



# DEEP LEARNING FRAMEWORKS

---

**Olivier Moindrot**

Data scientist - OWKIN

DEEP LEARNING PRACTICAL COURSE  
ECOLE POLYTECHNIQUE, 03/05/2018

# Program & Course Logistics

---

- Course 1 : (05-04-18)
  - Introduction to Deep Learning - Mouhidine SEIV (Riminder)
- Course 2 : (12-04-18)
  - Deep Learning in Computer Vision - Slim FRIKHA (Riminder)
- Course 3 : (19-04-18)
  - Deep Learning in NLP - Paul COURSAUX (Riminder)
- Course 4 : (03-05-18)
  - Introduction to Deep Learning Frameworks - Olivier Moindrot (Owkin)
- Course 5: (10-05-18)
  - Deployment in Production and Parallel Computing - INVITED GUEST
- Course 6: (17-05-18)
  - Efficient Methods and Compression for Deep Learning - Antoine Biard

**Location: Ecole Polytechnique from 6:30 pm to 7:30pm**



<https://github.com/riminder>

# Talk outline

---

- I. CPU vs GPU (vs TPU)*
- II. Frameworks overview: TensorFlow, PyTorch, Keras...*
- III. Focus on TensorFlow*
- IV. Best practices for high-quality code*

Link to these slides (with speaker notes):

<https://tinyurl.com/deepframeworks>

# CPU vs GPU

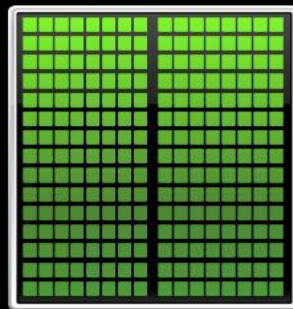
---

# What is a GPU

*The Difference between a CPU and GPU*



CPU



GPU

# CPU, GPU

	# Cores	Clock Speed	Memory	Price	Speed
<b>CPU</b> (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	\$339	~540 GFLOPs FP32
<b>GPU</b> (NVIDIA Titan Xp)	3584	1.6 GHz	11 GB GDDR5 X	\$699	~11.4 TFLOPs FP32

**CPU:** Fewer cores, but each core is much faster and much more capable; great at sequential tasks

**GPU:** More cores, but each core is much slower and “dumber”; great for parallel tasks

# CPU, GPU, TPU

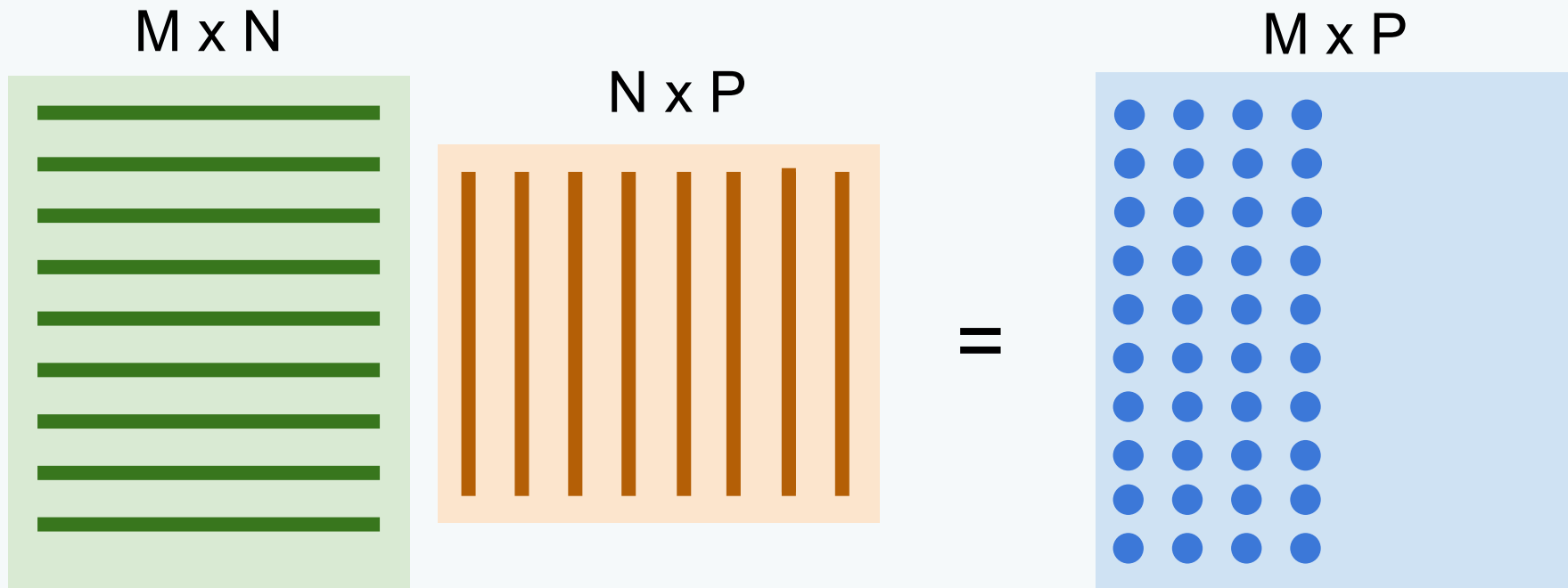
	# Cores	Clock Speed	Memory	Price	Speed
<b>CPU</b> (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	\$339	~540 GFLOPs FP32
<b>GPU</b> (NVIDIA Titan Xp)	3584	1.6 GHz	11 GB GDDR5 X	\$699	~11.4 TFLOPs FP32
<b>TPU</b> NVIDIA TITAN V	5120 CUDA, 640 Tensor	1.5 GHz	12 GB HBM2	\$2999	~14 TFLOPs FP32 ~112 TFLOPs FP16
<b>TPU</b> Google Cloud TPU	?	?	64 GB HBM	\$6.50 per hour	~180 TFLOPs

**CPU:** Fewer cores, but each core is much faster and much more capable; great at sequential tasks

**GPU:** More cores, but each core is much slower and “dumber”; great for parallel tasks

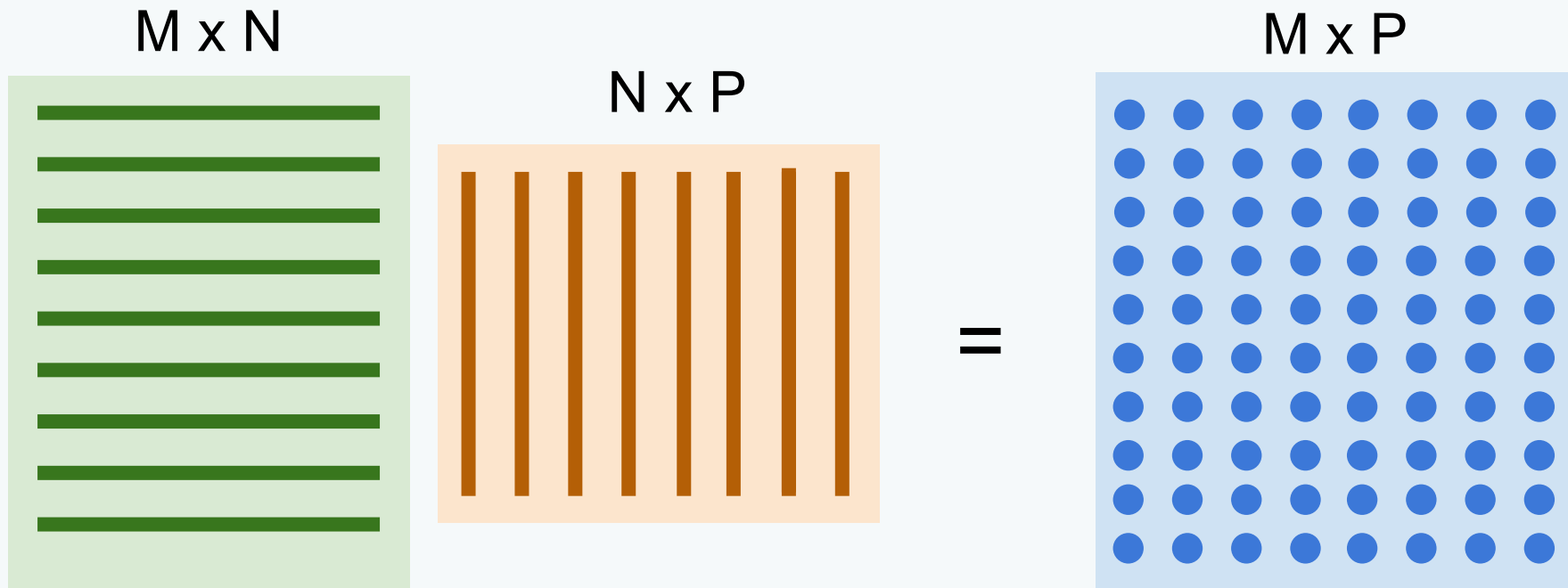
**TPU:** Specialized hardware for deep learning

