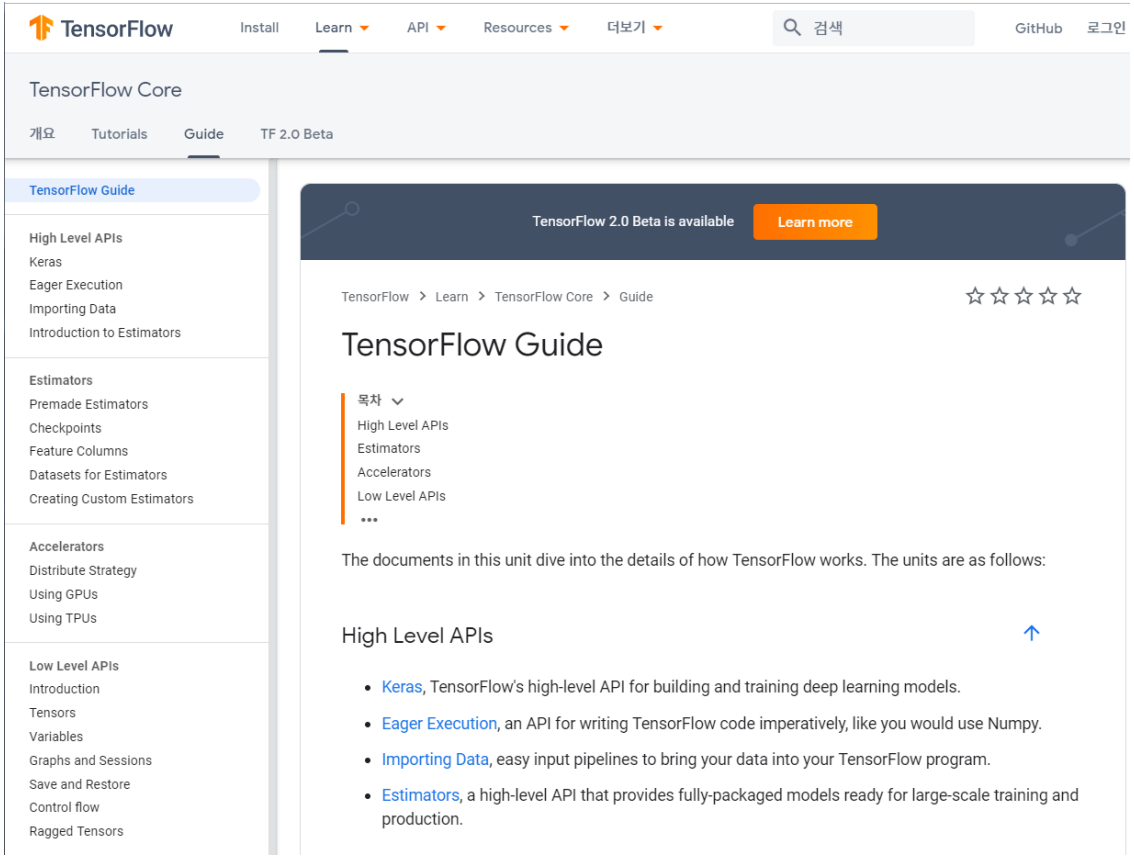
TensorFlow 2.0 Basics

Dong Kook Kim

Reference

https://www.tensorflow.org/guide?hl=ko

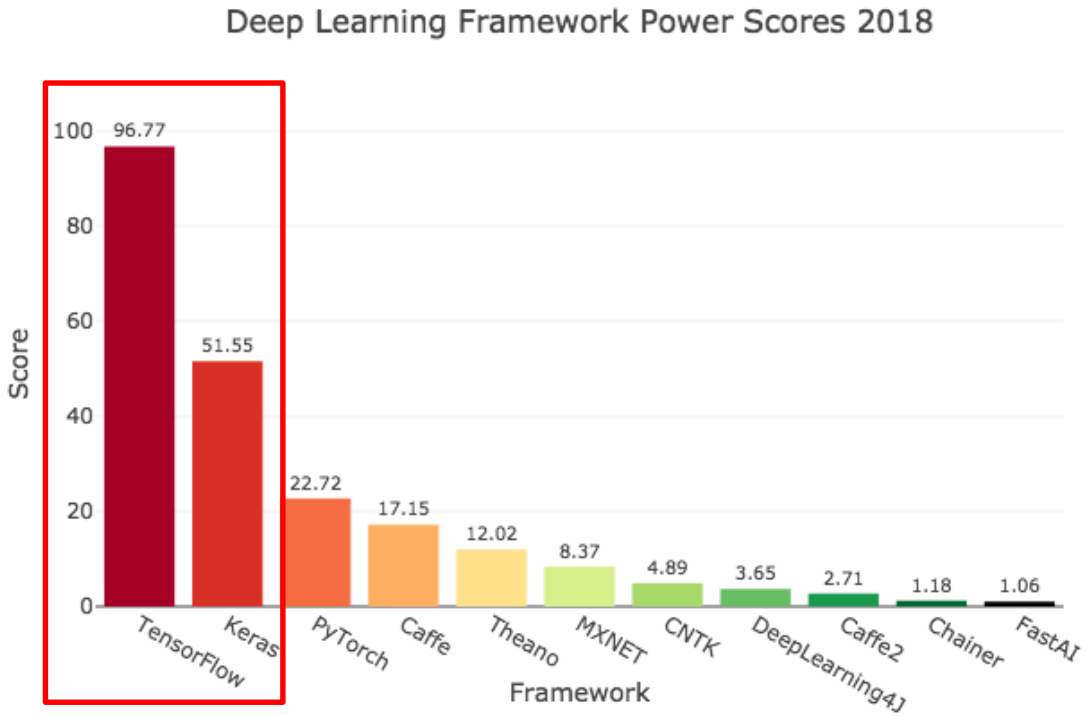


Comparison of DL software

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

Software	Creator	Software license[6]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Apache SINGA	Apache Incubator	Apache 2.0	Yes	Linux, Mac OS X, Windows	C++	Python, C++, Java	No	Yes	Yes	?	Yes	Yes	Yes	Yes	Yes
Caffe	Berkeley Vision and Learning Center	BSD license	Yes	Linux, Mac OS X, Windows[2]	C++	Python, MATLAB	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, Mac OS X, Windows, Android (Cross-platform)	java	Java, Scala, Clojure, Python (Keras)	Yes	On roadmap[5]	Yes[6]	Computational Graph	Yes[7]	Yes	Yes	Yes	Yes[8]
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Keras	François Chollet	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python	Only if using Theano as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes[9]	Yes	Yes	Yes	Yes[10]
MatConvNet	Andrea Vedaldi,Karel Lenc	BSD license	Yes	Windows, Linux[11] (OSX via Docker on roadmap)	C++	MATLAB, C++,	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[12]	Yes	Windows, Linux[13] (OSX via Docker on roadmap)	C++	Python, C++, Command line,[14] BrainScript[15] (.NET on roadmap[16])	Yes[17]	No	Yes	Yes	Yes[18]	Yes[19]	Yes[19]	No[20]	Yes[21]
