FRAMEWORKS FOR AI: TENSORFLOW*

LEGAL NOTICES AND DISCLAIMERS

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

This sample source code is released under the Intel Sample Source Code License Agreement.

Intel, the Intel logo, neon, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

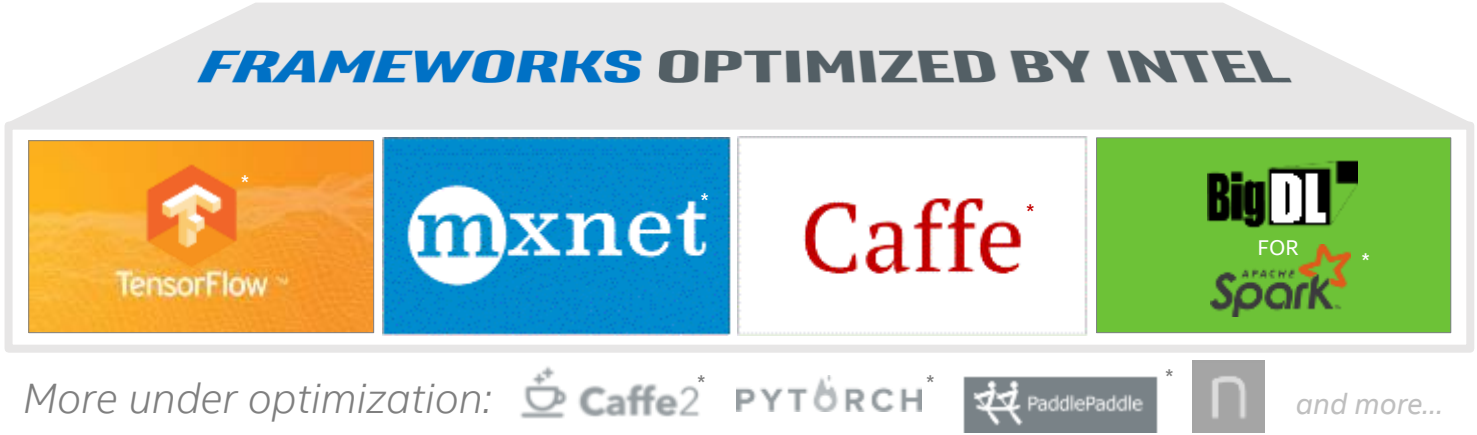
*Other names and brands may be claimed as the property of others.

OUTLINE: WHAT STUDENTS ARE EXPECTED TO LEARN

- Overview of Optimized Deep Learning Frameworks
- Introduction to TensorFlow*
- Why Tensorflow on PC
- Key Features
- Data Flow Graphs
- Tensors
- Constants
- Variables
- Placeholders
- Sessions: Instantiation and Execution
- Example: Defining and Running a Simple Graph
- Intel® Optimizations
- Summary

OPTIMIZED DEEP LEARNING FRAMEWORKS

Install an Intel-optimized framework and featured topology



Get started today at ai.intel.com/framework-optimizations/

SEE ALSO: Machine Learning Libraries for Python (Scikit-learn, Pandas, NumPy), R (Cart, randomForest, e1071), Distributed (MLLib on Spark, Mahout)
**Limited availability today*
Other names and brands may be claimed as the property of others.

TENSORFLOW*

TENSORFLOW*

- TensorFlow* is an open source library for numerical computation and large-scale machine learning created by Google.
- TensorFlow* bundles together a number of machine learning and deep learning models and algorithms and makes them useful by way of a common metaphor.
- TensorFlow* allows developers to create *dataflow graphs*—structures that describe how data moves through a graph, or a series of processing nodes.