# Example: matrix multiplication

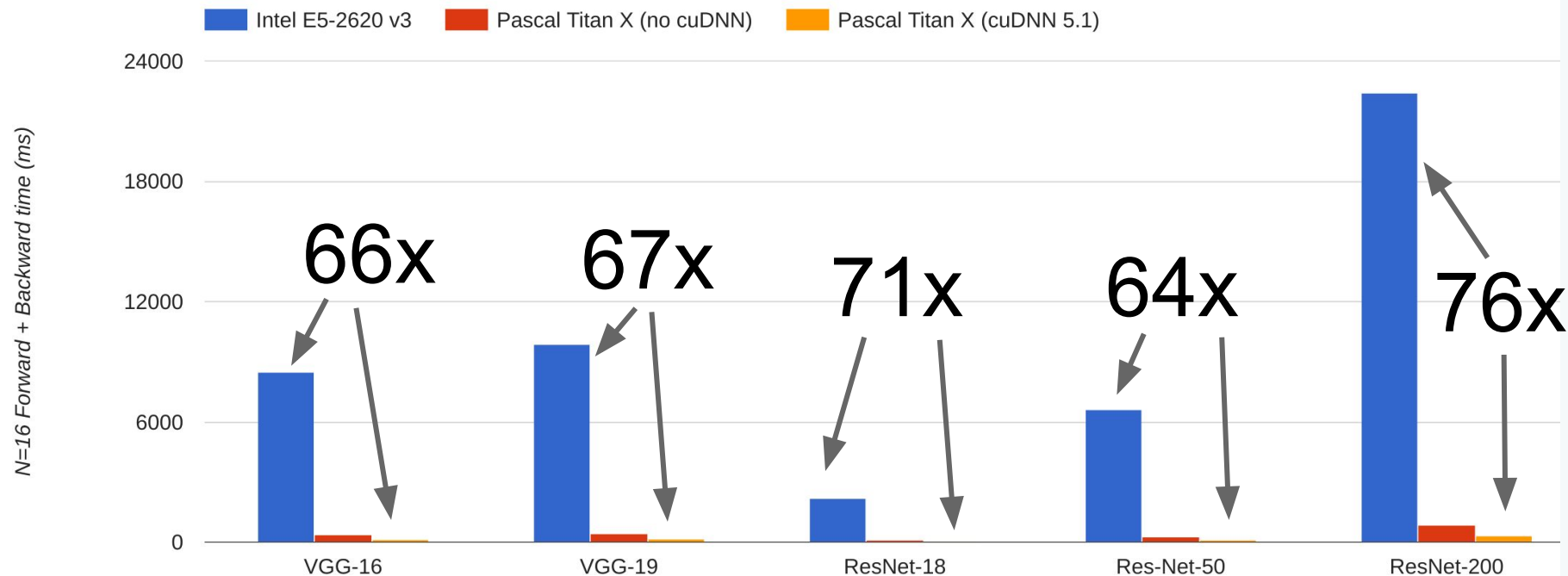




# Example: matrix multiplication



# Benchmark: CPU vs GPU

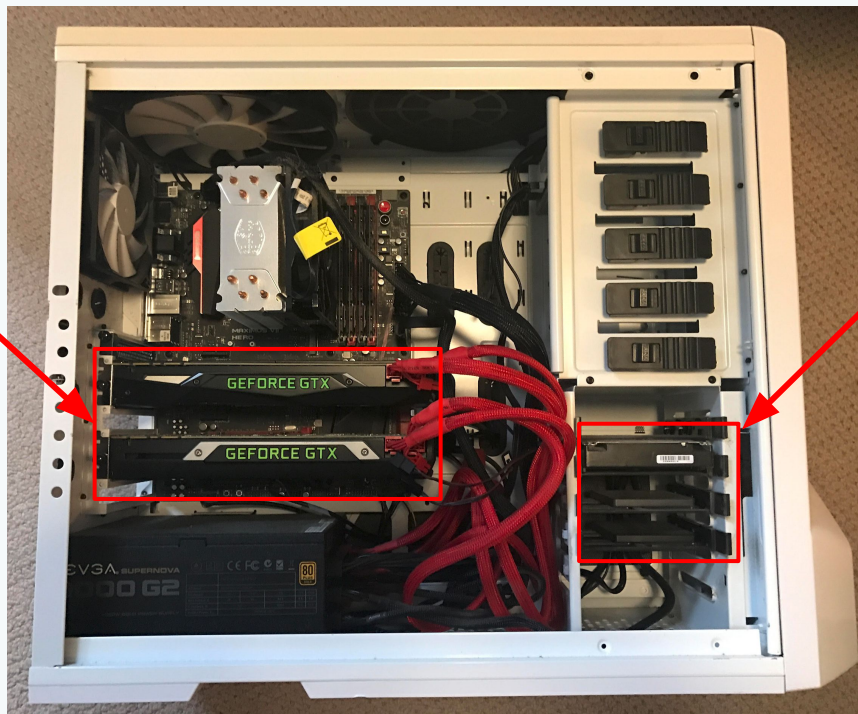


Data from <https://github.com/cjohnson/cnn-benchmarks>

(CPU performance not well-optimized, a little unfair)

# CPU / GPU communication

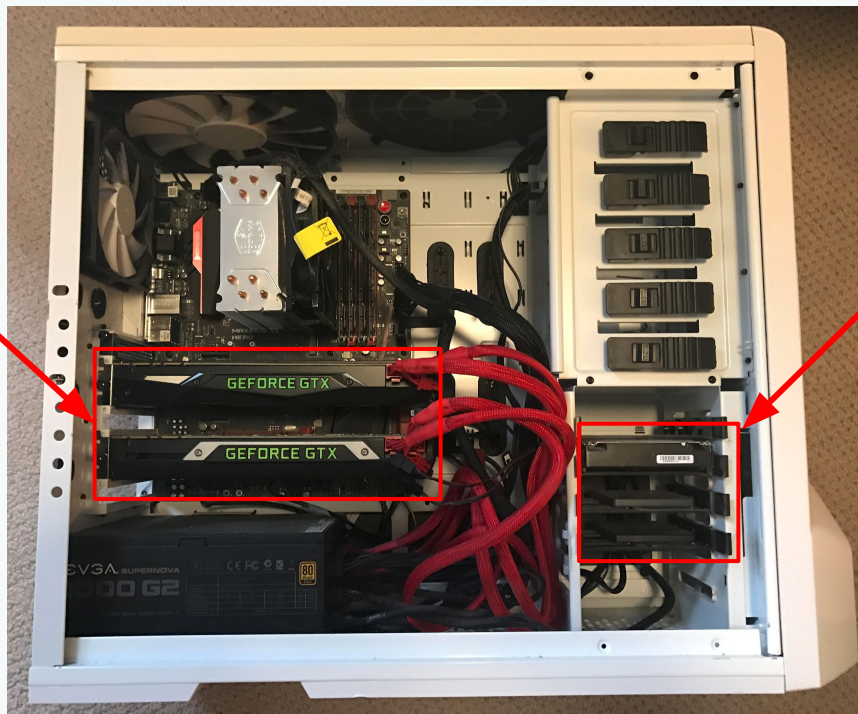
Model  
is here



Data is here

# CPU / GPU communication

Model  
is here



Data is here

If you aren't careful, training can bottleneck on reading data and transferring to GPU!

## Solutions:

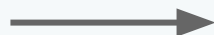
- Read all data into RAM
- Use SSD instead of HDD
- Use multiple CPU threads to prefetch data

# Deep Learning Frameworks

---

# Deep Learning Frameworks: Overview

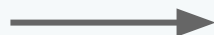
Caffe  
(UC Berkeley)



Caffe2  
(Facebook)

PaddlePaddle  
(Baidu)

Torch  
(NYU / Facebook)



Torch  
(NYU / Facebook)

Core ML  
(Apple)

Theano  
(U Montreal)