MXNet	Distributed (Deep) Machine Learning Community	Apache 2.0	Yes	Linux, Mac OS X, Windows,[22][23] AWS, Android,[24] iOS, JavaScript[25]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap[26]	Yes	Yes[27]	Yes[28]	Yes	Yes	Yes	Yes[29]
Neural Designer	Artelnics	Proprietary	No	Linux, Mac OS X, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	No	?	?	No	No	No	?
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, Mac OS X, Windows[30]	C++, Python	Python (Keras), C/C++, Java, Go, R[31]	No	On roadmap[32][33]	Yes	Yes[34]	Yes[35]	Yes	Yes	Yes	Yes
Theano	Université de Montréal	BSD license	Yes	Cross-platform	Python	Python	Yes	Under development[36]	Yes	Yes[37][38]	Through Lasagne's model zoo[39]	Yes	Yes	Yes	Yes[40]
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD license	Yes	Linux, Mac OS X, Windows,[41] Android,[42] iOS	C, Lua	Lua, LuaJIT,[43] C, utility library for C++/OpenCL[44]	Yes	Third party implementations[45][46]	Yes[47][48]	Through Twitter's Autograd[49]	Yes[50]	Yes	Yes	Yes	Yes[51]
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, Mac OS X, Linux, Cloud computing	C++	Wolfram Language	No	No	Yes	Yes	Yes[52]	Yes	Yes	Yes	Yes