KEY FEATURES

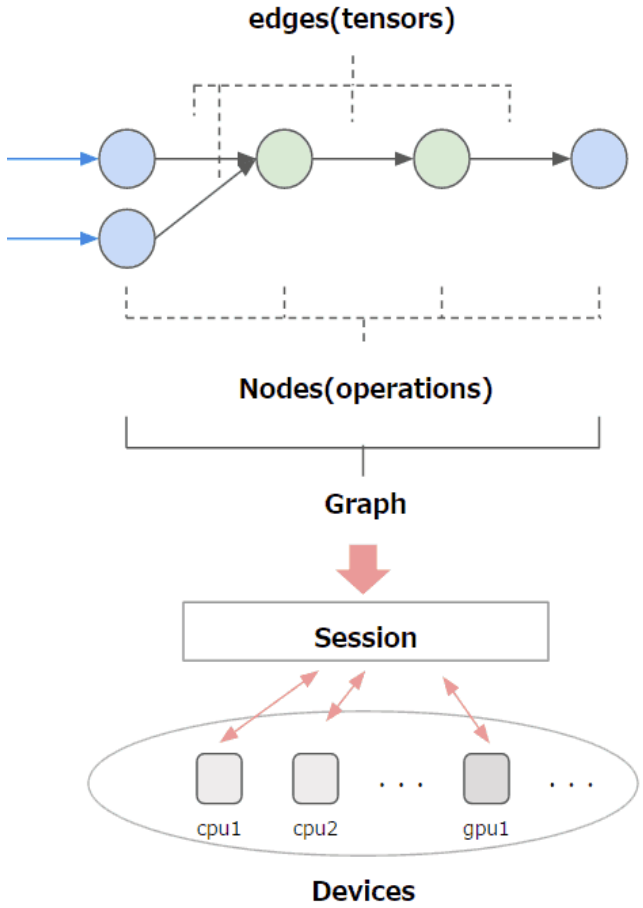
- Framework to efficiently perform calculations on multidimensional arrays and tensors.
- Supports operations required for machine learning and deep learning.
- Supports both CPU and GPU computation.
- Highly scalable for distributed applications.
- TensorFlow* provides for *abstraction.* Instead of dealing with the details of implementing algorithms, or figuring out how to move the output of one function to the input of another, the developer can focus on the overall logic of the application.

WHY TENSORFLOW ON PC

- AI on PC in relation to Tensorflow is about bringing cloud/server level technology to the PC.
- Some local training of models and their test evaluation are possible.
- As such Tensorflow on the PC is a great vehicle for experimentation.

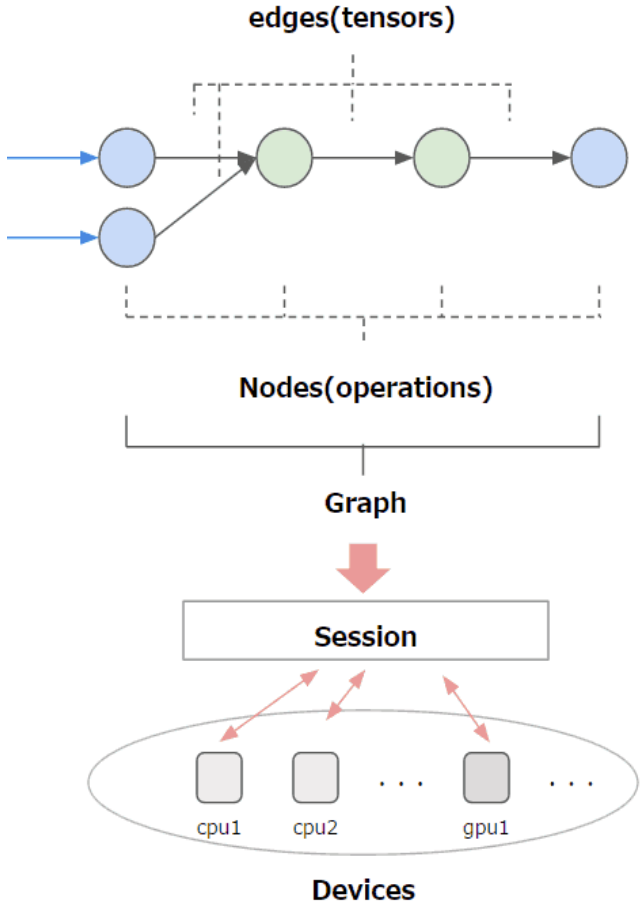
DATA FLOW GRAPHS

- Also known as “Computational graph”.
- Computation is described through operations.
- Node: Instance of mathematical operation.
- example: Addition, Multiplication ...etc
- Edge: Multi-dimensional data set(Tensor).
  - Two types:
    1. Normal Edge
    2. Special Edge

