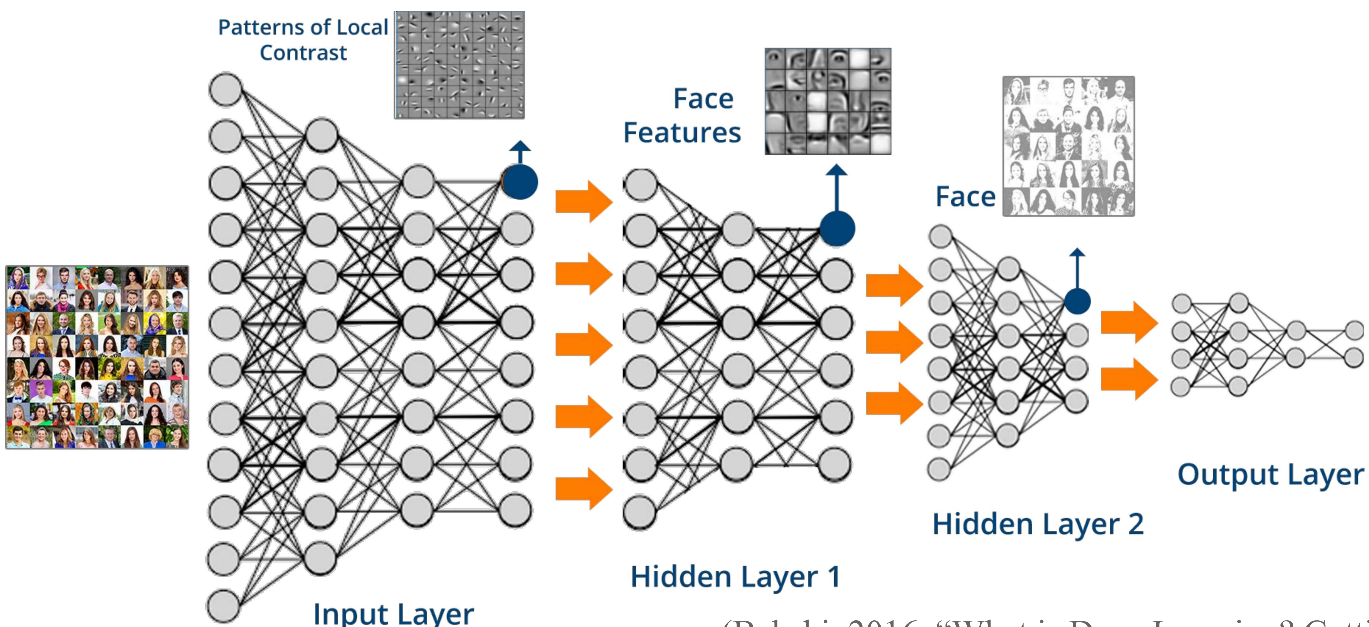
enables complex models, like deep learning

Then, why TensorFlow?

What is TensorFlow?

Efficient, high-level built-in **linear algebra** and **machine learning optimization** *operations*.

enables complex models, like deep learning



(Bakshi, 2016, “What is Deep Learning? Getting Started With Deep Learning”)

What is TensorFlow?

Efficient, high-level built-in **linear algebra** and **machine learning** *operations*.

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                   # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                       # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)             # Relu(Wx+b)
C = [...]                                          # Cost computed as a function
                                                   # of Relu

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...       # Create 100-d vector for input
  result = s.run(C, feed_dict={x: input})          # Fetch cost, feeding x=input
  print step, result
```

(Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.)

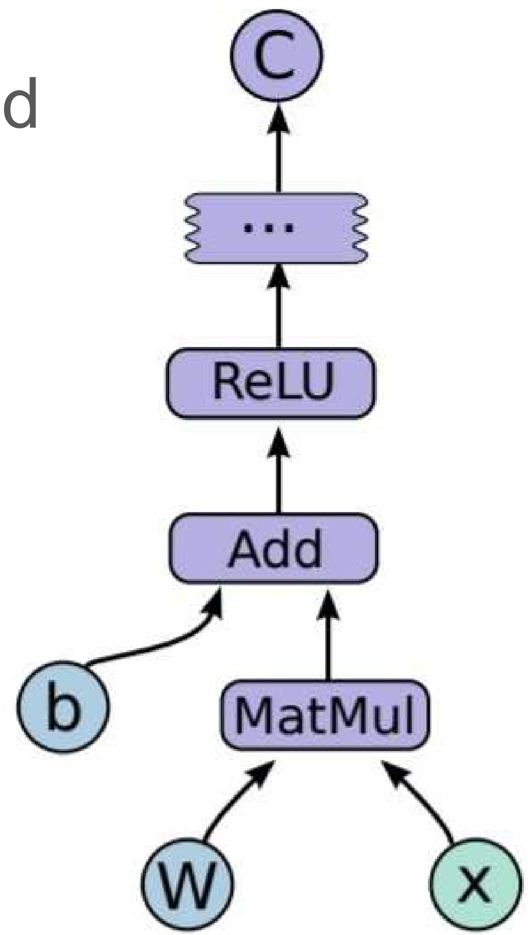
TensorFlow

Operations on tensors are often conceptualized as **graphs**:

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
C = [...]

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...
  result = s.run(C, feed_dict={x: input})
  print step, result
```