TensorFlow/Keras : Best DL Framework



https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a/

TensorFlow Install Learn API Resources 더보기 검색 GitHub 로그인

TensorFlow 2.0 Beta is available Learn more

An end-to-end open source machine learning platform

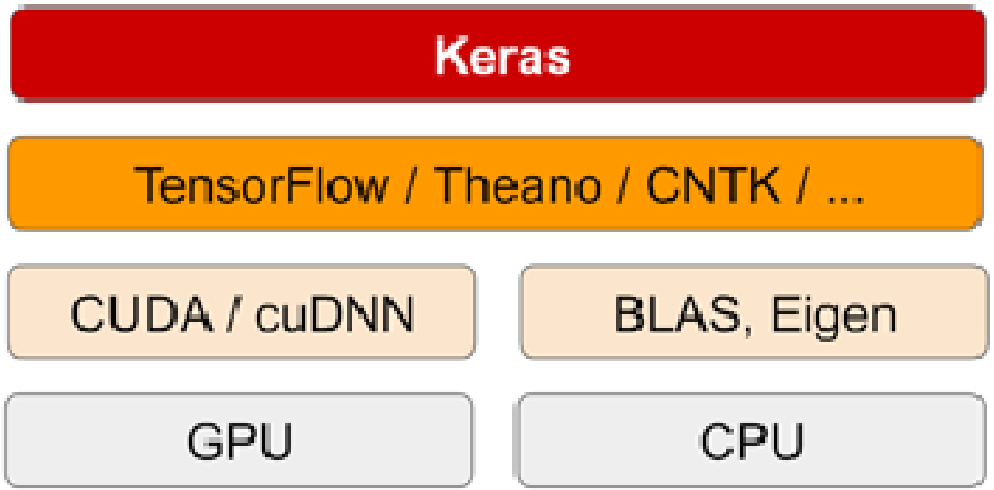
TensorFlow For JavaScript For Mobile & IoT For Production

