1. TensorFlow

History
Programming Model
Architecture
Linear Algebra

TensorFlow - Genesis

- Pre-TF – (Math libaries without distribution)
  - Matlab
  - SciPy
  - Octave
- Computational Graph, Automatic Differentiation
  - Theano
  - Torch
- On comes TensorFlow
  - Distributed, GPU-support
  - MxNet, CNTK also has similar capabilities
  - Keras is a higher level DL framework that can plug into TF, CNTK or Theano
- Popular non-differentiating approaches
  - Spark, Flink
  - H2O.ai

TensorFlow is not a panacea

- Actively evolving – current version is 1.7. De facto path for deep-learning solutions. But it has excellent community support
- Over 600 operations. Programming model can become more elegant
- Sometimes it can get cumbersome
  - MatLab: `A[:, 3:10]`
  - TensorFlow: `tf.transpose(tf.gather(tf.transpose(X), tf.range(3, 5)))`
- Mostly designed well, but you can see evidence of flaws

```
tf.while_loop(cond, body, loop_variables,
              parallel_iterations=10, back_prop=True, swap_memory=False, name=None)
```

What's that got to do with a while loop??

- Highly optimized for a specific class of problems
- Poor sparse support – Has been improving recently!
- Aggregations are limited. No custom aggregations

Tensors

- Generalization of Vectors & Matrices

- Has,
  - Shape
  - Data Type (float32, int32, string, etc.)

- Rank of tensor – Think dimensions of an array.
  - 1-Rank Tensor is called a Vector
  - 2-Rank Tensor is called a Matrix

- Main Types are,
  - `tf.Variable`
  - `tf.constant`
  - `tf.placeholder`
  - `tf.SparseTensors`

Tensors

- Changing the shape of a tensor

```
rank_three_tensor = tf.ones([3, 4, 5])
matrix = tf.reshape(rank_three_tensor, [6, 10])
```

- Casting the data type of a tensor

```
float_tensor = tf.cast(
        tf.constant([1, 2, 3]),
        dtype=tf.float32)
```

Constants & Variables

- Constants

```
x = tf.constant([35, 40, 45], name='x')
```

- Variables

```
W = tf.Variable(tf.ones([100,5]), name="W")
```

  - You could also use get_variable to create a Variable

```
my_var = tf.get_variable("my_var", [1, 2, 3])
```

- Variables get placed into two collections by default

```
tf.GraphKeys.GLOBAL_VARIABLES
tf.GraphKeys.TRAINABLE_VARIABLES
```

  - You can add your own collections using tf.add_to_collection

Placeholders

- Represents a data element that would be bound later.
  - Helps models be data agnostic
  - Model tend to load faster

- Declaring Placeholders

```
X = tf.placeholder(tf.float32,
                   shape=[None, 3],
                   name='X')
```