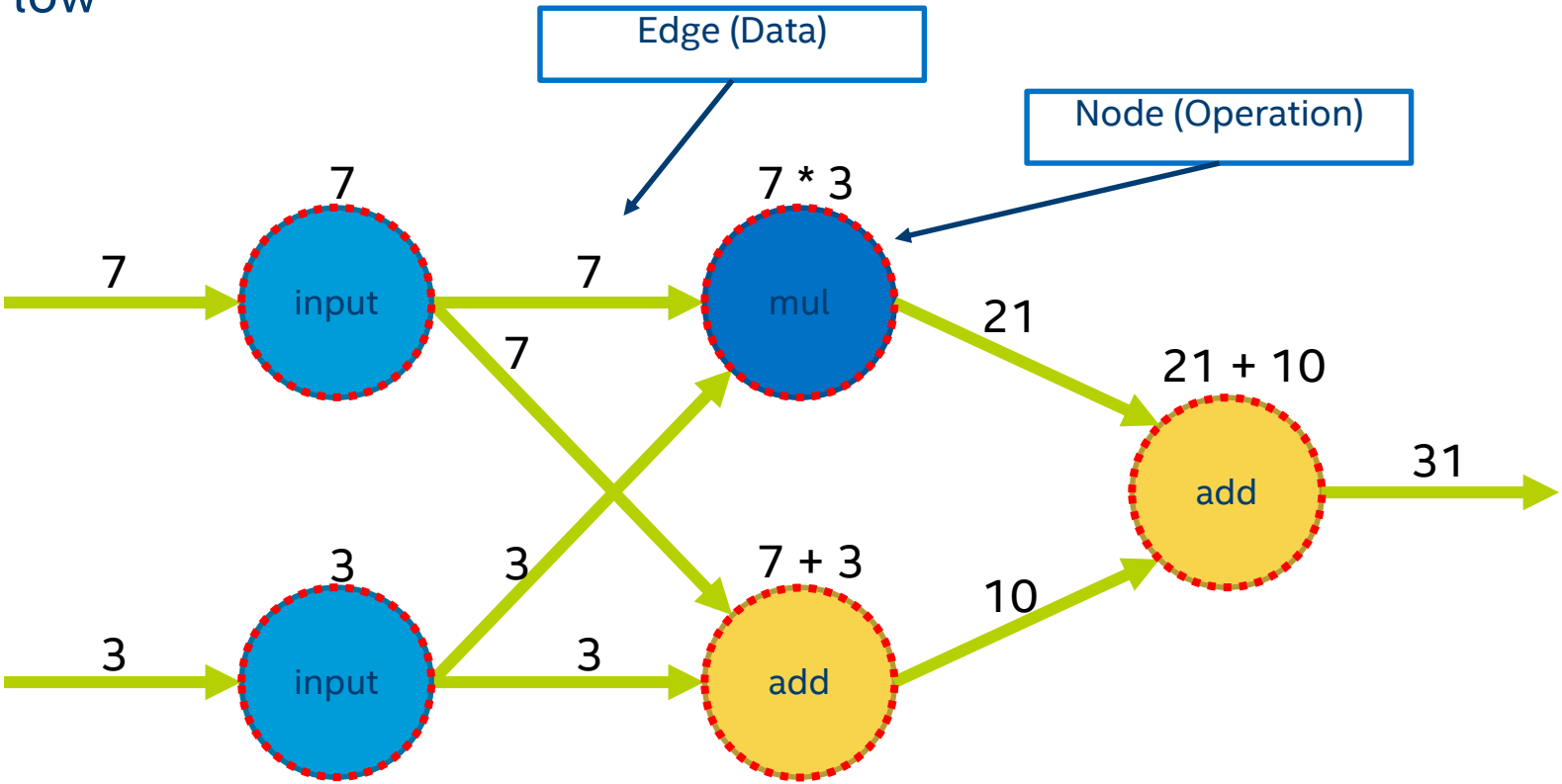


DATA FLOW GRAPHS: DEVICE PLACEMENT

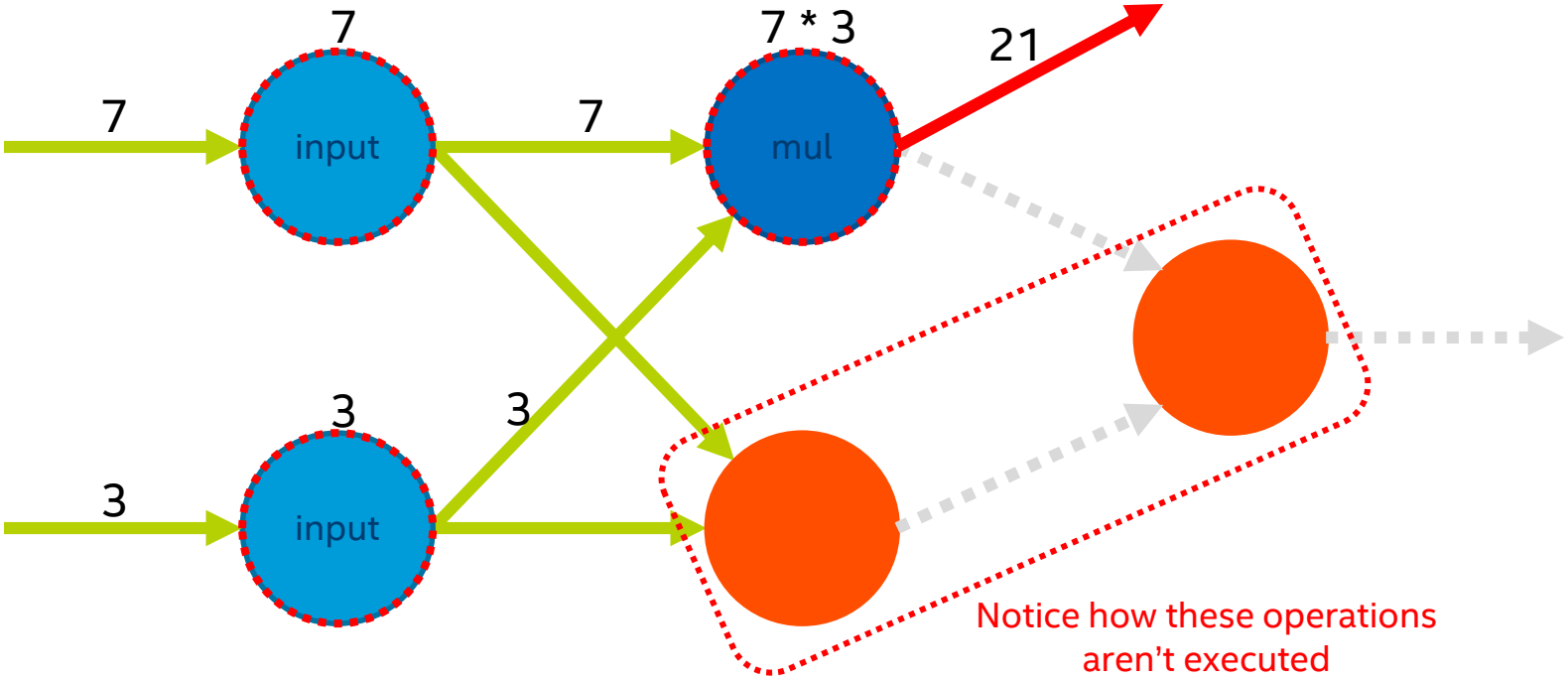
- TensorFlow* supports both CPU and GPU, as well as multiple devices.
- Which device an operation will be computed on can be explicitly specified using the a ‘with’ scope
- E.g.
  with tf.device(‘/cpu:0’):
    a = tf.constant(32)