The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

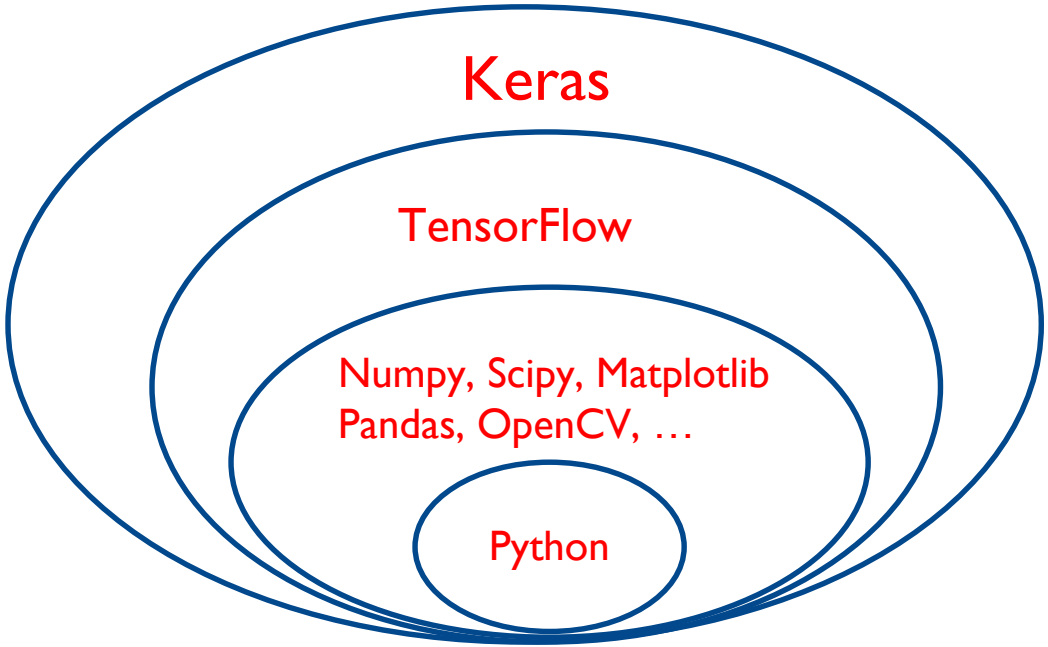
Get started with TensorFlow

https://www.tensorflow.org

DL SW and HW stack



DL Softwares



TensorFlow

- TensorFlow is an end-to-end open source platform for machine learning
- Created by Google (TF 1.x released 11/2015, TF 2.0 beta released 6/2019)
- Evolved from Google Brain
- Windows, Linux and Mac OS X support

What is TensorFlow 2.0 ?

- Support for Python, Java, C++
- Desktop, Server, Mobile, Web
- CPU/GPU/TPU support
- Visualization via TensorBoard
- Can be embedded in Python scripts

Major Changes in Tensorflow 2.0

- API cleanup – many APIs are either gone or moved
- Eager execution – default mode
- No more globals
- Generators : used with tf.data.Dataset
- Differentiation : calculates gradients
- tf.GradientTape : automatic differentiation
- Functions, not session
  - @tf.function decorator instead of tf.Session()
  - AutoGraph : graph generated in @tf.functions()

Removed from Tensorflow 2.0

- tf.Session()
- tf.placeholder()
- tf.global_initializer()
- Feed_dict
- Variable scopes
- tf.contrib code
- tf.flags and tf.contrib
- Global variables

→ Tensorflow 1.x functions moved to tf.compat.v1

Recommendations for Tensorflow 2.0

- Refactor your code into smaller functions
- Use Keras layers and models to manage variables
- Combine tf.data.Datasets and @tf.function
- Take advantage of AutoGraph with Python control flow
- tf.metrics aggregates data and tf.summary logs them

Tensorflow APIs

- Low Level APIs
  - Tensors and Operations
  - Variables
  - Graphs and Sessions
  - Save and Restore

- High Level APIs
  - Keras
  - Eager Execution vs. Graph Execution (TF 1.x)
  - Importing Data
  - Estimators

Low Level APIs

- TensorFlow use a dataflow graph to represent numeric computation in terms of the dependencies between individual operations.

- A computational graph is a series of TensorFlow operations arranged into a graph.

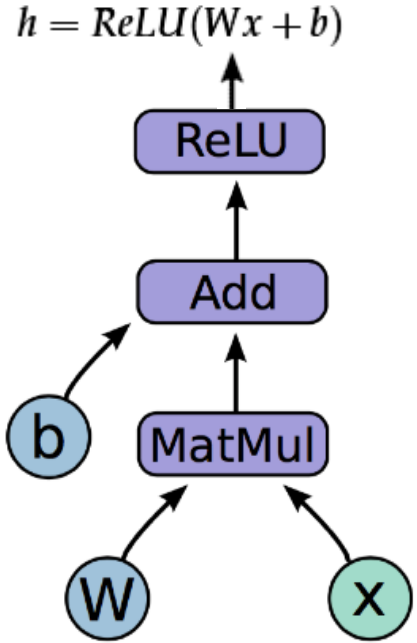
- The graph is composed of two types of objects
  - tf.Tensor : The edges in the graph
  - tf.Operation : the nodes of the graph

What is a Dataflow Graph ?

- Computational (dataflow) Graph

- Nodes in the graph represent mathematical operations
  - Graph nodes are operations which have any number of inputs and outputs

- Edges represent the multidimensional data arrays (tensors) communicated between them.
  - Graph edges are tensors which flow between nodes



Key Concepts for TensorFlow (TF 1.x)

- Represents data (edges) as tensors.
- Represents computations as graphs (nodes).
- Maintains state with Variables.