- Use them as any Tensor

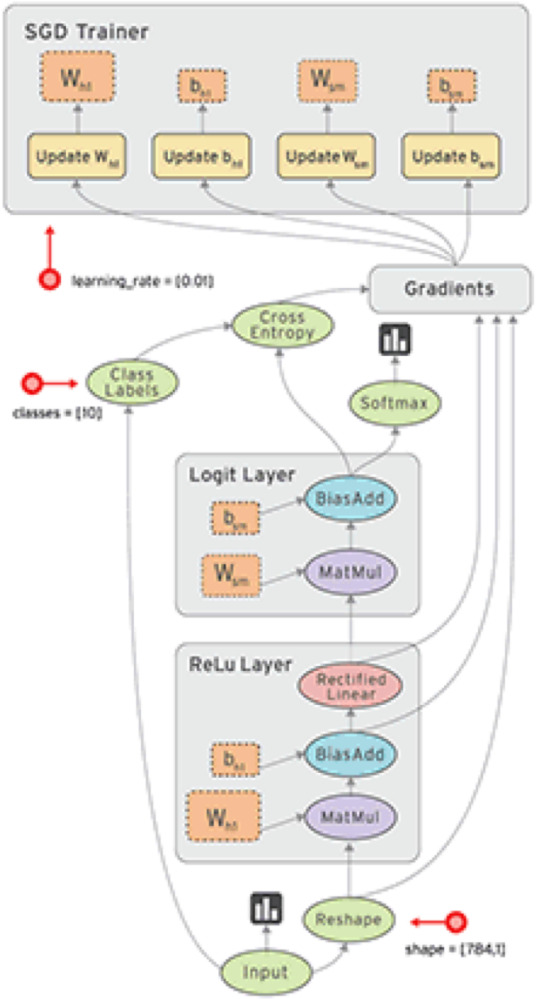
```
wX = tf.matmul(X, w)
```

- Feeding Data to a placeholder

```
session.run(wX, feed_dict={
                    x: [[1, 2, 3], [4, 5, 6]]})
```

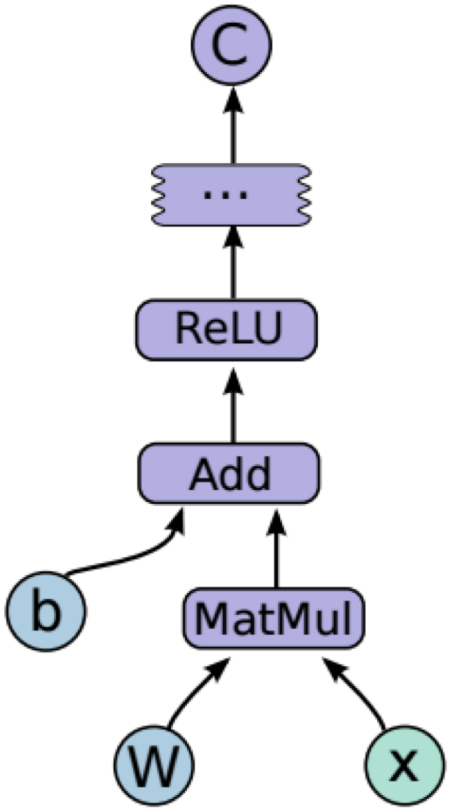
Graphs & Sessions

- Tensorflow is a declarative programming environment
  - Data flow graph represents the computation
    - Authored in language of choice. Our choice is Python, but TF support C++, Java and Go.
    - CAUTION: Only Python is covered by TF API stability promise
  - Session executes the data flow
    - Executors are written in C and are intrinsic to Tensorflow



Graphs

- Nodes can be
  - Data nodes
  - Operations
  - Summaries
- Edges show the data flow
- Data nodes can be,
  - Variables
  - Constants
  - Placeholders



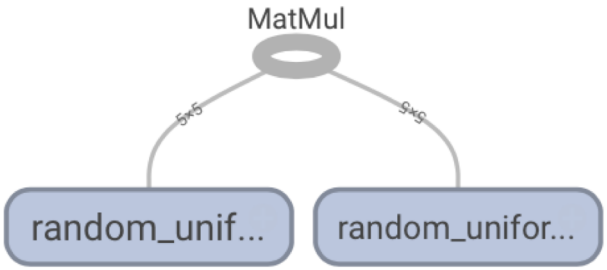
Source: TensorFlow Paper, Google Research

Program execution

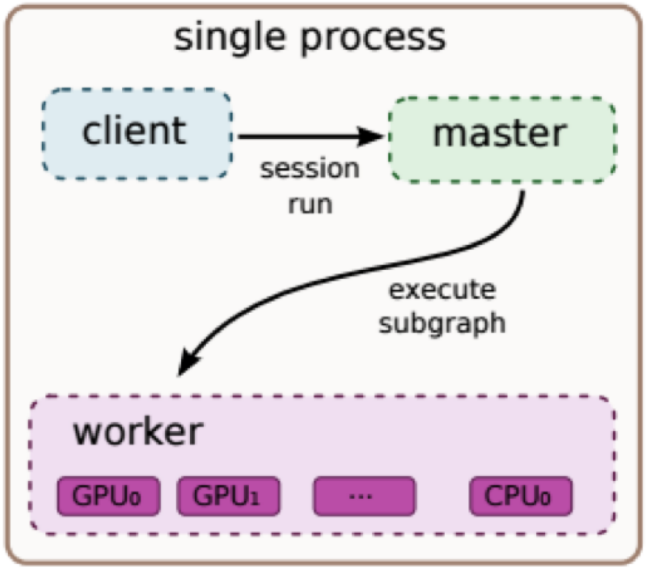
Write Code

```
a = tf.random_uniform([5, 5])
b = tf.random_uniform([5, 5])
c = tf.matmul(a, b)
```