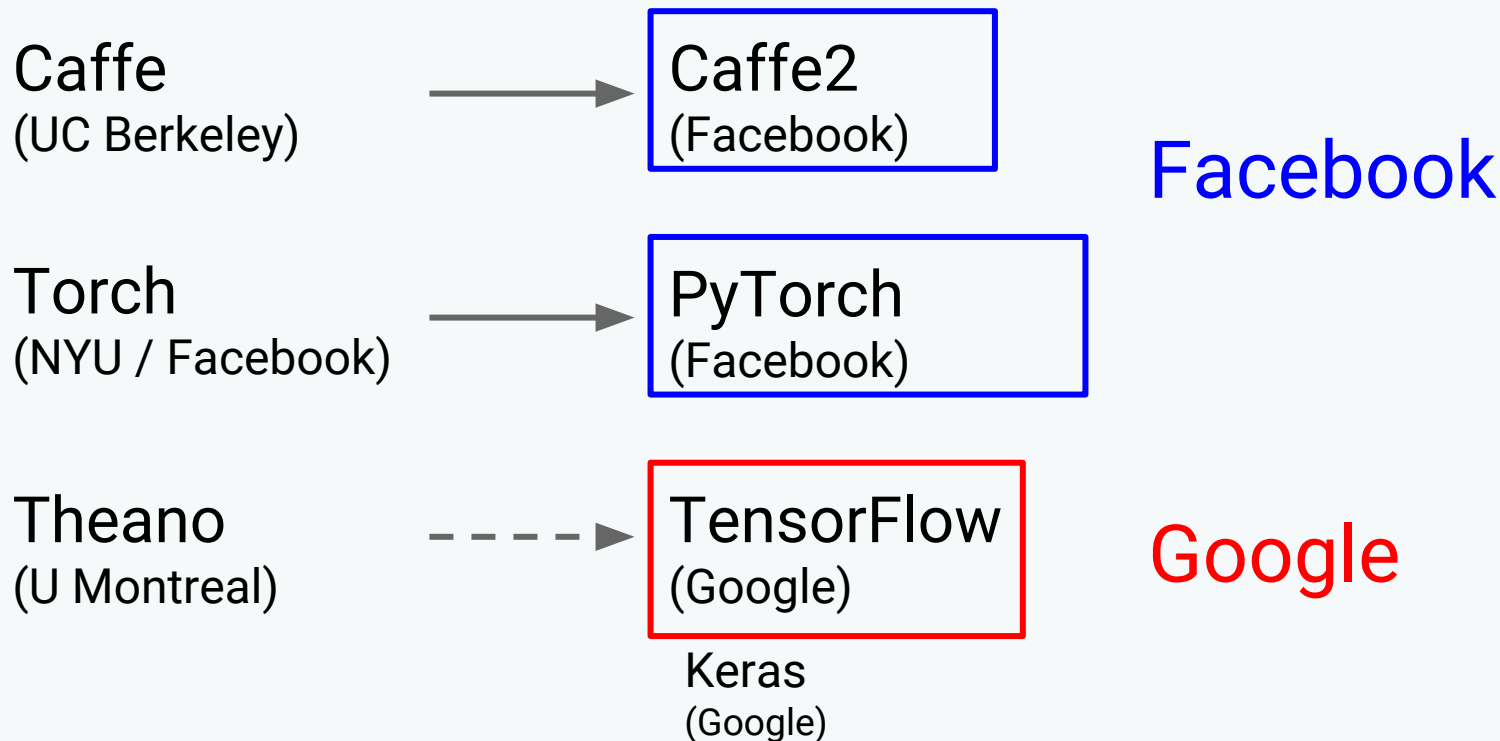
TensorFlow  
(Google)

CNTK  
(Microsoft)

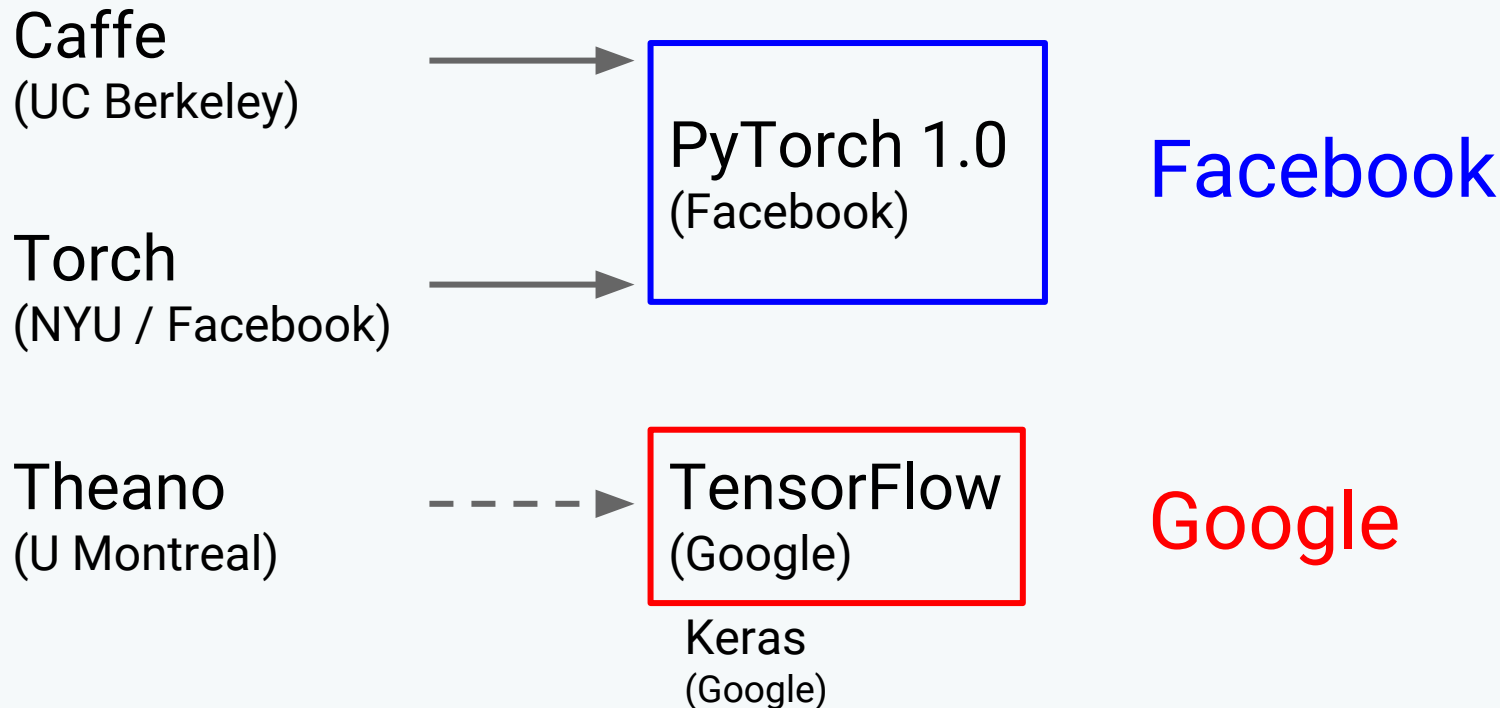
Keras  
(Google)

MXNet  
(Amazon)

# Deep Learning Frameworks: Overview



# Deep Learning Frameworks: Overview

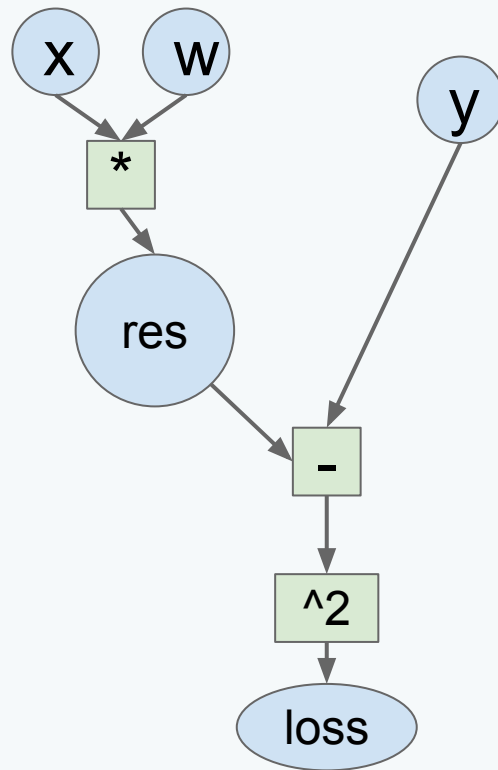




# What is a deep learning framework

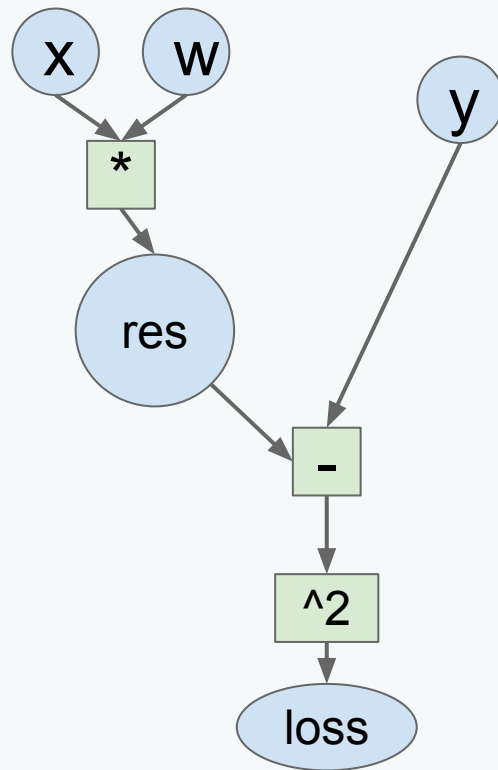
Computational  
graph

$$\text{loss} = (W^T x - y)^2$$



# Computational graph

Numpy



# Computational graph

## Numpy

```
import numpy as np

N = 3
D = 2

x = np.random.randn(N, D)
y = np.random.randn(N, 1)
w = np.random.randn(D, 1)

res = np.matmul(x, w)
loss = 0.5 * np.sum(np.square(res - y))

grad_w = x.T.dot(res - y)
```

### Issues:

- Need to compute gradients yourself
- Doesn't run on GPU

# Computational graph

## Numpy

```
import numpy as np

N = 3
D = 2

x = np.random.randn(N, D)
y = np.random.randn(N, 1)
w = np.random.randn(D, 1)

res = np.matmul(x, w)
loss = 0.5 * np.sum(np.square(res - y))

grad_w = x.T.dot(res - y)
```

## PyTorch

```
import torch

N = 3
D = 2

x = torch.randn(N, D)
y = torch.randn(N, 1)
w = torch.randn(D, 1)

res = torch.matmul(x, w)
loss = 0.5 * torch.sum(torch.pow(res - y, 2))
```

Looks like numpy !