Data Flow



Partial Execution

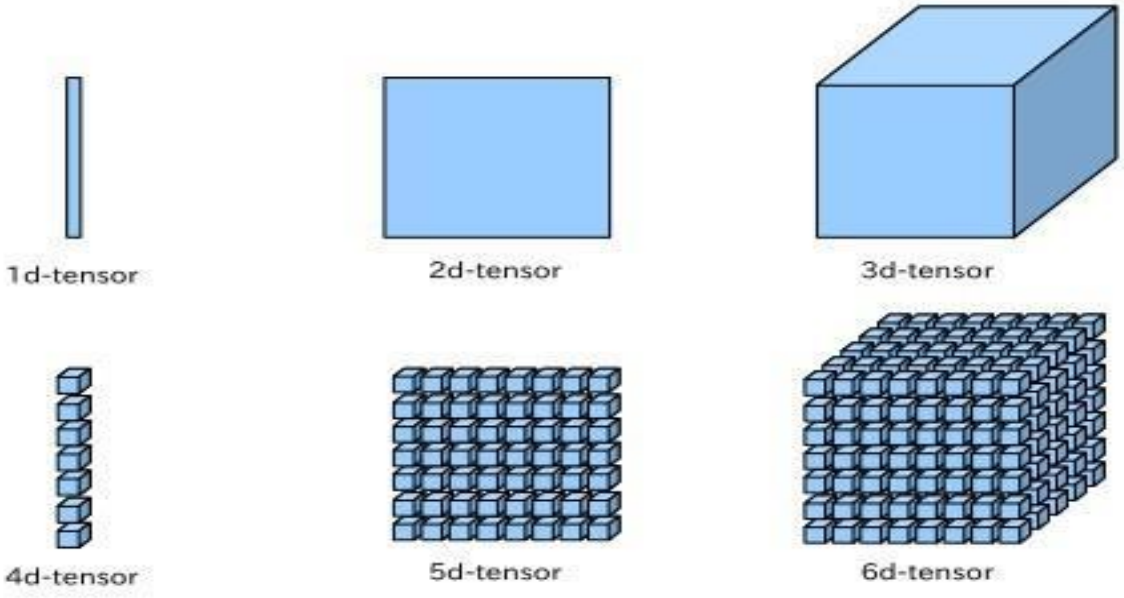


TENSORS

- Generalization of vectors and matrices to potentially higher dimensions.
- Represented as n-dimensional arrays of base datatypes.
- tf.Tensor Object has two properties:
  - A data type(float32, int32, ….etc.)
  - Shape
- Some main types of tensors:
  - tf.Variable
  - tf.constant
  - tf.Placeholder

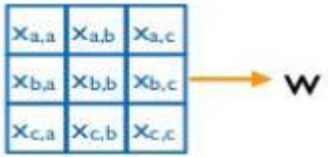
What's Tensor

Tensor is a general name of multi-way array data. For example, 1d-tensor is a vector, 2d-tensor is a matrix and 3d-tensor is a cube. We can image 4d-tensor as a vector of cubes. In similar way, 5d-tensor is a matrix of cubes, and 6d-tensor is a cube of cubes.



TENSORS

Define Tensors



Variable(<initial-value>,
name=<optional-name>)

```
import tensorflow as tf
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
```

Variable stores the state of current execution

Others are operations

CONSTANTS

- Represent constant values in the TensorFlow graph.
- constant(value, dtype=None, shape=None, name='Const', verify_shape=False).
  - value: Actual constant value
  - dtype: data type parameter
  - shape: dimension (optional)
  - name: name for tensor(optional)
  - verify_shape= Boolean which indicates verification of the shape of values.