Creates an execution graph




```
with tf.Session() as session:
    result = session.run(c)
```



Source: TensorFlow Paper, Google Research

LET'S CODE OUR FIRST GRAPH

November 9 2015

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems

(Preliminary White Paper, November 9, 2015)

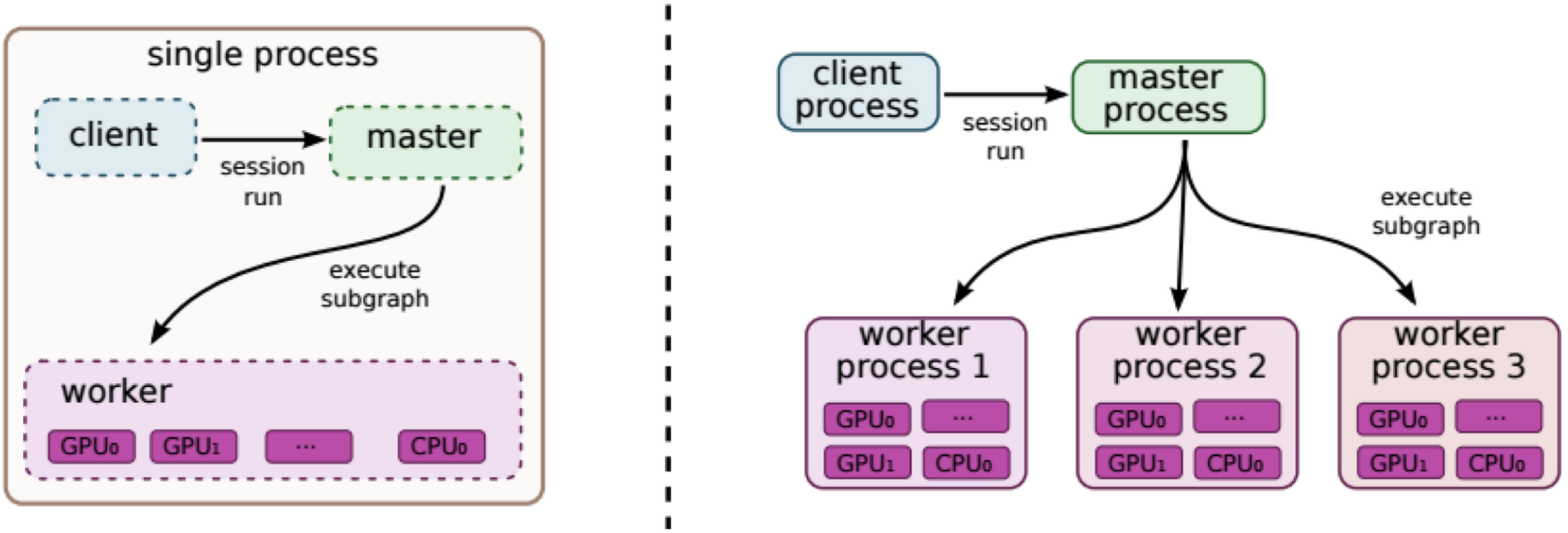
Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

Abstract

TensorFlow [1] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and sequence prediction [47], move selection for Go [34], pedestrian detection [2], reinforcement learning [38], and other areas [17, 5]. In addition, often in close collaboration with the Google Brain team, more than 50 teams at Google and other Alphabet companies have deployed deep neural networks using DistBelief in a wide variety of products, including Google Search [11], our advertising products, our speech recognition systems [50, 6, 46], Google Photos [43], Google Maps and StreetView [19], Google Translate [18], YouTube, and many others.

