

# Deep Learning

Vazgen Mikayelyan

October 31, 2020



## 1 Introduction to Tensorflow

## Difference Between



**CPU**



**GPU**



**TPU**

# CPU vs GPU vs TPU

# CPU vs GPU vs TPU

- Central Processing Unit is the electronic circuitry, which work as a brain of the computer that perform the basic arithmetic, logical, control and input/output operations specified by the instructions of a computer program.

# CPU vs GPU vs TPU

- Central Processing Unit is the electronic circuitry, which work as a brain of the computer that perform the basic arithmetic, logical, control and input/output operations specified by the instructions of a computer program.
- The Graphics Processing Unit is a specialized electronic circuit designed to render 2D and 3D graphics together with a CPU. GPU also known as Graphics Card in the Gammer's culture. Now GPU are being harnessed more broadly to accelerate computational workloads in areas such as financial modeling, cutting-edge scientific research, deep learning, analytics and oil and gas exploration etc.

# CPU vs GPU vs TPU

- Central Processing Unit is the electronic circuitry, which work as a brain of the computer that perform the basic arithmetic, logical, control and input/output operations specified by the instructions of a computer program.
- The Graphics Processing Unit is a specialized electronic circuit designed to render 2D and 3D graphics together with a CPU. GPU also known as Graphics Card in the Gammer's culture. Now GPU are being harnessed more broadly to accelerate computational workloads in areas such as financial modeling, cutting-edge scientific research, deep learning, analytics and oil and gas exploration etc.
- Tensor Processing Unit is a custom-built integrated circuit developed specifically for machine learning and tailored for TensorFlow, Google's open-source machine learning framework. TPU's have been powering Google data centers since 2015.

# Why Tensorflow?



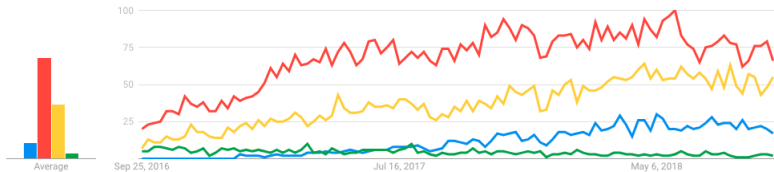


# Interest Over Time

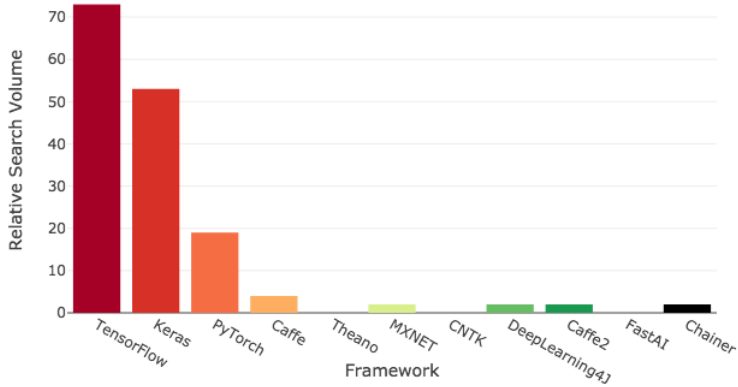
<span style="color: blue;">●</span> <b>pytorch</b> Search term	<span style="color: red;">●</span> <b>tensorflow</b> Search term	<span style="color: orange;">●</span> <b>keras</b> Search term	<span style="color: green;">●</span> <b>caffe</b> Search term	+
---	---	---	--	---

Worldwide ▾ 9/20/16 - 9/21/18 ▾ Machine Learning & Arti... ▾ Web Search ▾

Interest over time ?