VARIABLES

- Hold values which are subject to change.
- In-memory buffers.
- Have to be explicitly initialized.
- Variables are especially useful once you start with training models.
- Used to hold and update parameters.
- Define as:

```
k=tensorflow.Variable(tensorflow.zeros([1]),name="k")
```

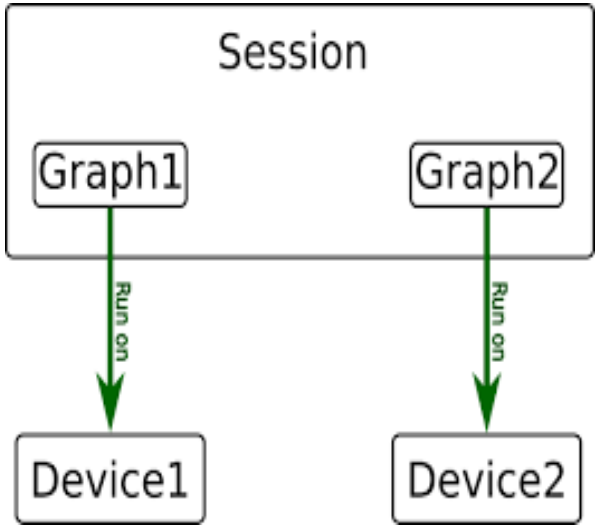
PLACEHOLDERS

- Abstract representation of data including its type and shape.
- Do not have to be explicitly initialized.
- Represent data which will be given concrete values during execution.
- Commonly used as the inputs of a graph, where data will be passed in to the network.
- Example, creation of a N x 32 x 32 float placeholder:

```
placeholder = tensorflow.placeholder(tf.float32, shape=[?,32,32],
name="input")
```

SESSION

- Manage resource for graph execution.
- In order to actually evaluate the nodes, we must run a computational graph within a session.
- Session encapsulates the control and state of the TensorFlow* runtime.
- Session may be explicitly passed a graph, otherwise it will run on the default graph.



SESSION CREATION

- Both type of the session creation has same behaviour.

```
>>> sess = tf.Session()
>>> print(sess.run(c))

>>> with tf.Session() as sess:
>>>     print(sess.run(c))
>>>     print(c.eval())
```

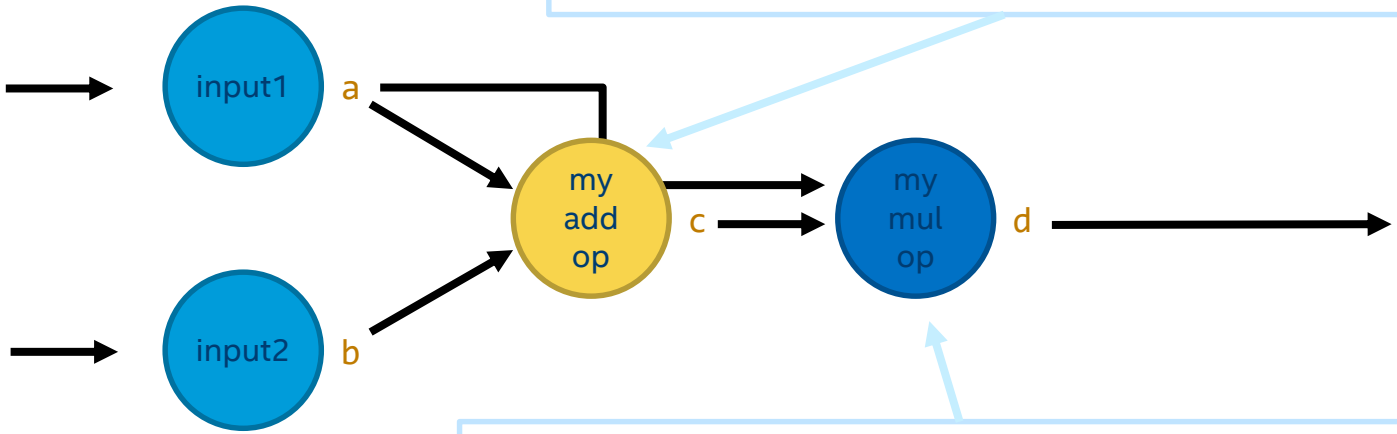
Syntactic sugar for `sess.run(c)` in the current active session.
(Must locate inside of the `tf.Session` scope)

EXAMPLE: DEFINING AND RUNNING A SIMPLE GRAPH

- Create placeholders as inputs.
- Add operations.
- Instantiate a Session.
- Create a feed_dict object – feed_dicts feed values to TensorFlow placeholders.
- Run the graph using the session.

```
>>> a = tf.placeholder(tf.float32, name="input1")
```

```
>>> c = tf.add(input1, input2, name="my_add_op")
```