Based on our experience with DistBelief and a more complete understanding of the desirable system properties and requirements for training and using neural networks, we have built TensorFlow, our second-generation system for the implementation and deployment of large-scale machine learning models. TensorFlow takes com-

Distributed execution



Source: TensorFlow Paper, Google Research

Partial Execution

- After defining the computational graph we can turn evaluate some parts
- Partial update is not necessarily incremental update
- Incremental update is possible but hard to debug

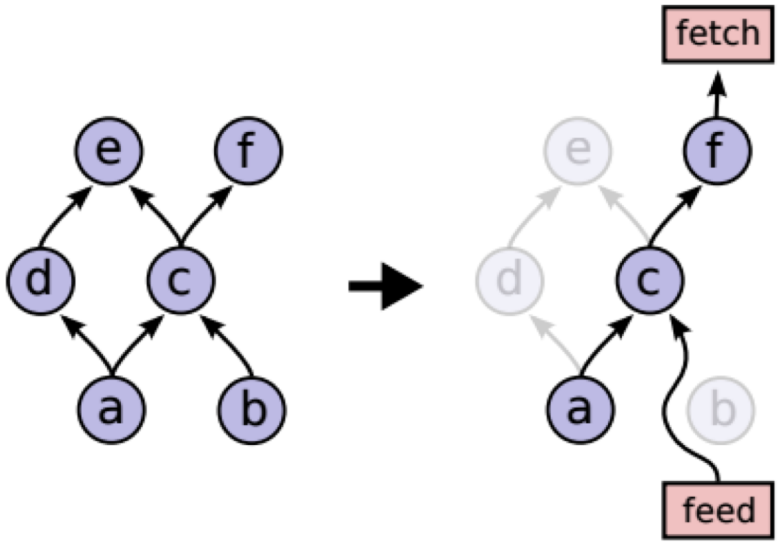