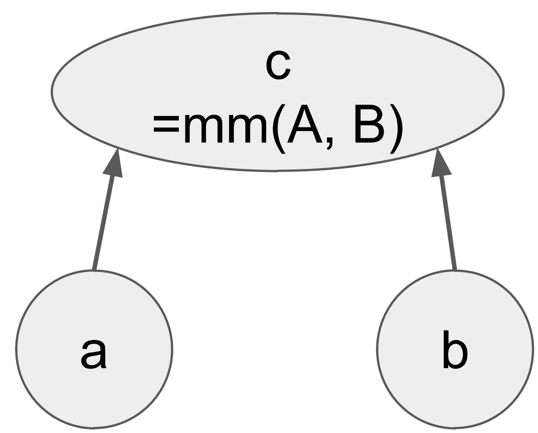
(Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.)

Tensor**Flow**

Operations on tensors are often conceptualized as graphs:

A simpler example:

c = tensorflow.matmul(a, b)

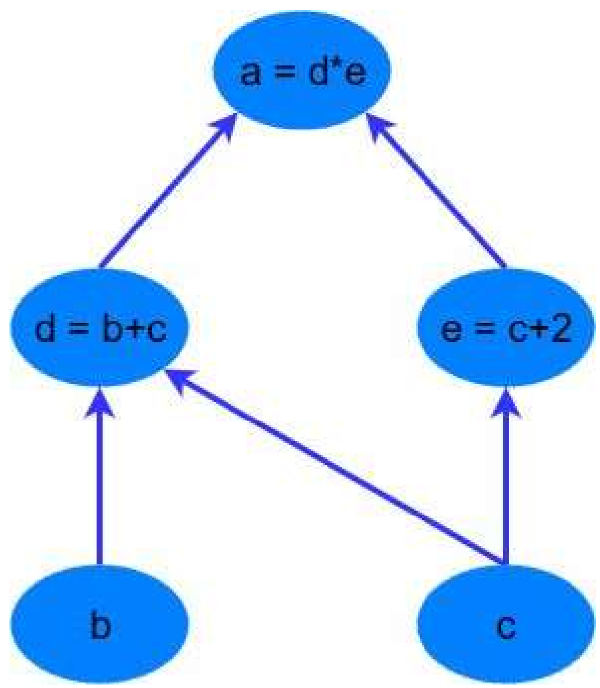


Tensor**Flow**

Operations on tensors are often conceptualized as graphs:

example:

d=b+c
e=c+2
a=d∗e



(Adventures in Machine Learning. *Python TensorFlow Tutorial*, 2017)

Ingredients of a TensorFlow

tensors*
variables - persistent mutable tensors
constants - constant
placeholders - from data

* technically, still *operations*

operations
an abstract computation (e.g. matrix multiply, add) executed by device *kernels*

graph

session
defines the environment in which operations *run*. (like a Spark context)

devices
the specific devices (cpus or gpus) on which to run the session.

Ingredients of a TensorFlow

tensors*
variables - persistent mutable tensors
constants - constant
placeholders - from data

* technically, still *operations*

operations
an abstract computation
(e.g. matrix multiply, add)
executed by device *kernels*

- tf.Variable(initial_value, name)
- tf.constant(value, type, name)
- tf.placeholder(type, shape, name)

graph

session
defines the environment in which operations *run*.
(like a Spark context)

devices
the specific devices (cpus or gpus) on which to run the session.

Ingredients of a TensorFlow

tensors*
variables - persistent
mutable tensors
constants - constant
placeholders - from data

operations
an abstract computation
(e.g. matrix multiply, add)
executed by device *kernels*

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural-net building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

Ingredients of a TensorFlow

- Places operations on devices
- Stores the values of variables (when not distributed)
- Carries out execution: eval() or run()

tensors*
variables - persistent mutable tensors
constants - constant
placeholders - from data

operations
an abstract computation (e.g. matrix multiply, add)
executed by device *kernels*

graph

session
defines the environment in which operations *run*.
(like a Spark context)

devices
the specific devices (cpus or gpus) on which to run the session.