# Computational graph

Free gradients !

Numpy

PyTorch

```
import numpy as np

N = 3
D = 2

x = np.random.randn(N, D)
y = np.random.randn(N, 1)
w = np.random.randn(D, 1)

res = np.matmul(x, w)
loss = 0.5 * np.sum(np.square(res - y))

grad_w = x.T.dot(res - y)
```

```
import torch

N = 3
D = 2

x = torch.randn(N, D)
y = torch.randn(N, 1)
w = torch.randn(D, 1, requires_grad=True)

res = torch.matmul(x, w)
loss = 0.5 * torch.sum(torch.pow(res - y, 2))

loss.backward()
print(w.grad)
```

# Computational graph

Trivial to make it  
work on GPU

Numpy

```
import numpy as np

N = 3
D = 2

x = np.random.randn(N, D)
y = np.random.randn(N, 1)
w = np.random.randn(D, 1)

res = np.matmul(x, w)
loss = 0.5 * np.sum(np.square(res - y))

grad_w = x.T.dot(res - y)
```

PyTorch

```
import torch

device = 'cuda:0'
N, D = 3, 2

x = torch.randn(N, D, device=device)
y = torch.randn(N, 1, device=device)
w = torch.randn(D, 1, requires_grad=True,
                 device=device)

res = torch.matmul(x, w)
loss = 0.5 * torch.sum(torch.pow(res - y, 2))

loss.backward()
print(w.grad)
```

# Why use a deep learning framework

- Easy to implement new neural networks
- Automatic gradient differentiation
- Seamless use of the GPU / TPU

# Focus on TensorFlow

---



# TensorFlow timeline

---

- Released in November 2015
  - **v1.0** in February 2017 (backward compatible)
  - Today we are at **v1.8**
- Moving very fast
  - New features come out fast
  - Huge team at Google working on TensorFlow

# Getting started

---

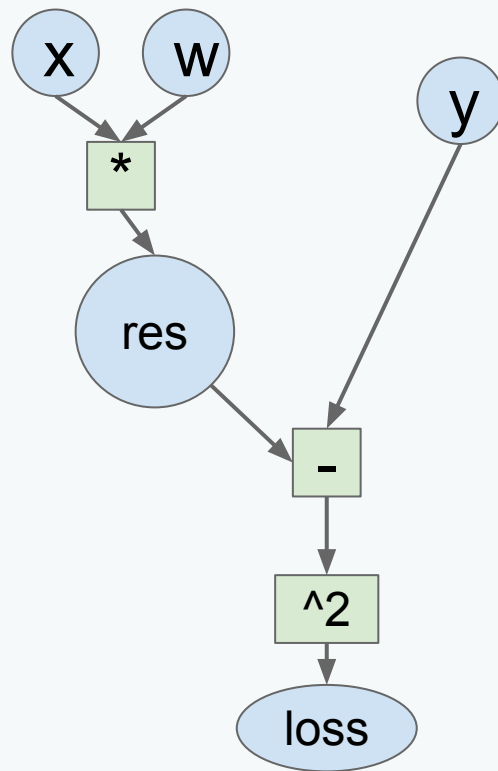
Read the official documentation:

[https://www.tensorflow.org/get\\_started/](https://www.tensorflow.org/get_started/)

# TensorFlow basics: graph and session

- Graph
  - Define the computational graph

$$loss = (W^T x - y)^2$$



# TensorFlow basics: graph and session

- **Graph**
  - Define the computational graph

```
import numpy as np
import tensorflow as tf

N, D = 3, 2

x = np.random.randn(N, D)
y = np.random.randn(N, 1)
w = np.random.randn(D, 1)

res = tf.matmul(x, w)
loss = 0.5 * tf.reduce_sum(tf.square(res - y))

sess = tf.Session()
print(sess.run(loss))
```

# TensorFlow basics: graph and session

- **Graph**
  - Define the computational graph
- **Session**
  - Execute operations of the graph
  - Ex: training operation

```
import numpy as np
import tensorflow as tf
```

```
N, D = 3, 2
```

```
x = np.random.randn(N, D)
```

```
y = np.random.randn(N, 1)
```

```
w = np.random.randn(D, 1)
```

```
res = tf.matmul(x, w)
```

```
loss = 0.5 * tf.reduce_sum(tf.square(res - y))
```

```
sess = tf.Session()
```

```
print(sess.run(loss))
```

# TensorFlow basics: placeholders

## Placeholders

- Way to input data into the graph
- Ex: feed an image to the model

```
N, D = 3, 2

x = tf.placeholder(tf.float32, [N, D])
y = tf.placeholder(tf.float32, [N, 1])
w = np.random.randn(D, 1).astype(np.float32)

res = tf.matmul(x, w)
loss = 0.5 * tf.reduce_sum(tf.square(res - y))

sess = tf.Session()
print(sess.run(loss))
```

**Error:** “You must feed a value for placeholder”

# TensorFlow basics: placeholders

## Placeholders

- Way to input data into the graph
- Ex: feed an image to the model

Need to feed the values of the placeholders

```
N, D = 3, 2
```

```
x = tf.placeholder(tf.float32, [N, D])
```

```
y = tf.placeholder(tf.float32, [N, 1])
```

```
w = np.random.randn(D, 1).astype(np.float32)
```

```
res = tf.matmul(x, w)
```

```
loss = 0.5 * tf.reduce_sum(tf.square(res - y))
```

```
sess = tf.Session()
```

```
feed_dict = {x: np.random.randn(N, D),  
              y: np.random.randn(N, 1)}
```

```
print(sess.run(loss, feed_dict=feed_dict))
```

# TensorFlow basics: variables

## Variables

- Its value can change over time
- Needs to be initialized

**Error:** “Attempting to use uninitialized value”

```
x = tf.placeholder(tf.float32, [N, D])
y = tf.placeholder(tf.float32, [N, 1])
w = np.random.randn(D, 1).astype(np.float32)
w = tf.Variable(w)

res = tf.matmul(x, w)
loss = 0.5 * tf.reduce_sum(tf.square(res - y))

sess = tf.Session()
feed_dict = {x: np.random.randn(N, D),
              y: np.random.randn(N, 1)}
print(sess.run(loss, feed_dict=feed_dict))
```



# TensorFlow basics: variables

## Variables

- Its value can change over time
- Needs to be initialized

```
x = tf.placeholder(tf.float32, [N, D])
y = tf.placeholder(tf.float32, [N, 1])
w = np.random.randn(D, 1).astype(np.float32)
w = tf.Variable(w)

res = tf.matmul(x, w)
loss = 0.5 * tf.reduce_sum(tf.square(res - y))

sess = tf.Session()
sess.run(w.initializer)
feed_dict = {x: np.random.randn(N, D),
              y: np.random.randn(N, 1)}
print(sess.run(loss, feed_dict=feed_dict))
```

# The data pipeline

- Create the input data pipeline (memory -> GPU)
- Library: **tf.data**
- Guides:
  - [https://www.tensorflow.org/get\\_started/datasets\\_quickstart](https://www.tensorflow.org/get_started/datasets_quickstart)
  - [https://www.tensorflow.org/programmers\\_guide/datasets](https://www.tensorflow.org/programmers_guide/datasets)
  - [https://www.tensorflow.org/performance/datasets\\_performance](https://www.tensorflow.org/performance/datasets_performance)



# Creating the model

- Easily define the model
- Module `tf.layers`
- Tutorials:
  - <https://www.tensorflow.org/tutorials/layers>