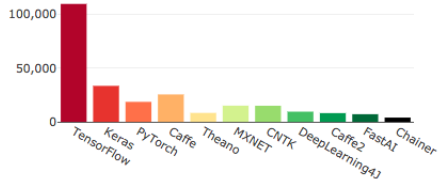
## Google Search Volume



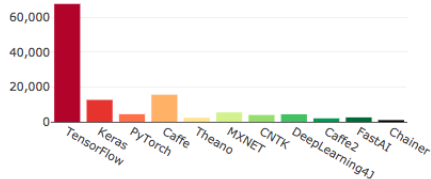
# GitHub Activity

GitHub Activity

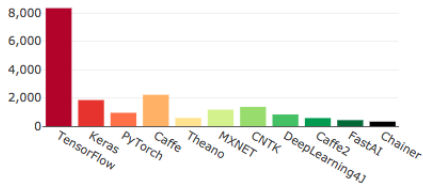
Stars



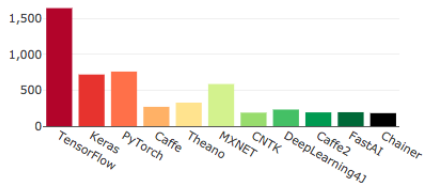
Forks



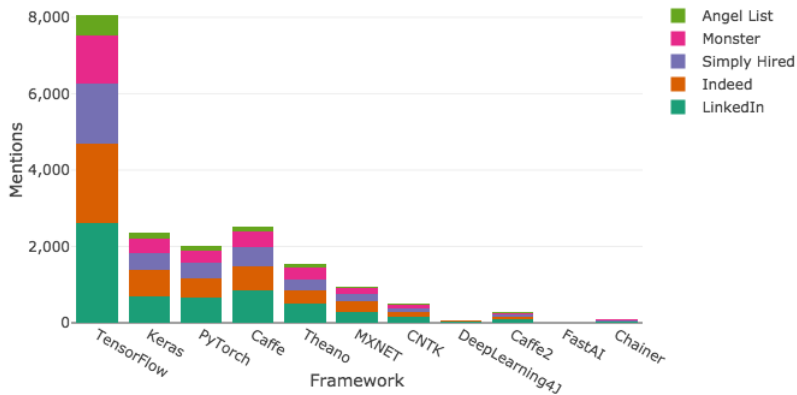
Watchers



Contributors



## Online Job Listings



# What can do Tensorflow?

- TensorFlow lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device in a very simple way. This way the things can be done very fast.

# What can do Tensorflow?

- TensorFlow lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device in a very simple way. This way the things can be done very fast.
- TensorFlow lets you express your computation as a data flow graph.

# What can do Tensorflow?

- TensorFlow lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device in a very simple way. This way the things can be done very fast.
- TensorFlow lets you express your computation as a data flow graph.
- TensorFlow lets you visualize the graph using the in-built tensorboard. You can inspect and debug the graph very easily.

# What can do Tensorflow?

- TensorFlow lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device in a very simple way. This way the things can be done very fast.
- TensorFlow lets you express your computation as a data flow graph.
- TensorFlow lets you visualize the graph using the in-built tensorboard. You can inspect and debug the graph very easily.
- TensorFlow gives the best performance with an ability to iterate quickly, train models faster and run more experiments.



# What can do Tensorflow?

- TensorFlow lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device in a very simple way. This way the things can be done very fast.
- TensorFlow lets you express your computation as a data flow graph.
- TensorFlow lets you visualize the graph using the in-built tensorboard. You can inspect and debug the graph very easily.
- TensorFlow gives the best performance with an ability to iterate quickly, train models faster and run more experiments.
- TensorFlow runs on nearly everything: GPUs and CPUs—including mobile and embedded platforms—and tensor processing units (TPUs), which are specialized hardware to do the tensor math on.

- View functions as computational graphs.

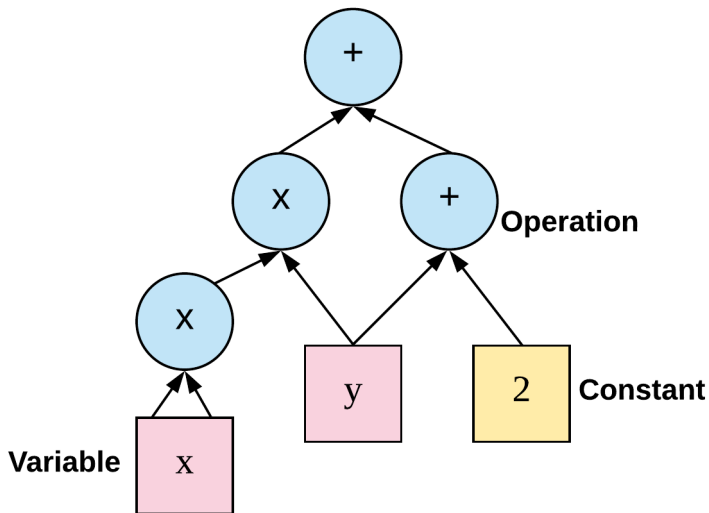
# Basic Code Structure

- View functions as computational graphs.
- First build a computational graph, and then use a session to execute operations in the graph.