Ingredients of a TensorFlow

tensors*
variables - persistent mutable tensors
constants - constant
placeholders - from data

operations
an abstract computation (e.g. matrix multiply, add) executed by device *kernels*

graph

session
defines the environment in which operations *run*.
(like a Spark context)

devices
the specific devices (cpus or gpus) on which to run the session.

Distributed TensorFlow

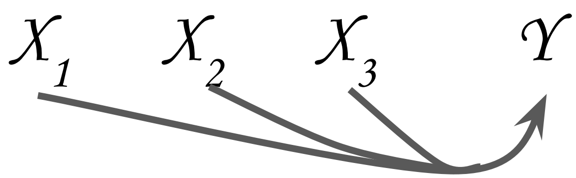
Typical use-case: (Supervised Machine Learning)

Determine weights, $\mathcal{W}$, of a function, f, such that $\boldsymbol{\varepsilon}$ is minimized: $f(\mathcal{X} \mid \mathcal{W}) = \mathcal{Y} + \boldsymbol{\varepsilon}$

Distributed TensorFlow

Typical use-case:

Determine weights, $\mathcal{W}$, of a function, f, such that $\boldsymbol{\varepsilon}$ is minimized: $f(\mathcal{X} \mid \mathcal{W}) = \mathcal{Y} + \boldsymbol{\varepsilon}$



Distributed TensorFlow

Typical use-case:

Determine weights, $\mathcal{W}$, of a function, f, such that $\boldsymbol{\varepsilon}$ is minimized: $f(\mathcal{X} \mid \mathcal{W}) = \mathcal{Y} + \boldsymbol{\varepsilon}$

$X_1 \quad X_2 \quad X_3 \qquad Y$

$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \quad X_6$

$X_7 \quad X_8 \quad X_9 \quad X_{10} \quad X_{11} \quad X_{12} \qquad\qquad Y$

$X_{13} \quad X_{14} \quad X_{15} \quad \ldots \quad X_m$

Distributed TensorFlow

Typical use-case:

Determine weights, $\mathcal{W}$, of a function, f, such that **ε** is minimized: $f(X \mid \mathcal{W}) = Y + \varepsilon$

$X_1 \quad X_2 \quad X_3 \quad Y$

$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \quad X_6$

$X_7 \quad X_8 \quad X_9$

$X_{13} \quad X_{14} \quad X_{15} \quad \ldots \quad X_m$

$\xrightarrow{f \text{ given } w_1, w_2, \ldots, w_p} Y$

(typically, $p >= m$)

Distributed TensorFlow

$$f(X|W) = \hat{Y}$$
$$Y = (X|W) + \varepsilon$$
$$Y = \hat{Y} + \varepsilon$$
$$\varepsilon = \hat{Y} - Y$$

Typical use-case:

Determine weights, $\mathcal{W}$, of a function, f, such that $|\varepsilon|$ is minimized:

$X_1 \quad X_2 \quad X_3 \quad \rightarrow \quad Y$

$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \quad X_6$

$X_7 \quad X_8 \quad X_9$ $\xrightarrow{f \text{ given } w_1, w_2, \ldots, w_p}$ Y

$X_{13} \quad X_{14} \quad X_{15} \quad \ldots \quad X_m$

(typically, $p >= m$)

Distributed TensorFlow

Typical use-case:

Determine weights, $\mathcal{W}$, of a function, f, such that ε is minimized: f

$$f(X|W) = \hat{Y}$$
$$Y = (X|W) + \varepsilon$$
$$Y = \hat{Y} + \varepsilon$$
$$\varepsilon = \hat{Y} - Y$$

$X_1 \quad X_2 \quad X_3 \quad \Upsilon$

Typically, very complex!

$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \quad X_6$

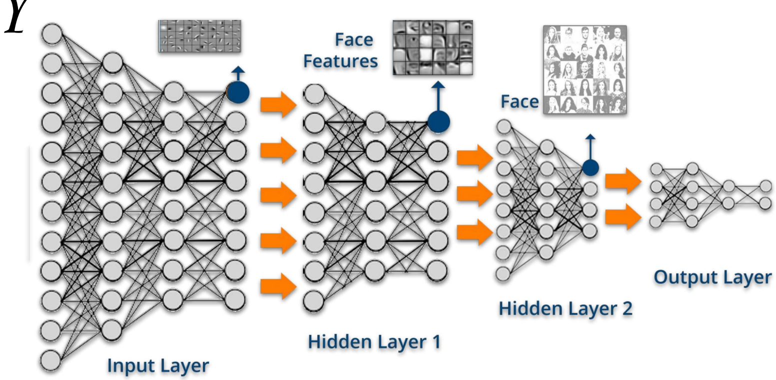
$X_7 \quad X_8 \quad X_9$

$X_{13} \quad X_{14} \quad X_{15} \quad \ldots \quad X_m$

f given $w_1, w_2, \ldots, w_p$

(typically, $p >= m$)