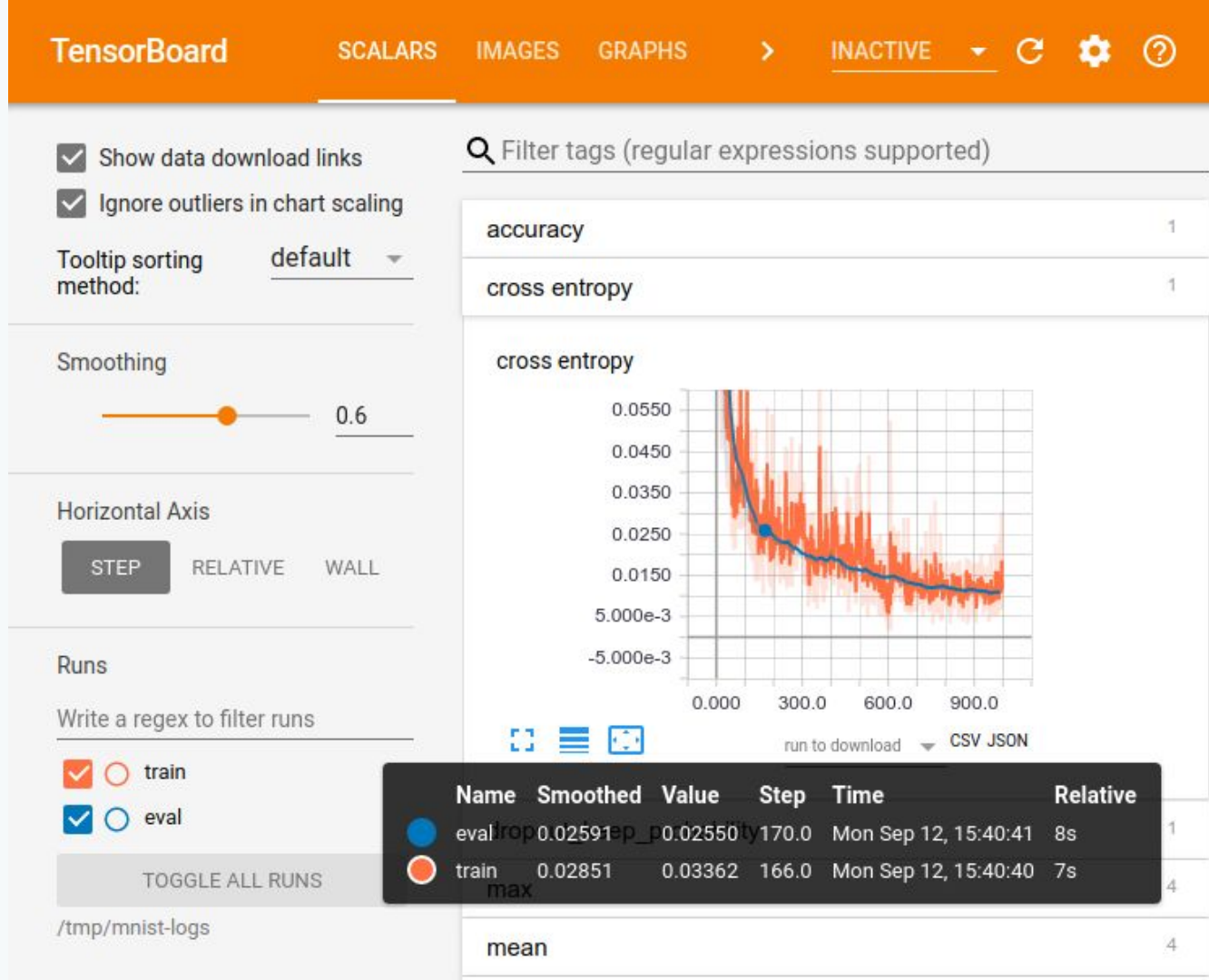
```
x = tf.placeholder(tf.float32, [N, D])
y = tf.placeholder(tf.float32, [N, 1])

res = tf.layers.dense(x, 1)
loss = 0.5 * tf.reduce_sum(tf.square(res - y))
```

# Visualization with TensorBoard

Online demo:

<http://projector.tensorflow.org/>



# Estimators: a high-level API

- Takes care of training, metrics, visualization...
- Premade or custom estimators
- Tutorials
  - [https://www.tensorflow.org/get\\_started/custom\\_estimators](https://www.tensorflow.org/get_started/custom_estimators)
  - [https://www.tensorflow.org/get\\_started/get\\_started\\_for\\_beginners](https://www.tensorflow.org/get_started/get_started_for_beginners)
  - [https://www.tensorflow.org/get\\_started/premade\\_estimators](https://www.tensorflow.org/get_started/premade_estimators)
  - [https://www.tensorflow.org/programmers\\_guide/estimators](https://www.tensorflow.org/programmers_guide/estimators)

# TensorFlow Programming Stack

Low-level  
TensorFlow APIs

Python

C++

Java

Go

TensorFlow  
Kernel

TensorFlow Distributed Execution Engine

# TensorFlow Programming Stack

Mid-Level  
TensorFlow APIs

Layers

Datasets

Metrics

Low-level  
TensorFlow APIs

Python

C++

Java

Go

TensorFlow  
Kernel

TensorFlow Distributed Execution Engine

# TensorFlow Programming Stack

High-Level  
TensorFlow APIs

Estimators

Mid-Level  
TensorFlow APIs

Layers

Datasets

Metrics

Low-level  
TensorFlow APIs

Python

C++

Java

Go

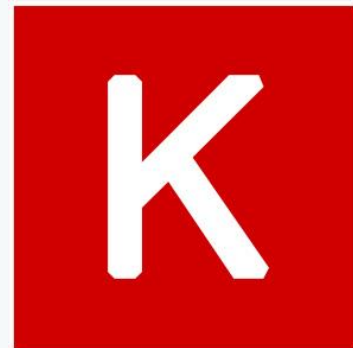
TensorFlow  
Kernel

TensorFlow Distributed Execution Engine



# Keras

- High-level API on top of TensorFlow
- Principles
  - User friendliness
  - Modularity
  - Easy extensibility
- Now inside TensorFlow: `tf.keras`



# Keras

- Define **model**

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
```

```
model = Sequential()

model.add(Dense(units=64, activation='relu',
input_dim=100))
model.add(Dense(units=1, activation=None))
```

# Keras

- Define **model**
- **Compile** to add the loss and optimizer

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
```

```
model = Sequential()

model.add(Dense(units=64, activation='relu',
input_dim=100))
model.add(Dense(units=1, activation=None))
```

```
model.compile(loss='mean_squared_error', optimizer='sgd')
```

# Keras

- Define **model**
- **Compile** to add the loss and optimizer
- Run the **training** on the data

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
```

```
model = Sequential()

model.add(Dense(units=64, activation='relu',
input_dim=100))
model.add(Dense(units=1, activation=None))
```

```
model.compile(loss='mean_squared_error', optimizer='sgd')
```

```
# x_train and y_train are Numpy arrays
x_train = np.random.randn(1000, 100)
y_train = np.random.randn(1000, 1)
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

# Keras

- Define **model**
- **Compile** to add the loss and optimizer
- Run the **training** on the data

```
Epoch 1/5
1000/1000 [=====] - 0s 188us/step - loss: 1.6796
Epoch 2/5
1000/1000 [=====] - 0s 54us/step - loss: 1.2774
Epoch 3/5
1000/1000 [=====] - 0s 44us/step - loss: 1.1319
Epoch 4/5
1000/1000 [=====] - 0s 44us/step - loss: 1.0620
Epoch 5/5
1000/1000 [=====] - 0s 44us/step - loss: 1.0084
```

# PyTorch vs. TensorFlow

---

- Which one to choose?
  - Facebook vs. Google
- **TensorFlow** more used today, **PyTorch** gaining ground

# PyTorch vs. TensorFlow

- Which one to choose?
  - Facebook vs. Google
- **TensorFlow** more used today, **PyTorch** gaining ground
- TensorFlow: more features, static graph
  - Better in **production**
- PyTorch: dynamic graph, closer to numpy, easier
  - Currently difficult for production
  - Release 1.0 will make it easier (this summer)

# PyTorch vs. TensorFlow in Paris

## PyTorch

- Doctrine
- Facebook AI Research

## TensorFlow (+ Keras)

- Riminder
- Owkin
- Cardiologs
- Google Brain
- DeepMind



# List of resources for TensorFlow

---

- Machine learning crash course
  - <https://developers.google.com/machine-learning/crash-course/>
- TensorFlow documentation
  - <https://tensorflow.org>
- TensorFlow Dev Summit videos
- TensorFlow blog
  - <https://blog.tensorflow.org>
- Official examples:  
<https://github.com/tensorflow/models>

# List of resources for PyTorch

---

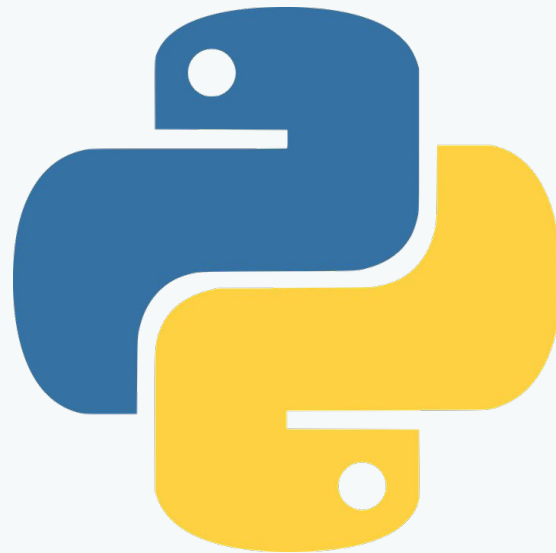
- PyTorch documentation:
  - <https://pytorch.org/>
- Official tutorials:
  - <https://github.com/pytorch/tutorials>
- Official examples:
  - <https://github.com/pytorch/examples>
- PyTorch discussion forum:
  - <https://discuss.pytorch.org/>
- Justin Johnson (CS231n) tutorials:
  - <https://github.com/jcjohnson/pytorch-examples>

# Best practices

---

# Python

- Python3 is great
  - Don't use python2



# Python

- Python is great
- Have a good install:
  - <http://docs.python-guide.org>
  - Use pip
  - Virtual environments

```
# Create project
mkdir project
cd project

# Create virtual env named .env
python -m venv .env
# Activate the virtual env
source .env/bin/activate

# Install python packages
pip install -r requirements.txt
```



# Python

- Python is great
- Have a good install:  
<http://docs.python-guide.org>
- Coding style: PEP8, Pylint

```
> pylint model/input_fn.py
***** Module model.input_fn
W: 3, 0: Unused tensorflow imported as tf (unused-import)

-----
Your code has been rated at 9.29/10 (previous run: 9.29/10, +0.00)
```