- Executes graphs in the context of Sessions.
- Uses feeds and fetches to get data into and out of arbitrary operations.

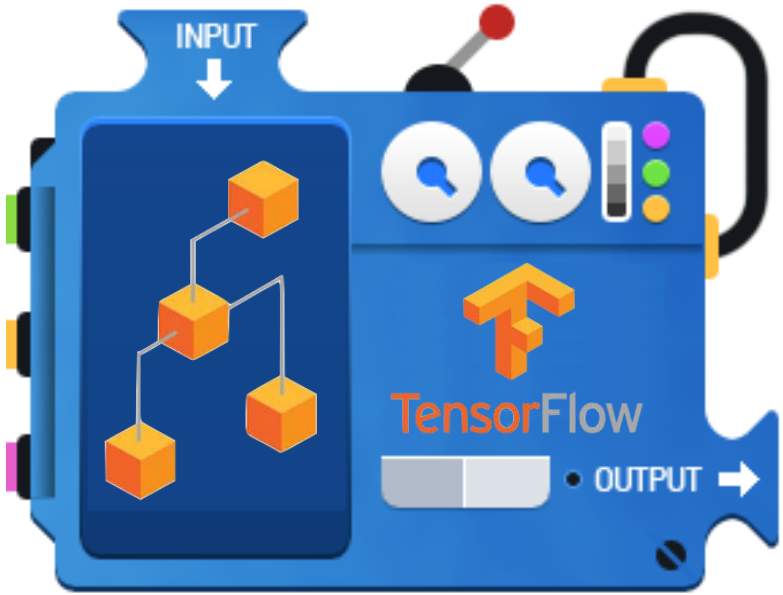
Two Computation Sections (TF 1.x)

1. Building the computational graph (tf.Graph)
   - create a graph to represent and train a neural network

2. Running the computational graph(using a tf.Session)
   - use a session to execute operations in the graph
   - repeatedly execute a set of training ops in the graph

TensorFlow Mechanics

2 feed data and run graph (operation)
sess.run (op)



1 Build graph using
TensorFlow operations

3 update variables
in the graph
(and return values)

WWW.MATHWAREHOUSE.COM

Exercise 01-1.

tf2-01-1-v1_example.py

tf2-01-1-v2_example.py

TensorFlow 1.x Examples

```
import tensorflow.compat.v1 as tf

tf.disable_v2_behavior()

W = tf.Variable([2.])
b = tf.Variable([1.])

x = tf.placeholder(tf.float32)

h = tf.nn.relu(W*x + b)

sess = tf.Session()
sess.run(tf.initialize_all_variables())

print("W = ", sess.run(W))
print("b = ", sess.run(b))
print("h = ", sess.run(h, feed_dict={x:1.}))
print("h = ", sess.run(h, feed_dict={x:-1.}))
```

1. Build graph using TensorFlow operations

2. feed data and run graph (operation) ***sess.run (op)***

3. update variables in the graph (and return values)

Results :

```
W =  [2.]
b =  [1.]
h =  [3.]
h =  [0.]
```

TensorFlow 2.0 Examples

```
import tensorflow as tf

W = tf.Variable([2.])
b = tf.Variable([1.])
x = tf.constant([1., -1.])

print("W = ", W.numpy())
print("b = ", b.numpy())

for a in x:
    h = tf.nn.relu(W*x + b)
    print('x = ', x.numpy(), 'h = ',
h.numpy())
```

Results :

W = [2.]
b = [1.]
x = [1. -1.] h = [3. 0.]