Figure 6: Before and after graph transformation for partial execution

Source: TensorFlow Paper, Google Research

Data Parallel execution

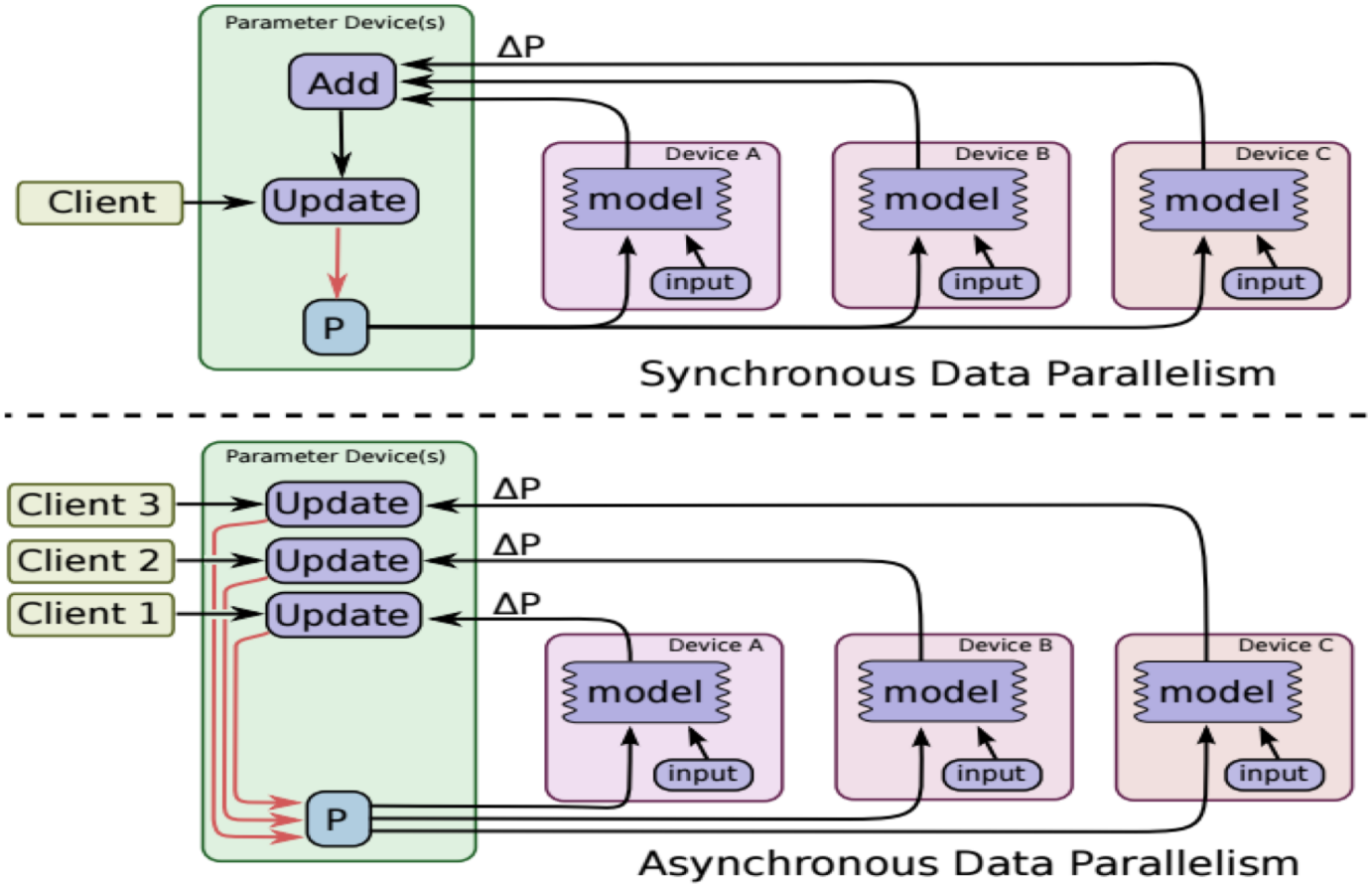


Figure 7: Synchronous and asynchronous data parallel training

Source: TensorFlow Paper, Google Research

Model Parallel execution

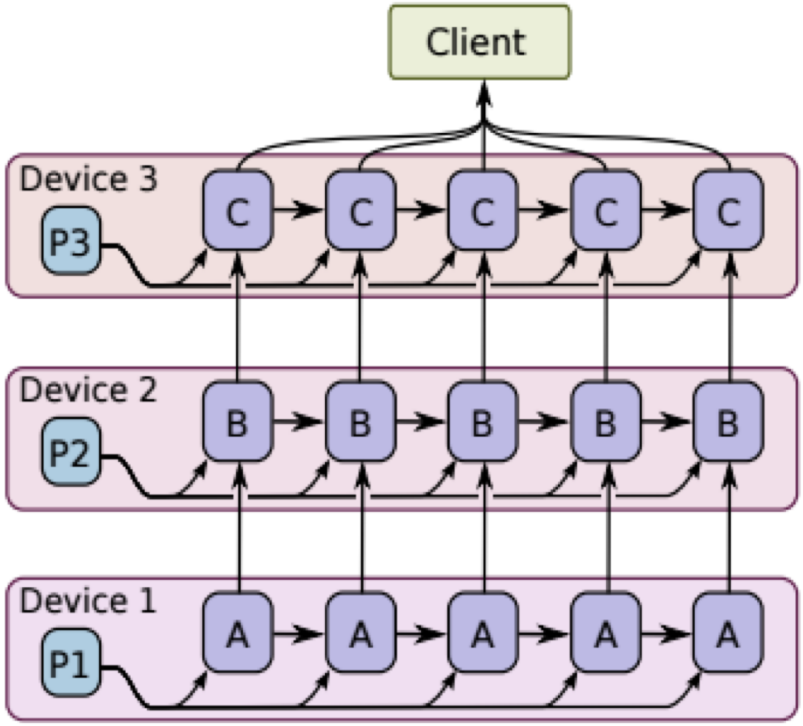


Figure 8: Model parallel training

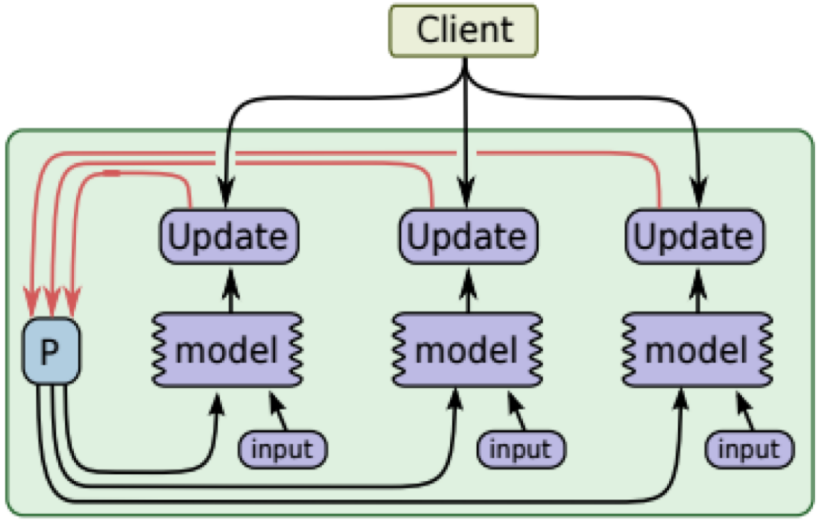


Figure 9: Concurrent steps

Source: TensorFlow Paper, Google Research

Debugging – Tensorboard is your friend

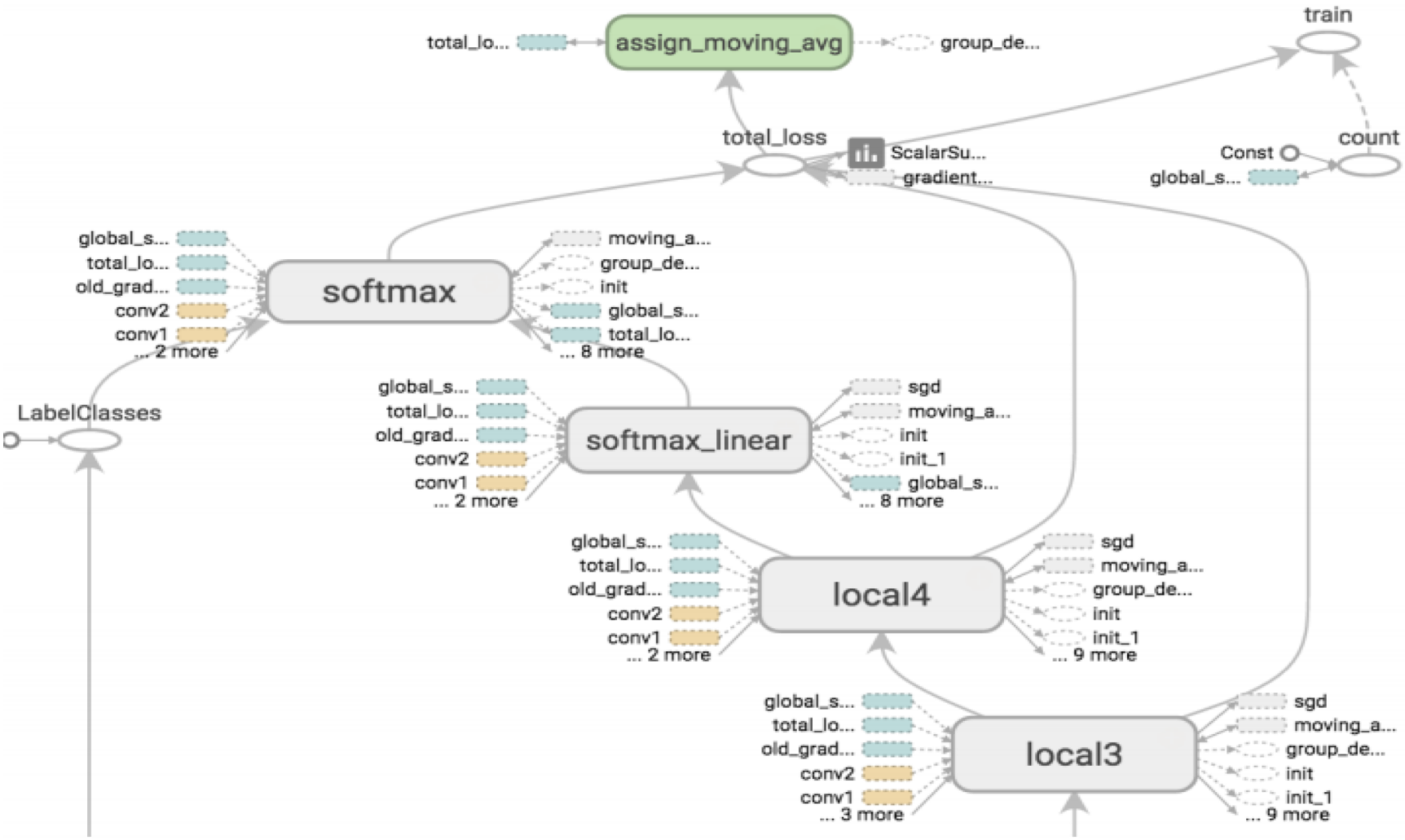


Figure 10: TensorBoard graph visualization of a convolutional neural network model

