Deep Learning

Vazgen Mikayelyan

YSU, Krisp

September 27, 2019

Outline

Stochastic Gradient Descent

Let L be a loss function that we know:

Stochastic Gradient Descent

Let L be a loss function that we know:

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \left(f_w(x_i) - y_i\right)^2,$$

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \left(-y_l \log f_w(x_i) - (1 - y_i) \log\left(1 - f_w(x_i)\right)\right),$$

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \left(-y_i^T \log f_w(x_i)\right).$$

Stochastic Gradient Descent

Let L be a loss function that we know:

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} (f_w(x_i) - y_i)^2,$$

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} (-y_l \log f_w(x_i) - (1 - y_i) \log(1 - f_w(x_i))),$$

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \left(-y_i^T \log f_w(x_i)\right).$$

Do you see problems in finding minimum of these functions using GD?

Stochastic Gradient Descent

Note that in each case we can represent the loss function by the following form:

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} L_i(w).$$

Stochastic Gradient Descent

Note that in each case we can represent the loss function by the following form:

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} L_i(w).$$

SGD algorithm is the following:

- Choose an initial vector of parameters w and learning rate α.

Stochastic Gradient Descent

Note that in each case we can represent the loss function by the following form:

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} L_i(w).$$

SGD algorithm is the following:

- Choose an initial vector of parameters w and learning rate α.
- Repeat until an approximate minimum is obtained.

Stochastic Gradient Descent

Note that in each case we can represent the loss function by the following form:

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} L_i(w).$$

SGD algorithm is the following:

- Choose an initial vector of parameters w and learning rate α.
- Repeat until an approximate minimum is obtained.
    - Randomly shuffle examples in the training set.

Stochastic Gradient Descent

Note that in each case we can represent the loss function by the following form:

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} L_i(w).$$

SGD algorithm is the following:

- Choose an initial vector of parameters w and learning rate α.
- Repeat until an approximate minimum is obtained.
    - Randomly shuffle examples in the training set.
    - For $i = 1, 2, ..., n$, do $w \leftarrow w - \alpha L_i(w)$.

Stochastic Gradient Descent

Note that in each case we can represent the loss function by the following form:

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} L_i(w).$$

SGD algorithm is the following:

- Choose an initial vector of parameters w and learning rate α.
- Repeat until an approximate minimum is obtained.
  - Randomly shuffle examples in the training set.
  - For $i = 1, 2, ..., n$, do $w \leftarrow w - \alpha L_i(w)$.

Do you see problems in this optimization method?

Mini-Batch Gradient Descent

MBGD algorithm is the following:

- Choose an initial vector of parameters w, learning rate α and batch size B.

Mini-Batch Gradient Descent

MBGD algorithm is the following:

- Choose an initial vector of parameters w, learning rate α and batch size B.
- Repeat until an approximate minimum is obtained.

Mini-Batch Gradient Descent

MBGD algorithm is the following:

- Choose an initial vector of parameters w, learning rate α and batch size B.
- Repeat until an approximate minimum is obtained.
  - Randomly shuffle examples in the training set.

Mini-Batch Gradient Descent

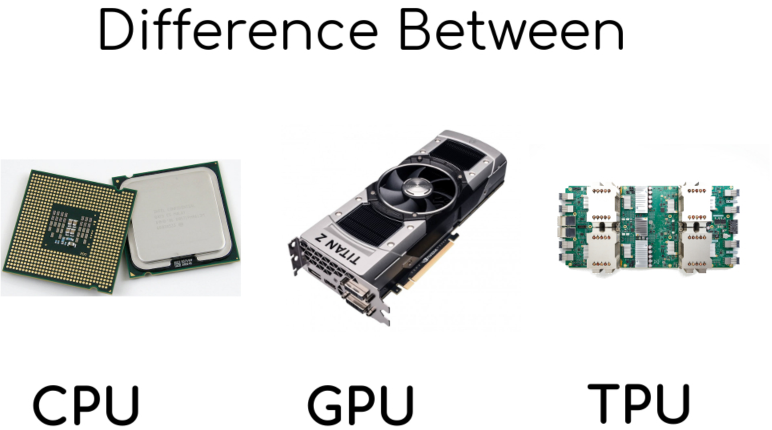
MBGD algorithm is the following:

- Choose an initial vector of parameters w, learning rate α and batch size B.
- Repeat until an approximate minimum is obtained.
  - Randomly shuffle examples in the training set.
  - For $i = 1, 2, ..., \lceil \frac{n}{B} \rceil$, do

$$w \leftarrow w - \alpha \frac{1}{B} \sum_{k=(i-1)\cdot B+1}^{i \cdot B} L_k(w).$$

Outline

CPU vs GPU vs 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

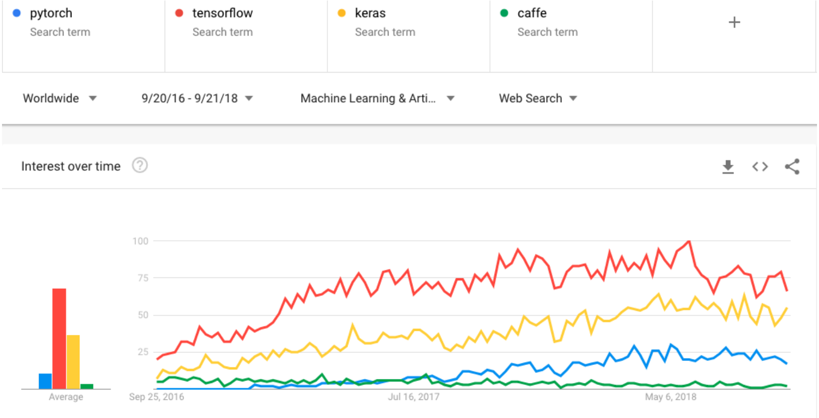
- Central Processing Unit is the electronic circuitry, which work as a brain of the computer that perform the basic arithmetic, logical, control and input/output operations specified by the instructions of a computer program.
- The Graphics Processing Unit is a specialized electronic circuit designed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

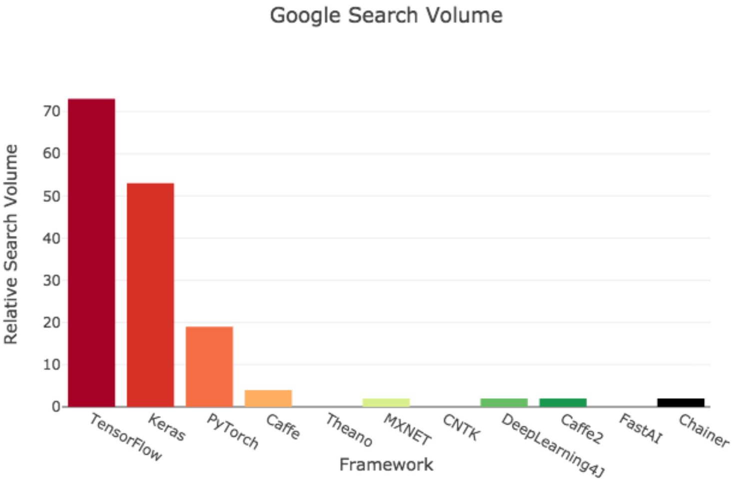
- Central Processing Unit is the electronic circuitry, which work as a brain of the computer that perform the basic arithmetic, logical, control and input/output operations specified by the instructions of a computer program.
- The Graphics Processing Unit is a specialized electronic circuit designed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