Two Computation Sections (TF 1.x)

1. Key concepts for building the graph
   - Tensor
   - TensorFlow data type : constant, variable, placeholder
   - Operations

2. Key concepts for running the graph
   - Session
   - Feeds and Fetches

Tensor

- Tensor is a n-dimensional numpy array or list to represent all data
- tf.Tensor object represents a tensor

Tensors

```
In [3]: 3 # a rank 0 tensor; this is a scalar with shape []
        [1. ,2., 3.] # a rank 1 tensor; this is a vector with shape [3]
        [[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
        [[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]

Out[3]: [[[1.0, 2.0, 3.0]], [[7.0, 8.0, 9.0]]]
```

```
t = tf.Constant([1., 2., 3.])
```

Tensor has 3 properties

- A rank (dimension) : the number of dimensions
- A shape : the number of elements in each dimension
- A data type

Tensor Ranks

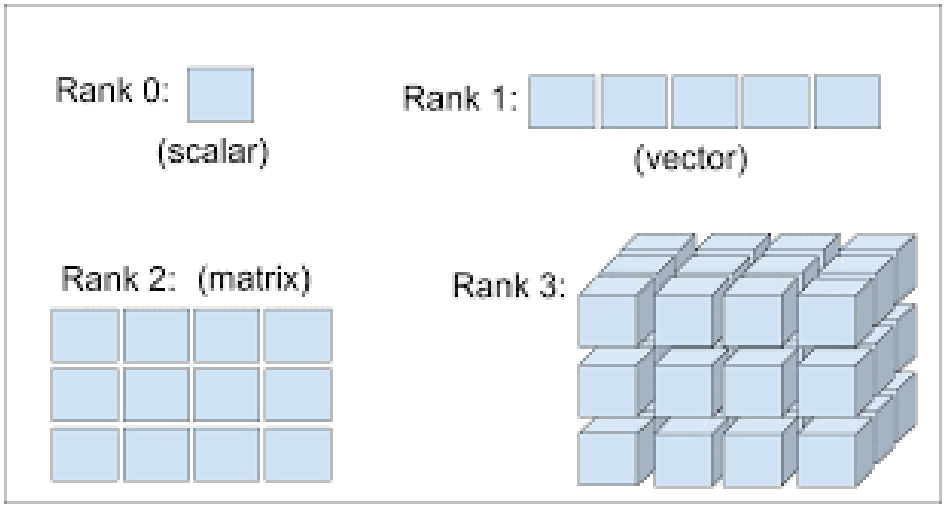
Rank	Math entity	Python example
0	Scalar (magnitude only)	`s = 483`
1	Vector (magnitude and direction)	`v = [1.1, 2.2, 3.3]`
2	Matrix (table of numbers)	`m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`
3	3-Tensor (cube of numbers)	`t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]`
n	n-Tensor (you get the idea)	`....`

```
>>x=tf.const([[1,2], [3,4]])
>>tf.rank(x)
```

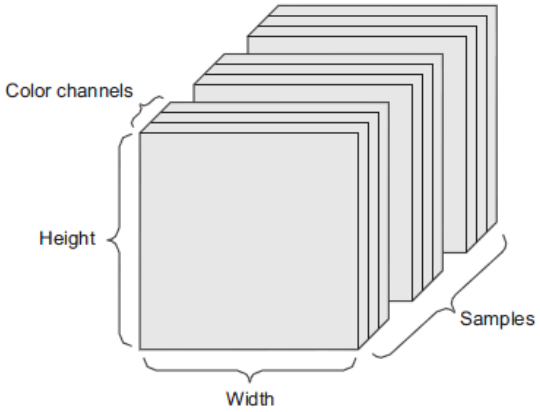
https://www.tensorflow.org/guide/tensors

Tensor

- Rank examples



Rank 4:



https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781787125933/14/ch14lvl1sec85/tensorflow-ranks-and-tensors

Tensor Ranks and Shapes

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

```
>>x=tf.const([[1,2], [3,4]])
>>tf.shape(x) or tf.get_shape(x)
```

https://www.tensorflow.org/guide/tensors

Real-world examples of data tensors

- Vector data - 2D tensors of shape (samples, features)
- Timeseries data or sequence data - 3D tensors of shape (samples, timesteps, features)
- Images - 4D tensors of shape (samples, height, width, channels) or (samples, channels, height, width)
- Video - 5D tensors of shape (samples, frames, height, width, channels) or (samples, frames, channels, height, width)