# Basic Code Structure

- View functions as computational graphs.
- First build a computational graph, and then use a session to execute operations in the graph.
- This is the basic approach, there is also a dynamic approach implemented in the recently introduced eager mode.

# Computational Graphs



# Computational Graphs

- Nodes are operators (ops), variables and constants.

# Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.

# Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- Constants are fixed value tensors - not trainable.



# Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- Constants are fixed value tensors - not trainable.
- Variables are tensors initialized in a session - trainable / not trainable.

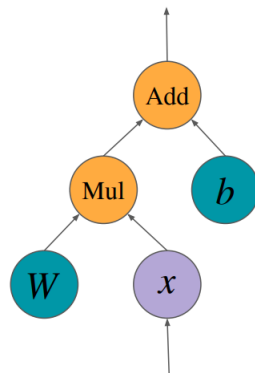
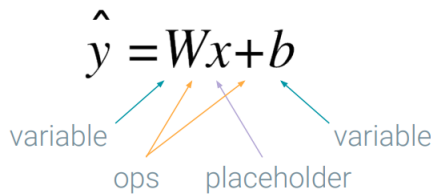
# Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- Constants are fixed value tensors - not trainable.
- Variables are tensors initialized in a session - trainable / not trainable.
- Placeholders are tensors of values that are unknown during the graph construction, but passed as input during a session.

# Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- Constants are fixed value tensors - not trainable.
- Variables are tensors initialized in a session - trainable / not trainable.
- Placeholders are tensors of values that are unknown during the graph construction, but passed as input during a session.
- Ops are functions on tensors.

# Computational Graphs



- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> c = tf.add(a, b)
>>> with tf.Session() as sess:
...     print(sess.run(a))
...     print(sess.run(b))
...     print(sess.run(c))
...
1
2
3
```

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> c = tf.add(a, b)
>>> with tf.Session() as sess:
...     print(sess.run(a))
...     print(sess.run(b))
...     print(sess.run(c))
...
1
2
3
```

- `a.eval()` is equivalent to `session.run(a)`, but in general, “eval” is limited to executions of a single op and ops that returns a value.



- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> c = tf.add(a, b)
>>> with tf.Session() as sess:
...     print(sess.run(a))
...     print(sess.run(b))
...     print(sess.run(c))
...
1
2
3
```

- `a.eval()` is equivalent to `session.run(a)`, but in general, “eval” is limited to executions of a single op and ops that returns a value.
- Upon op execution, only the subgraph (required for calculating its value) is evaluated

1. Assembling the graph
  - Create placeholders.

## 1. Assembling the graph

- Create placeholders.
- Create variables / neural network.

## 1. Assembling the graph

- Create placeholders.
- Create variables / neural network.
- Define a loss function.

## 1. Assembling the graph

- Create placeholders.
- Create variables / neural network.
- Define a loss function.
- Define an optimizer.

# Structure of Training Code

1. Assembling the graph
  - Create placeholders.
  - Create variables / neural network.
  - Define a loss function.
  - Define an optimizer.
2. Training in a session
  - Start a session.

## 1. Assembling the graph

- Create placeholders.
- Create variables / neural network.
- Define a loss function.
- Define an optimizer.

## 2. Training in a session

- Start a session.
- Initialize variables / restore from checkpoint.

## 1. Assembling the graph

- Create placeholders.
- Create variables / neural network.
- Define a loss function.
- Define an optimizer.

## 2. Training in a session

- Start a session.
- Initialize variables / restore from checkpoint.
- Run the optimizer over batches.



TensorBoard

SCALARS

GRAPHS

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.942

Horizontal Axis

STEP

RELATIVE

WALL

Runs

Write a regex to filter runs

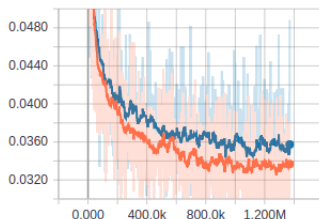
train

validation

🔍 Filter tags (regular expressions supported)

Loss

Loss



Loss1