Source: TensorFlow Paper, Google Research

Linear Algebra

- Vector

$$a = \begin{bmatrix} 2 & 5 & 0 & 9 \end{bmatrix}$$

- Matrix

$$X = \begin{bmatrix} 2 & 5 & 0 & 9 \\ 5 & 2 & 3 & 4 \\ 9 & 4 & 7 & 5 \\ 8 & 1 & 2 & 3 \end{bmatrix}$$

- Tensor

    n-Dimensional Array/Matrix

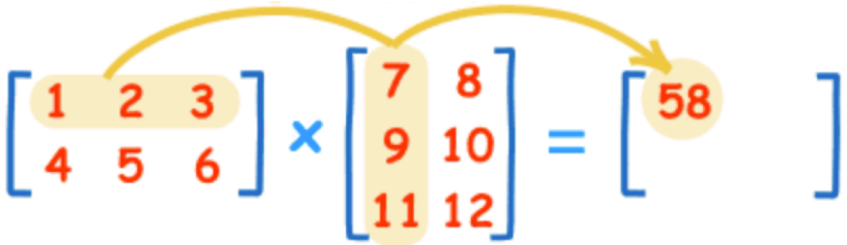
Matrix Manipulation

- Addition/Substraction

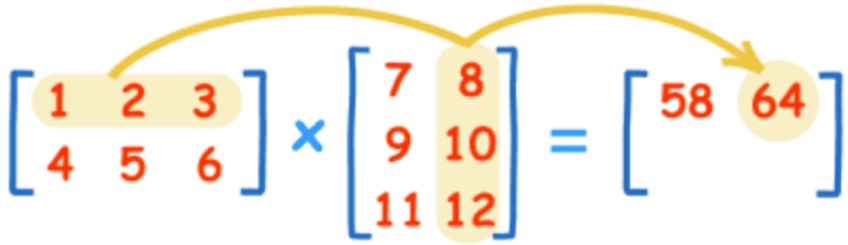
$$\begin{bmatrix} 2 & 5 & 0 & 9 \\ 5 & 2 & 3 & 4 \\ 9 & 4 & 7 & 5 \\ 8 & 1 & 2 & 3 \end{bmatrix} + \begin{bmatrix} 4 & 4 & 0 & 7 \\ 3 & 7 & 1 & 3 \\ 5 & 4 & 4 & 6 \\ 7 & 5 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 6 & 9 & 0 & 9 \\ 5 & 2 & 3 & 4 \\ 9 & 4 & 7 & 5 \\ 8 & 1 & 2 & 3 \end{bmatrix}$$

Multiplication



$(1, 2, 3) \bullet (7, 9, 11) = 1\times7 + 2\times9 + 3\times11 = 58$



$(1, 2, 3) \bullet (8, 10, 12) = 1\times8 + 2\times10 + 3\times12 = 64$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Source: MathIsFun.com

Transpose

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

Source: MathIsFun.com

Identity and Inverse

- Identity Matrix

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A x I = A

- Inverse of a Matrix

$A \times A^{-1} = I$

Source: MathIsFun.com

LET'S TRY TO CODE THIS IN TF