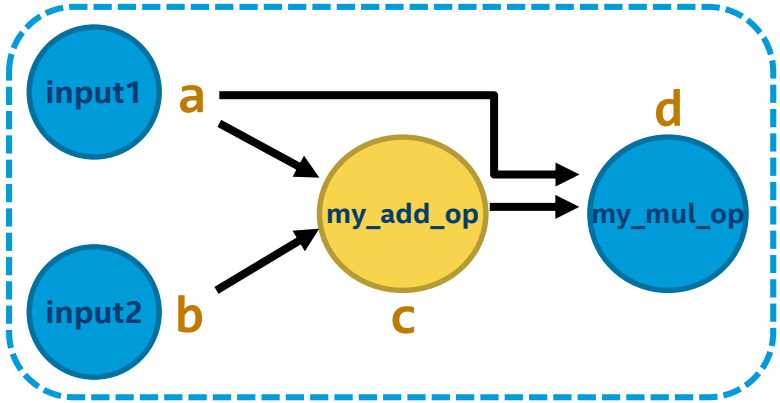



```
>>> d = tf.matmul(input1, my_add_op, name="my_mul_op")
```

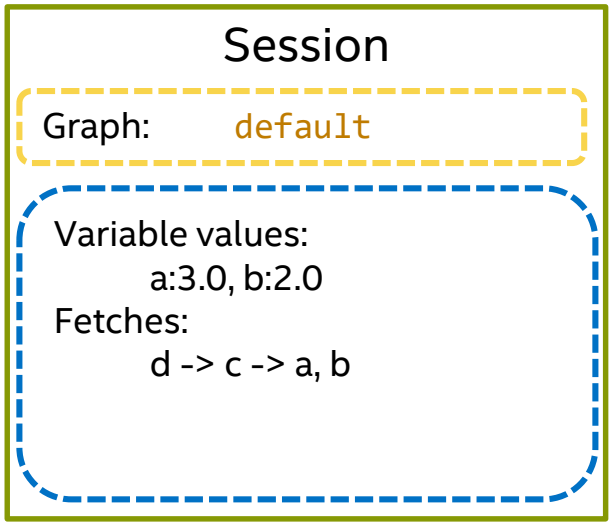
```
>>> b = tf.placeholder(tf.float32, name="input2")
```

```
>>> with tf.Session() as sess:
>>>    out = sess.run(d, feed_dict={a:3.0, b:2.0})
>>>    print(out)

15
```