Tensor data types

- Tensors have a data type

- tf.float16: 16-bit half-precision floating-point.
- tf.float32: 32-bit single-precision floating-point.
- tf.float64: 64-bit double-precision floating-point.
- tf.bfloat16: 16-bit truncated floating-point.
- tf.complex64: 64-bit single-precision complex.
- tf.complex128: 128-bit double-precision complex.
- tf.int8: 8-bit signed integer.
- tf.uint8: 8-bit unsigned integer.
- tf.uint16: 16-bit unsigned integer.
- tf.uint32: 32-bit unsigned integer.
- tf.uint64: 64-bit unsigned integer.
- tf.int16: 16-bit signed integer.
- tf.int32: 32-bit signed integer.
- tf.int64: 64-bit signed integer.

- tf.bool: Boolean.
- tf.string: String.
- tf.qint8: Quantized 8-bit signed integer.
- tf.quint8: Quantized 8-bit unsigned integer.
- tf.qint16: Quantized 16-bit signed integer.
- tf.quint16: Quantized 16-bit unsigned integer.
- tf.qint32: Quantized 32-bit signed integer.
- tf.resource: Handle to a mutable resource.
- tf.variant: Values of aritrary types.

```
>>x=tf.const([1,2], [3,4])
>>x.dtype
```

https://www.tensorflow.org/guide/tensors

Numpy and Tensor

- Tensor → Numpy
  - x.numpy()
  - np.array(x)
  - numpy operation

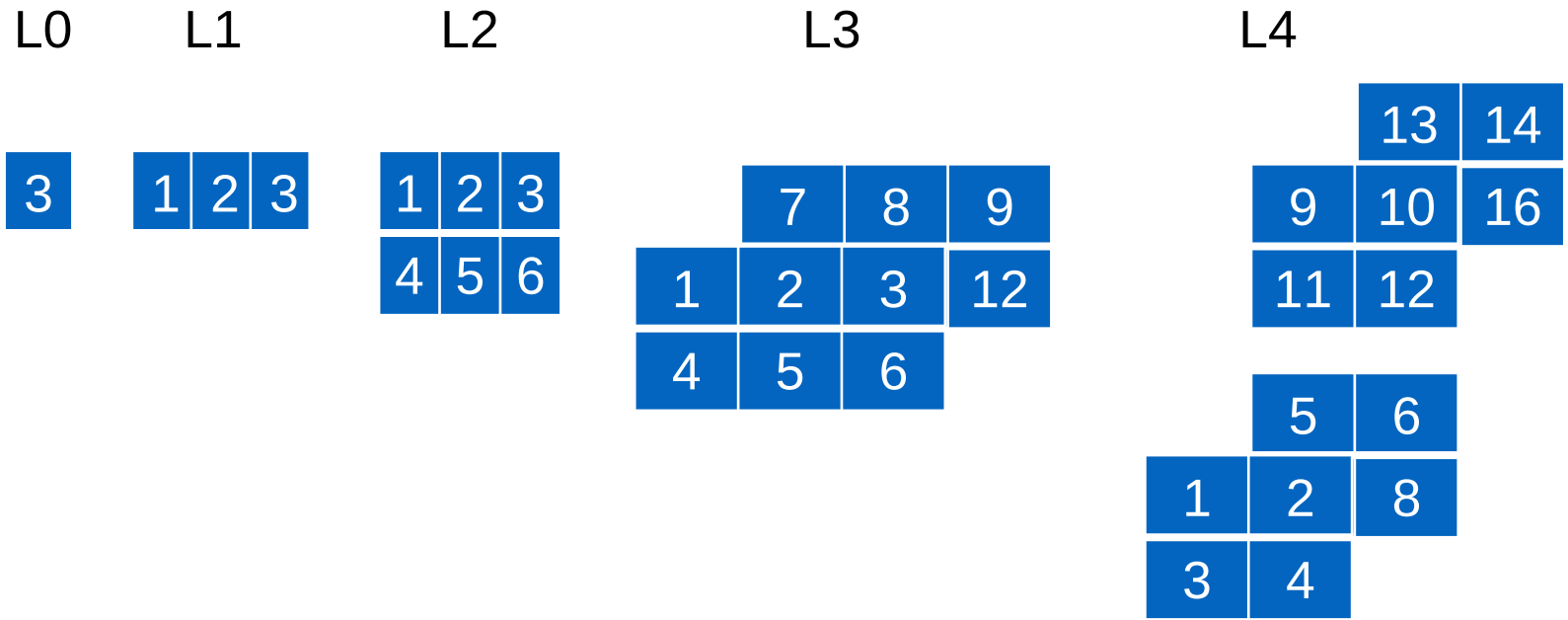
- Numpy → Tensorflow
  - tf.constant(x), tf.Variable(x)
  - tensorflow operation