# Python

- Python is great
- Have a good install:  
<http://docs.python-guide.org>
- Coding style
  - PEP8
  - Pylint
- Code organization

```
▼ model/  
  ▼ tests/  
    __init__.py  
    test_triplet_loss.py  
    __init__.py  
    input_fn.py  
    mnist_dataset.py  
    model_fn.py  
    triplet_loss.py  
    utils.py  
  evaluate.py  
  LICENSE  
  README.md  
  requirements_cpu.txt  
  requirements_gpu.txt  
  search_hyperparams.py  
  train.py  
  visualize_embeddings.py
```



# Python

- Python is great
- Have a good install:  
<http://docs.python-guide.org>
- Coding style
  - PEP8
  - Pylint
- Code organization

Scripts

```
▼ model/  
  ▼ tests/  
    __init__.py  
    test_triplet_loss.py  
    __init__.py  
    input_fn.py  
    mnist_dataset.py  
    model_fn.py  
    triplet_loss.py  
    utils.py  
    evaluate.py  
    LICENSE  
    README.md  
    requirements_cpu.txt  
    requirements_gpu.txt  
    search_hyperparams.py  
    train.py  
    visualize_embeddings.py
```

# Python

- Python is great
- Have a good install:  
<http://docs.python-guide.org>
- Coding style
  - PEP8
  - Pylint
- Code organization

Module

Scripts

```
▼ model/  
  ▼ tests/  
    __init__.py  
    test_triplet_loss.py  
    __init__.py  
    input_fn.py  
    mnist_dataset.py  
    model_fn.py  
    triplet_loss.py  
    utils.py  
  evaluate.py  
  LICENSE  
  README.md  
  requirements_cpu.txt  
  requirements_gpu.txt  
  search_hyperparams.py  
  train.py  
  visualize_embeddings.py
```

# Python

- Python is great
- Have a good install:  
<http://docs.python-guide.org>
- Coding style
  - PEP8
  - Pylint
- Code organization

## Tests

## Module

## Scripts

```
▼ model/  
  ▼ tests/  
    __init__.py  
    test_triplet_loss.py  
    __init__.py  
    input_fn.py  
    mnist_dataset.py  
    model_fn.py  
    triplet_loss.py  
    utils.py  
  evaluate.py  
  LICENSE  
  README.md  
  requirements_cpu.txt  
  requirements_gpu.txt  
  search_hyperparams.py  
  train.py  
  visualize_embeddings.py
```

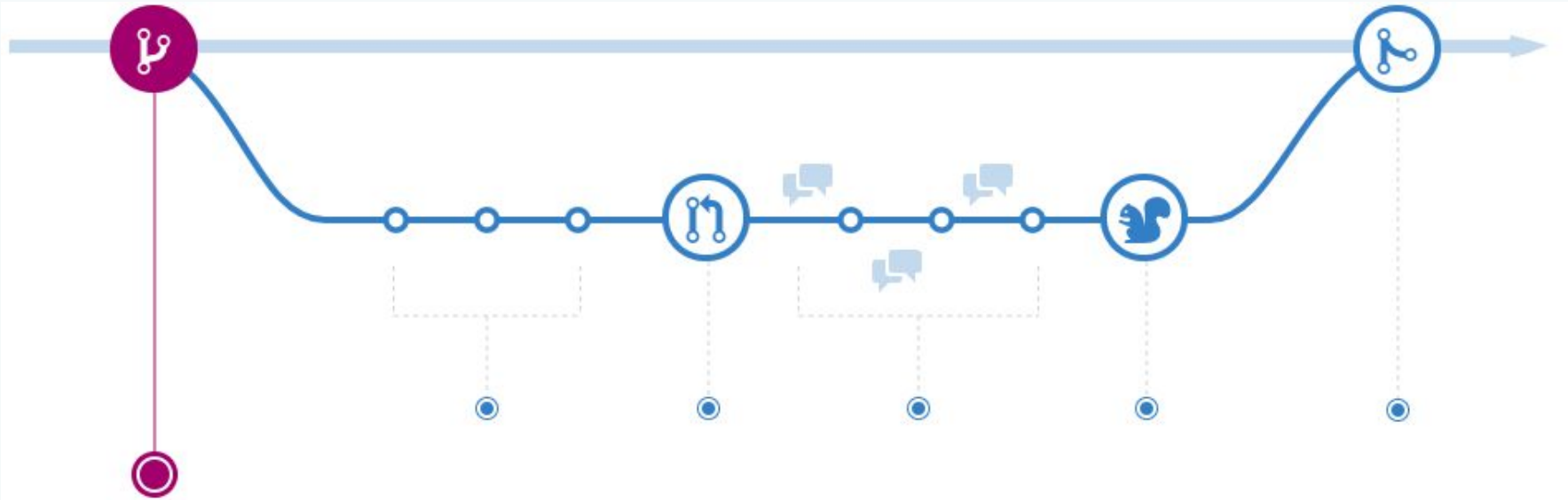
# Github

---

- Version control
  - (tracks changes in files)
- Commit = record change to the repository
- Collaboration made easy