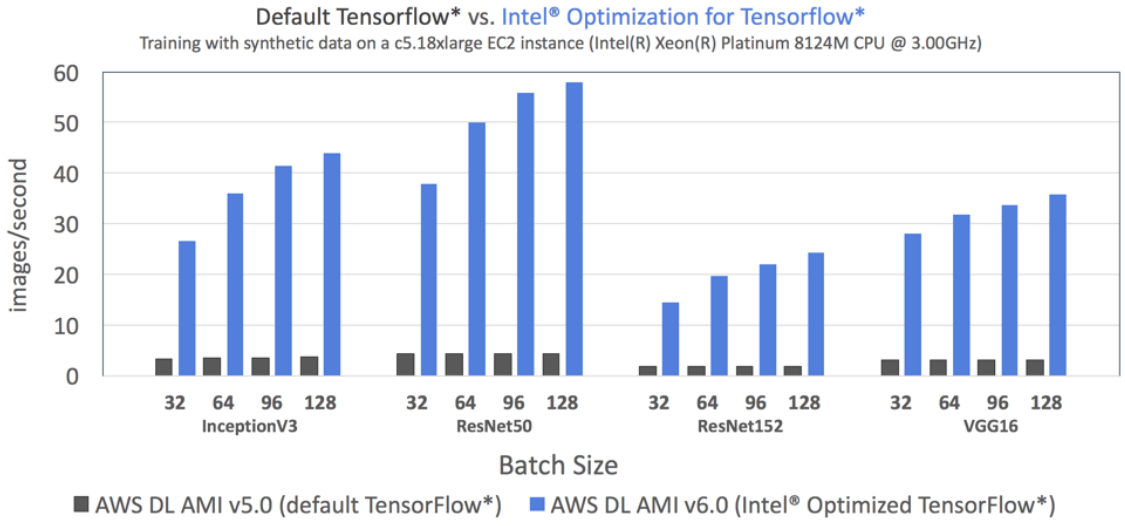


feed_dict: {a: 3.0, b: 2.0}



WHY INTEL OPTIMIZATION FOR TENSORFLOW?

- The latest releases of TensorFlow* now come with **Intel® Math Kernel Library (Intel® MKL)** optimizations standard.
- Leads to dramatic performance improvements with no modifications at the model level.



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance

INTEL® OPTIMIZATION FOR TENSORFLOW*

- Intel developed a number of optimized deep learning primitives.
- Intel® MKL provides primitives for most widely used operations implemented for vectorization-friendly data layout:
  - Direct batched convolution.
  - Inner product.
  - Pooling: maximum, minimum, average.
  - Normalization.
  - Activation: rectified linear unit (ReLU).
  - Data manipulation: multi-dimensional transposition (conversion), split, concat, sum and scale.

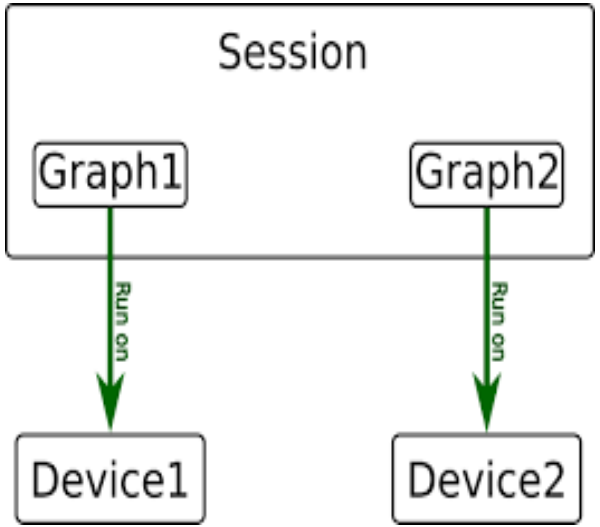
INTEL® OPTIMIZATION FOR TENSORFLOW*

We introduced a number of graph optimization passes to:

- 1. Replace default TensorFlow* operations with Intel optimized versions when running on CPU. This ensures that users can run their existing Python* programs and realize the performance gains without changes to their neural network model.
- 2. Eliminate unnecessary and costly data layout conversions.
- 3. Fuse multiple operations together to enable efficient cache reuse on CPU.
- Handle intermediate states that allow for faster backpropagation.

SESSION: CONFIGPROTO

- Offers further configurability of the Session object
- Environment variables such as inter_op_parallelism_threads, intra_op_parallelism_threads, and kmp_blocktime



SUMMARY

SUMMARY

- **TensorFlow*** is system for machine learning and deep learning computations based on the concept of dataflow graphs – and that tensors are the main data that flow through these graphs.
- **TensorFlow on the PC** enables some training and inference capabilities at the Edge.
- **Dataflow graphs** have description of input and output nodes, as well as intermediary nodes that express mathematical operations. The nodes are connected by edges that represent the data that flows between them.
- **Tensors** are a generalization of vectors and matrices to potentially higher dimensions and are represented as n-dimensional arrays of base datatypes.
- **Constants**, **Variables** and **Placeholders** each represent constant data values, data values that can change, or simply define the shape of data to be filled in later in a computation.
- **Sessions** are used to evaluate TensorFlow* graphs. When using TensorFlow*, all objects are left as operations until evaluated by a Session object. Session encapsulates the control and state of the TensorFlow* runtime.
- Intel developed a number of optimized deep learning primitives for TensorFlow* via **Intel® MKL** primitives for most widely used operations implemented for vectorization-friendly data layout.

experience
what's inside™