https://www.tensorflow.org/guide/tensors

Exercise 01-2.

tf2-01-2-tensor_rank_shape.py

Exercise 01-2. : Tensors



Some Types of Special Tensor

- Constant : tf.constant
  - create a constant tensor
- Variables : tf.Variable
  - stateful nodes which output their current value
- Placeholder : tf.placeholder
  - nodes whose value is fed in at execution time

Constant

- Constant node takes zero tensors as inputs and produces a tensor as an output.
- Function

  tf.constant(value, dtype=None, shape=None, name='Const', verify_shape=False)

- Examples

```
tensor = tf.constant([1, 2, 3, 4, 5, 6, 7]) => [1 2 3 4 5 6 7]
# Constant 1-D Tensor populated with value list

tensor = tf.constant(-1.0, shape=[2, 3]) => [[-1. -1. -1.], [-1. -1. -1.]]
# Constant 2-D tensor populated with scalar value -1.
```

Exercise 01-3.

tf2-01-3-tf_constant.py

b’String’ ‘**b**’ indicates *Bytes literals*. http://stackoverflow.com/questions/6269765/

Variables

- Variables are tensors that represent shared, persistent state manipulated by program
- Variables can be saved (or loaded from) to disk during and after training
- Variables are parameters (weights W and bias b) in a neural network

- class tf.Variable
  - Constructor: an initial value for the variable, which can be a tensor of any type and shape
  - After construction, the type and shape are fixed

Exercise 01-4.

tf2-01-4-tf_variables.py

Tensorflow APIs

- Low Level APIs
  - Tensors and Operations
  - Variables
  - Graphs and Sessions
  - Save and Restore

- High Level APIs
  - Keras
  - Eager Execution vs. Graph Execution (TF 1.x)
  - Importing Data
  - Estimators

High Level APIs

- Keras : TensorFlow's high-level API for building and training deep learning models
- Eager Execution : an API for writing TensorFlow code imperatively
- Importing Data : easy input pipelines to bring your data into your TensorFlow program
- Estimators : a high-level API that provides fully-packaged models ready for large-scale training and production.

Eager Execution

- An imperative interface to TF
- Fast debugging & immediate run-time errors
- TF 2 requires Python 3.x (not Python 2.x)
- No static graphs or sessions
- Integration with Python tools
- Supports dynamic models + Python control flow
- Support for custom and higher-order gradients
- Default mode in Tensorflow 2.0

Eager Execution

- Default mode in TF 2.0
  >> print(tf.executing_eagerly())
  >> True

- Eager execution works nicely with NumPy

- Dynamic control flow

- Computing gradients
  - tape= tf.GradientTape() : to trace operations for computing gradients later
  - tape.gradient(loss, w) : to calculate gradients

https://www.tensorflow.org/guide/eager

Exercise 01-5.

tf2-01-5-tf_eager_ex.py

Importing Data

- tf.data
  - API to build complex input pipelines from simple, reusable pieces
  - API to deal with large amounts of data, different data formats

- tf.data.Dataset
  - to represents a sequence of one or more Tensor objects

- tf.data.Iterator
  - to provide the main way to extract elements from a dataset

https://www.tensorflow.org/guide/datasets