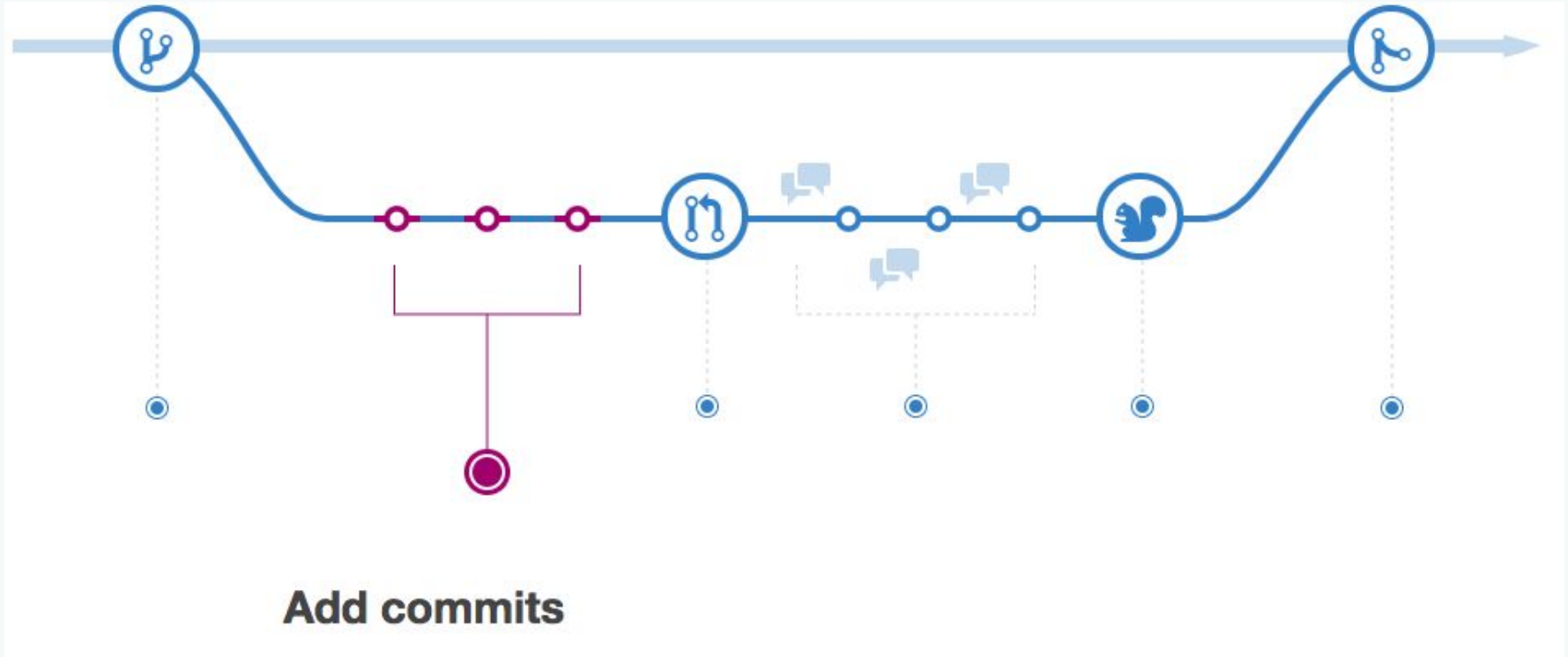


# Github

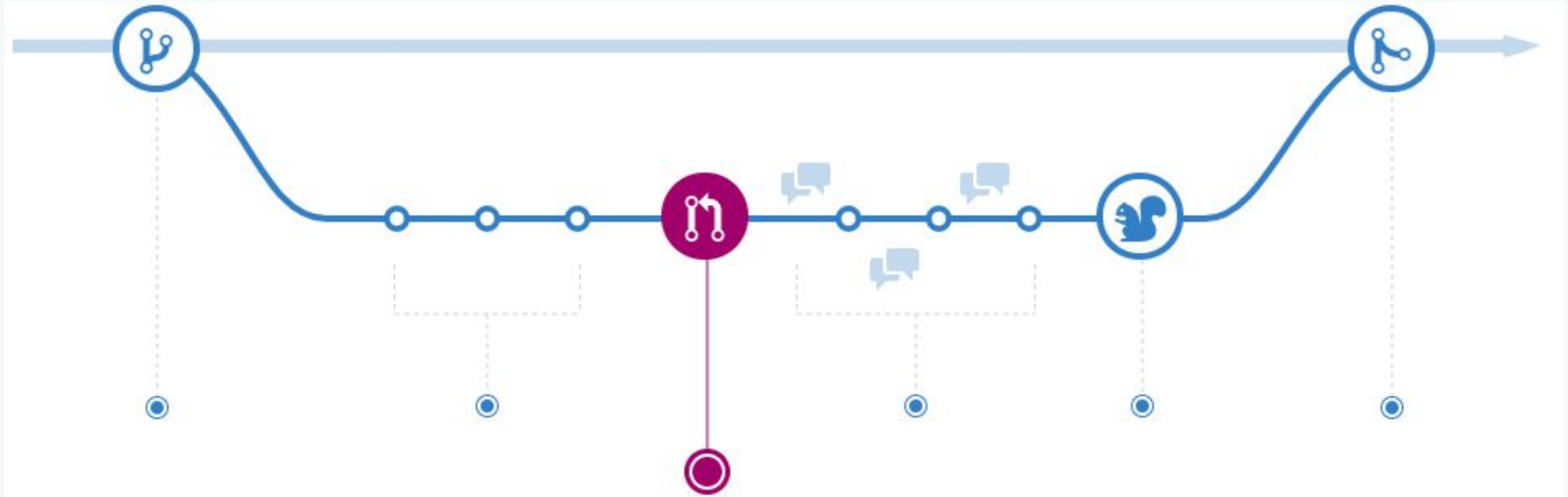


## Create a branch

# Github

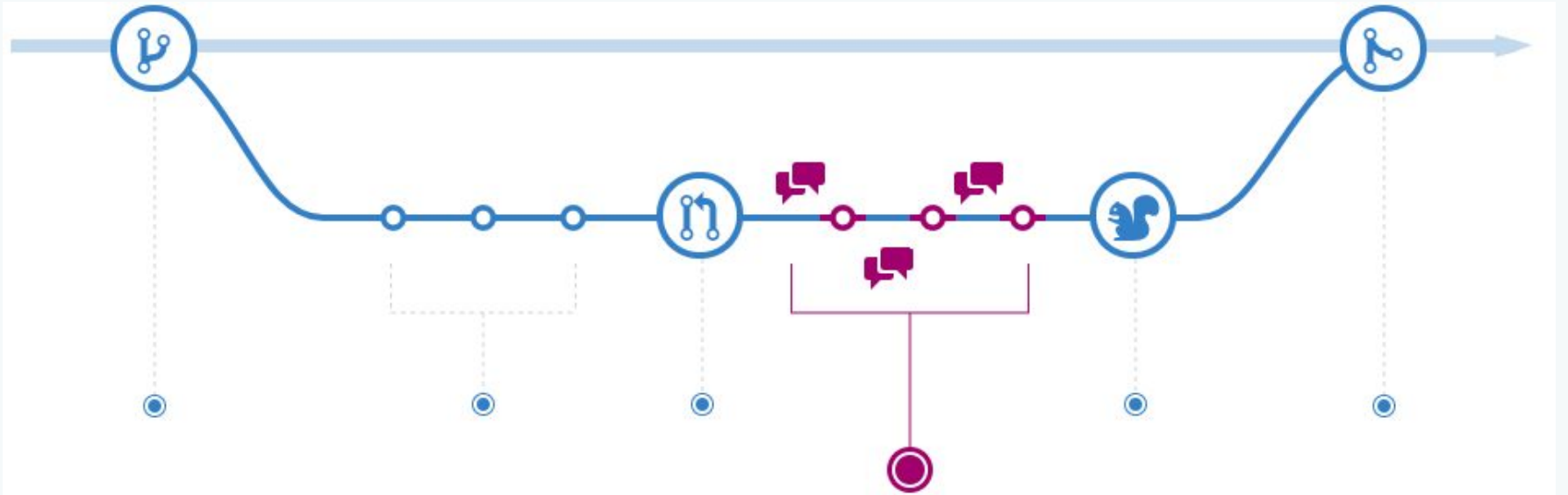


# Github



**Open a Pull Request**

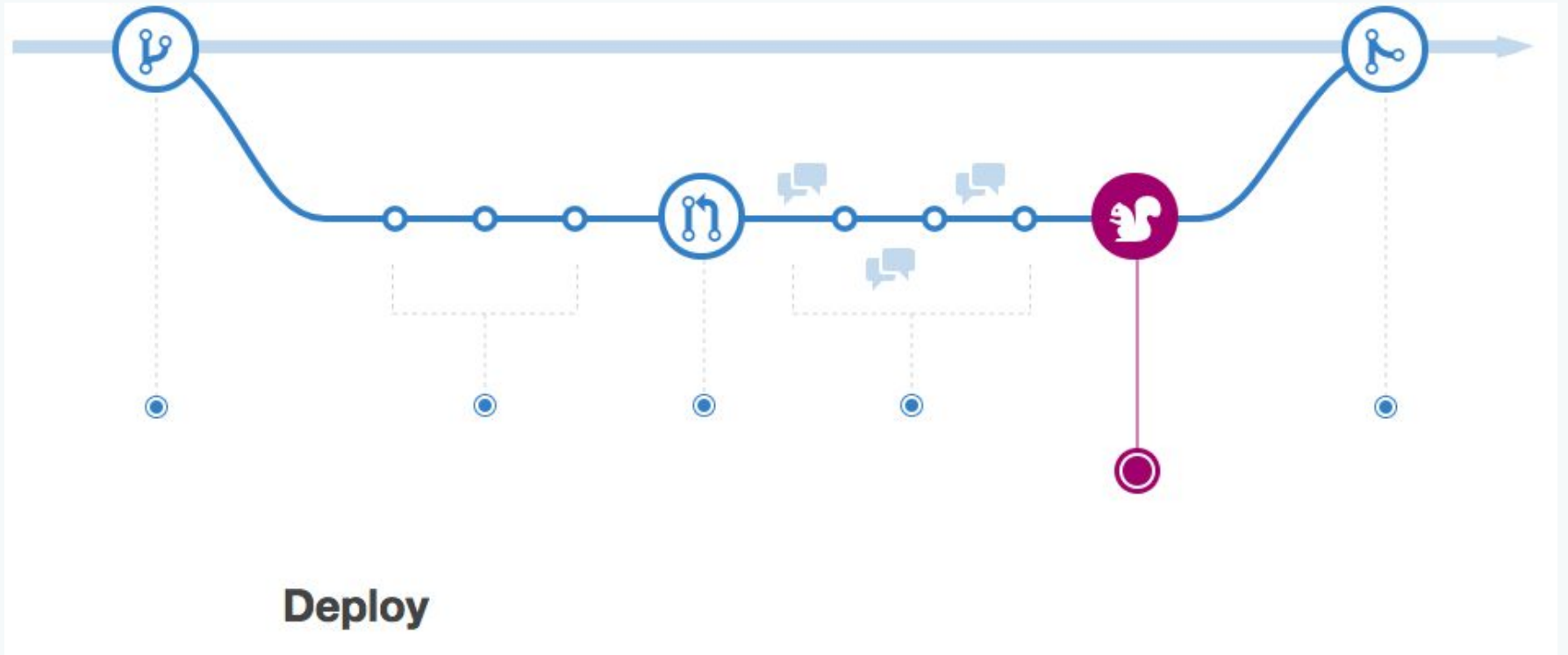
# Github



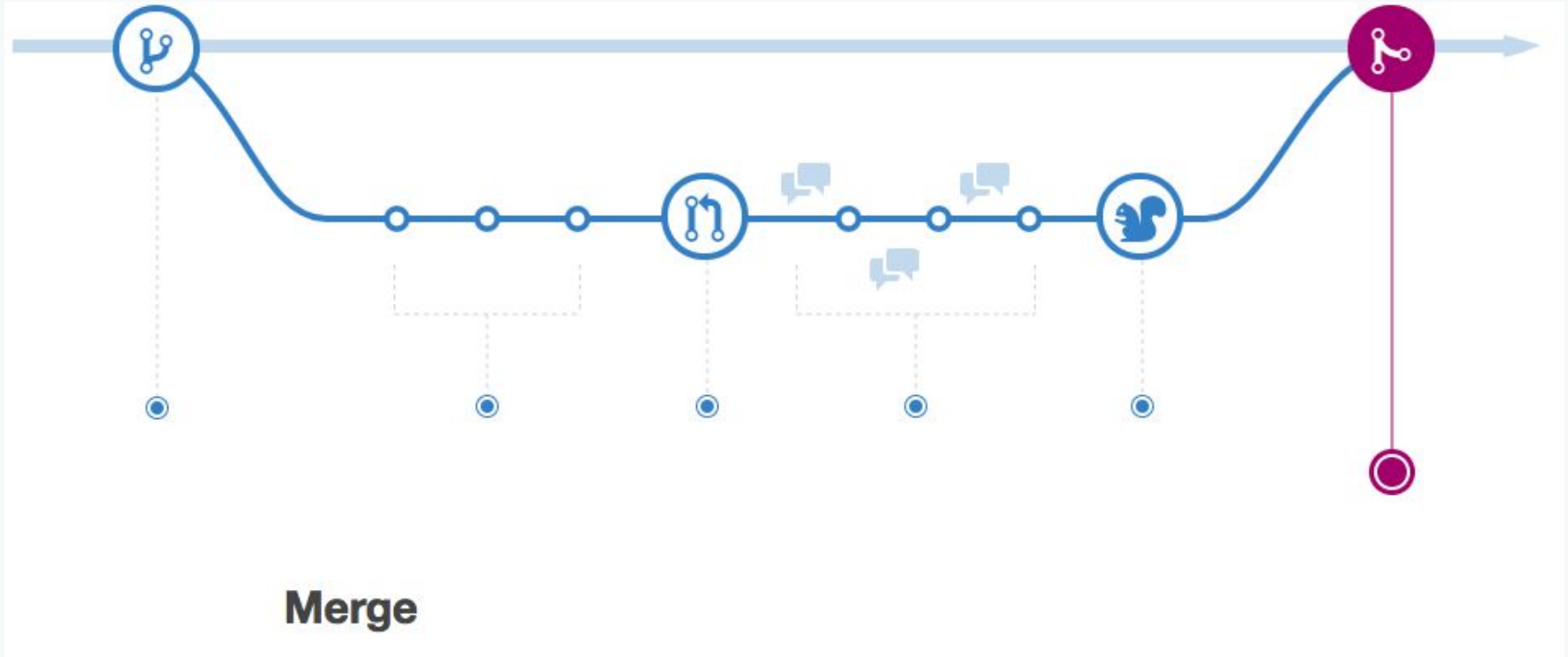
**Discuss and review your code**



# Github



# Github



# Testing your code

- Writing tests = good practice

- <http://docs.python-guide.org/en/latest/writing/tests/>

Tests

Module

Scripts

```
▼ model/  
  ▼ tests/  
    __init__.py  
    test_triplet_loss.py  
    __init__.py  
    input_fn.py  
    mnist_dataset.py  
    model_fn.py  
    triplet_loss.py  
    utils.py  
  evaluate.py  
  LICENSE  
  README.md  
  requirements_cpu.txt  
  requirements_gpu.txt  
  search_hyperparams.py  
  train.py  
  visualize_embeddings.py
```

# Testing your code

- Writing tests = good practice
- How to write a test
  - Write test function

```
def dummy(x):  
    return x + 2  
  
def test_dummy():  
    x = 3  
    res = dummy(x)  
    assert res == 5
```

# Testing your code

- Writing tests = good practice
- How to write a test
  - Write test function
  - Run with pytest  
(<https://pytest.org>)

```
def dummy(x):  
    return x + 2  
  
def test_dummy():  
    x = 3  
    res = dummy(x)  
    assert res == 4
```

# Testing your code

```
collected 1 item

tf_test.py F [100%]

===== FAILURES =====
_____ test_dummy _____

    def test_dummy():
        x = 3
        res = dummy(x)
>       assert res == 4
E       assert 5 == 4

tf_test.py:7: AssertionError
===== 1 failed in 0.08 seconds =====
```

```
def dummy(x):
    return x + 2

def test_dummy():
    x = 3
    res = dummy(x)
    assert res == 4
```

# Testing your code

- Writing tests = good practice
- How to write a test
- Continuous integration
  - (A bit advanced)
  - <https://travis-ci.org>

Travis.ci configuration

```
language: python
python:
  - "2.7"
  - "3.5"
  - "3.6"
# command to install dependencies
install:
  - pip install -r requirements_cpu.txt
# command to run tests
script:
  - pytest
```

# An example

<https://github.com/omoindrot/tensorflow-triplet-loss>



# Text editor

---

- Vim (+ tmux)
  - High learning curve, but a lot of keybinds / shortcuts
- Sublime Text 3 or Atom
  - Easiest to get started with, good default editor
- *PyCharm*
  - Slower, with more features for big projects

# Developing in python

- Python files
- Ipython notebooks
- Ipython in terminal

Live  
demo(My work  
environment)

```

1 train.py
parser.add_argument('--model_dir', default='experiments/base_model',
                    help="Experiment directory containing params.json")
parser.add_argument('--data_dir', default='data/mnist',
                    help="Directory containing the dataset")

if __name__ == '__main__':
    tf.reset_default_graph()
    tf.logging.set_verbosity(tf.logging.INFO)

    # Load the parameters from json file
    args = parser.parse_args()
    json_path = os.path.join(args.model_dir, 'params.json')
    assert os.path.isfile(json_path), "No json configuration file found at {}".format(json_path)
    params = Params(json_path)

    # Define the model
    tf.logging.info("Creating the model...")
    config = tf.estimator.RunConfig(tf_random_seed=230,
                                    model_dir=args.model_dir,
                                    save_summary_steps=params.save_summary_steps)
    estimator = tf.estimator.Estimator(model_fn, params=params, config=config)

    # Train the model
    tf.logging.info("Starting training for {} epoch(s)".format(params.num_epochs))
    estimator.train(lambda: train_input_fn(args.data_dir, params))

    # Evaluate the model on the test set
    tf.logging.info("Evaluation on test set.")
    res = estimator.evaluate(lambda: test_input_fn(args.data_dir, params))