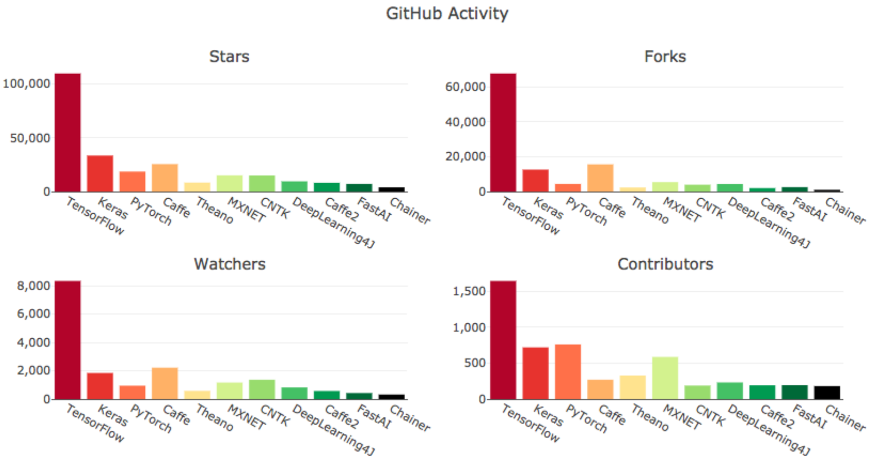
Interest Over Time



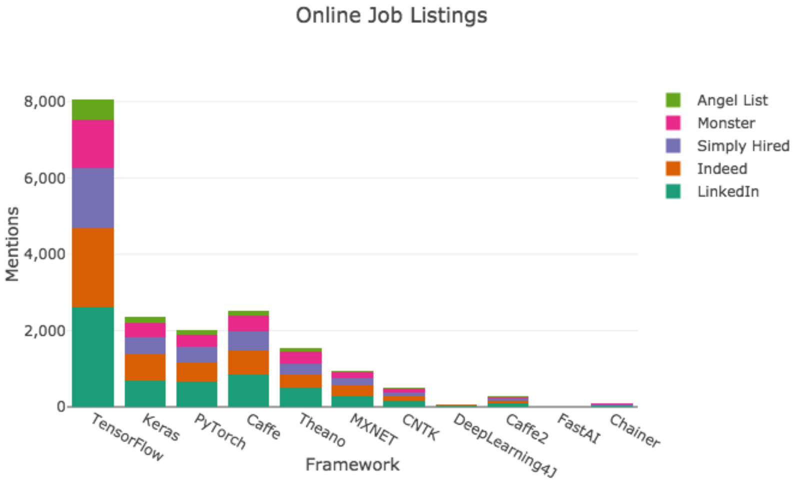
Google Search Activity



GitHub Activity



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.

Basic Code Structure

- View functions as computational graphs.

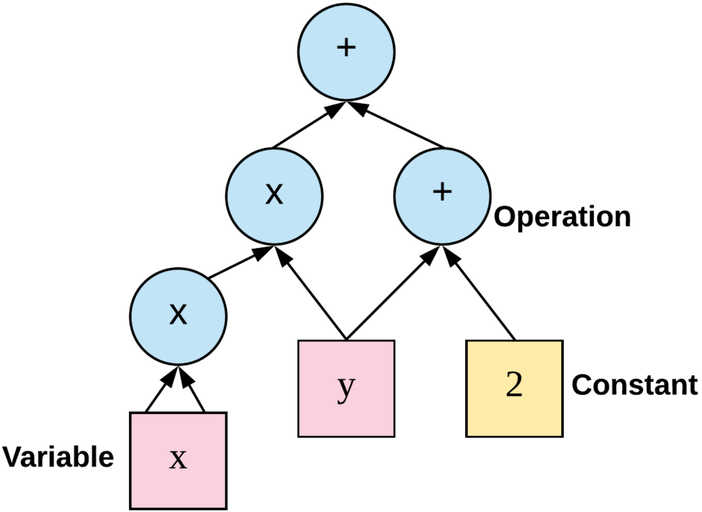
Basic Code Structure

- 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.

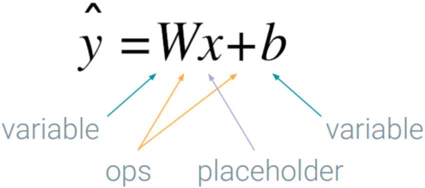
Computational Graphs

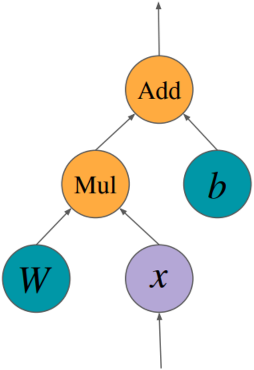
- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- Constants are fixed value tensors - not trainable.
- Variables are tensors initialized in a session - trainable / not trainable.
- Placeholders are tensors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





Sessions

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

Sessions

- 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)
```

Sessions

- 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
```

Sessions

- 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.

Sessions

- 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

Structure of Training Code

1. Assembling the graph
    - Create placeholders.

Structure of Training Code

1. Assembling the graph
    - Create placeholders.
    - Create variables / neural network.

Structure of Training Code

1. Assembling the graph
    - Create placeholders.
    - Create variables / neural network.
    - Define a loss function.

Structure of Training Code

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.

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.
   - Initialize variables / restore from checkpoint.

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

Tensorboard