Υ



Distributed TensorFlow

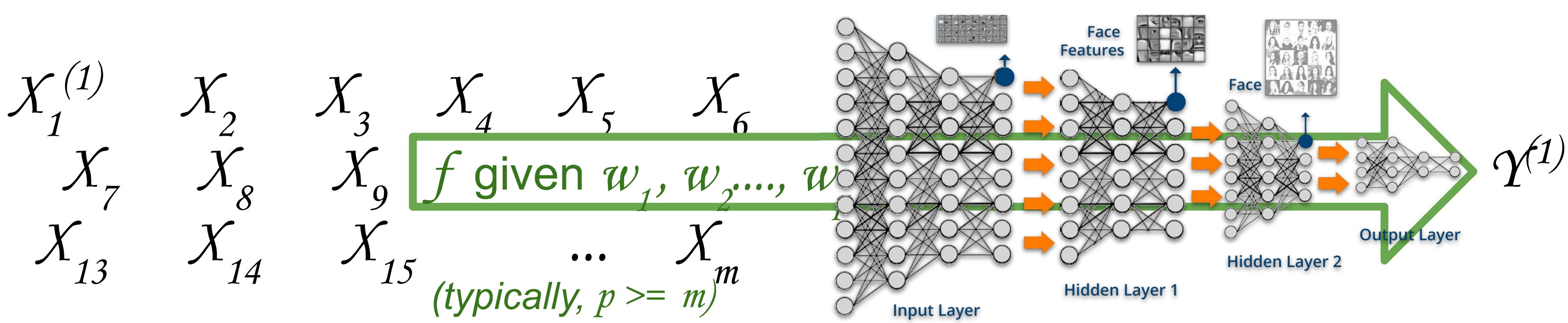
$$f(X|W) = \hat{Y}$$
$$Y = (X|W) + \varepsilon$$
$$Y = \hat{Y} + \varepsilon$$
$$\varepsilon = \hat{Y} - Y$$

Typical use-case:

Determine weights, $\mathcal{W}$, of a function, f, such that ε is minimized: f

$\mathcal{W}$ determined through *gradient descent*:

back propagating error across the network that defines f.



Distributed TensorFlow

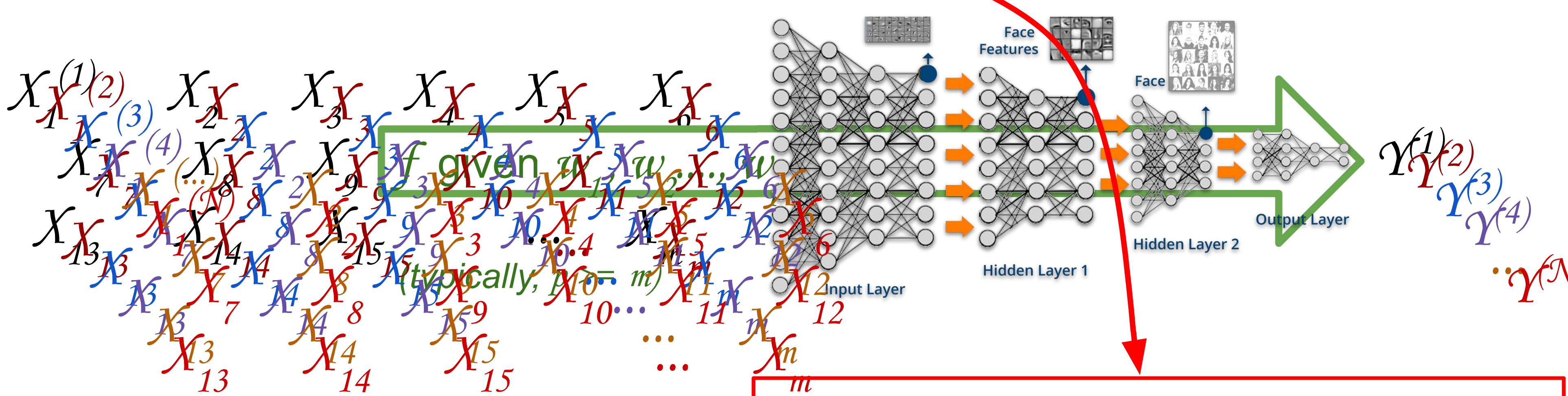
Typical use-case:

Determine weights, $\mathcal{W}$, of a function, f, such that ε is minimized:

$$f(X|W) = \hat{Y}$$
$$Y = (X|W) + \varepsilon$$
$$Y = \hat{Y} + \varepsilon$$
$$\varepsilon = \hat{Y} - Y$$

$\mathcal{W}$ determined through *gradient descent*:

back propagating error across the network that defines f.



minimizes ε on ***N*** training examples