    for key in res:
        print("{}: {}".format(key, res[key]))

```

NORMAL master train.py 78% 36:5  
~/workspace/tensorflow-triplet-loss/train.py" 46L, 1637C

```

~/workspace/tensorflow-triplet-loss master*
> source .env/bin/activate

~/workspace/tensorflow-triplet-loss master*
.env > ipython
Python 3.6.5 (default, Mar 30 2018, 06:41:53)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: run train.py --model_dir experiments/base_model

```

macbook-ai

```

1 model_fn.py
with tf.variable_scope('model'):
    # Compute the embeddings with the model
    embeddings = build_model(is_training, images, params)
    embedding_mean_norm = tf.reduce_mean(tf.norm(embeddings, axis=1))
    tf.summary.scalar("embedding_mean_norm", embedding_mean_norm)

    if mode == tf.estimator.ModeKeys.PREDICT:
        predictions = {'embeddings': embeddings}
        return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)

    labels = tf.cast(labels, tf.int64)

    # Define triplet loss
    if params.triplet_strategy == "batch_all":
        loss, fraction = batch_all_triplet_loss(labels, embeddings, margin=params.margin,
                                                squared=params.squared)
    elif params.triplet_strategy == "batch_hard":
        loss = batch_hard_triplet_loss(labels, embeddings, margin=params.margin,
                                       squared=params.squared)
    else:
        raise ValueError("Triplet strategy not recognized: {}".format(params.triplet_strategy))

    # -----
    # METRICS AND SUMMARIES
    # Metrics for evaluation using tf.metrics (average over whole dataset)
    # TODO: some other metrics like rank-1 accuracy?
    with tf.variable_scope("metrics"):
        eval_metric_ops = {"embedding_mean_norm": tf.metrics.mean(embedding_mean_norm)}

        if params.triplet_strategy == "batch_all":
            eval_metric_ops['fraction_positive_triplets'] = tf.metrics.mean(fraction)

```

NORMAL master model\_fn.py 67% 79:49  
:NEROTreeToggle

```

2018-05-03 14:11:16.188875: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports in
structions that this TensorFlow binary was not compiled to use: AVX2 FMA
W0503 14:11:16.271594 Reloader tf_logging.py:121] Found more than one graph event per run, or there w
as a metagraph containing a graph_def, as well as one or more graph events. Overwriting the graph wi
th the newest event.
W0503 14:11:16.273255 Reloader tf_logging.py:121] Found more than one metagraph event per run. Overwr
iting the metagraph with the newest event.
W0503 14:11:16.556826 Reloader tf_logging.py:121] Found more than one graph event per run, or there w
as a metagraph containing a graph_def, as well as one or more graph events. Overwriting the graph wi
th the newest event.
W0503 14:11:16.560618 Reloader tf_logging.py:121] Found more than one metagraph event per run. Overwr
iting the metagraph with the newest event.
TensorBoard 1.7.0 at http://macbook-air-de-olivier-2.home:6006 (Press CTRL+C to quit)

```

tensorboard

14:11 03/05/2018

# Daily life in deep learning

- Read papers
- Think about models on a whiteboard
- **Implement stuff**
  - Clean data, boilerplate around the model...
  - Small part for the model implementation
- Train models

# Resources for ML/AI

---

- This course
- Stanford classes
  - CS229: Machine Learning
  - **CS231n: Vision**
  - CS224n: NLP
  - CS230: Deep Learning  
(<https://cs230-stanford.github.io/>)
- Read papers
- AI community
  - Reddit ML
  - Facebook “Apprentissage profond”
  - <http://www.arxiv-sanity.com/>
- List of resources:  
<https://github.com/BAILOOL/DoYouEvenLearn>

# Point of Contact

---



Olivier Moindrot  
Data scientist, Owkin

omoindrot@gmail.com

<https://omoindrot.github.io>

Link to these slides (with speaker notes):

<https://tinyurl.com/deepframeworks>