
# 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**

riminder

https://github.com/riminder

# Talk outline

I. *CPU vs GPU (vs TPU)*

II. *Frameworks overview: TensorFlow, PyTorch, Keras...*

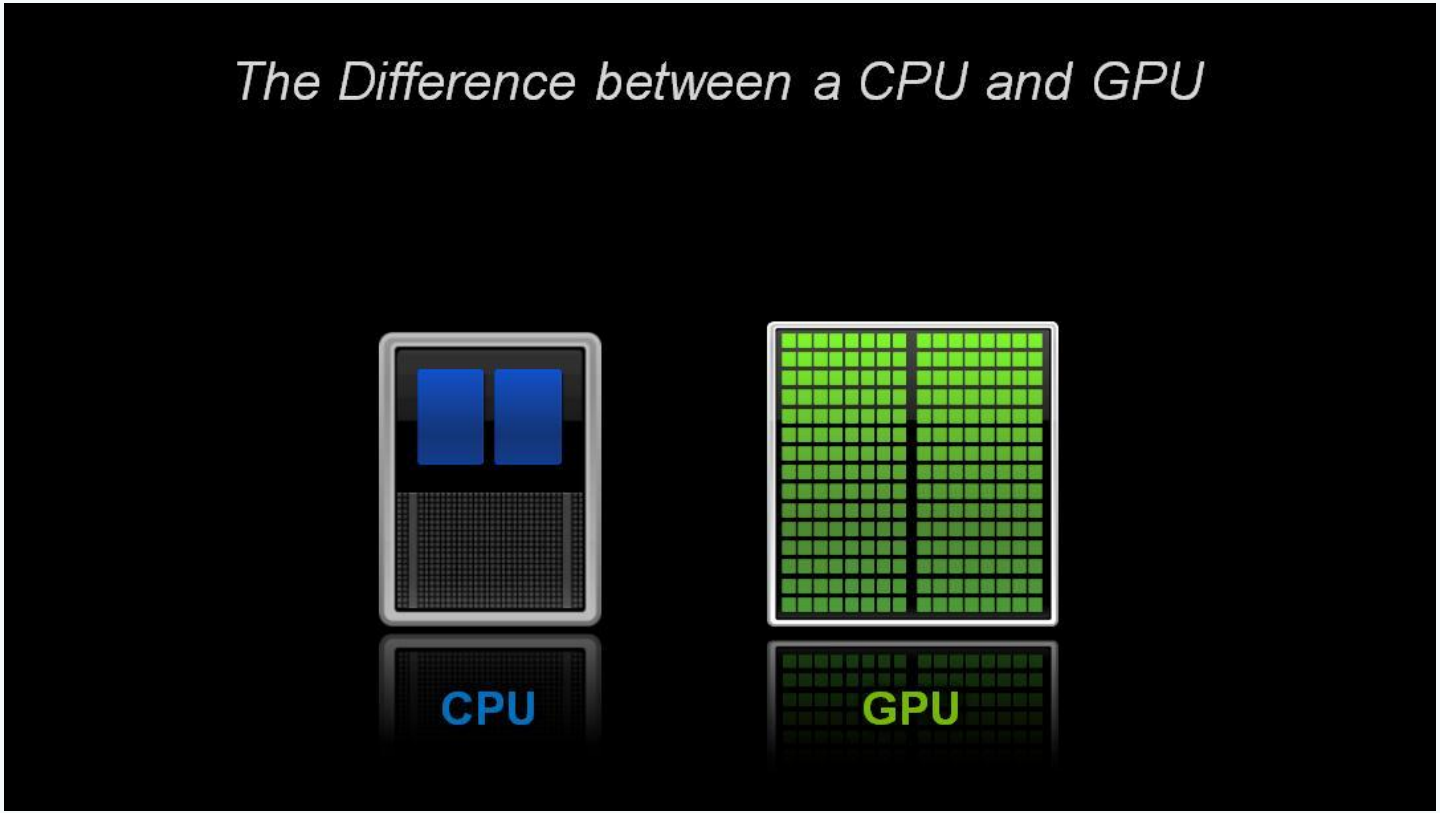
III. *Focus on TensorFlow*

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




https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

# CPU, GPU

| | # Cores | Clock Speed | Memory | Price | Speed |
|---|---|---|---|---|---|
| **CPU** (Intel Core i7-7700k) | 4 (8 threads with hyperthreading) | 4.2 GHz | System RAM | $339 | ~540 GFLOPs FP32 |
| **GPU** (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

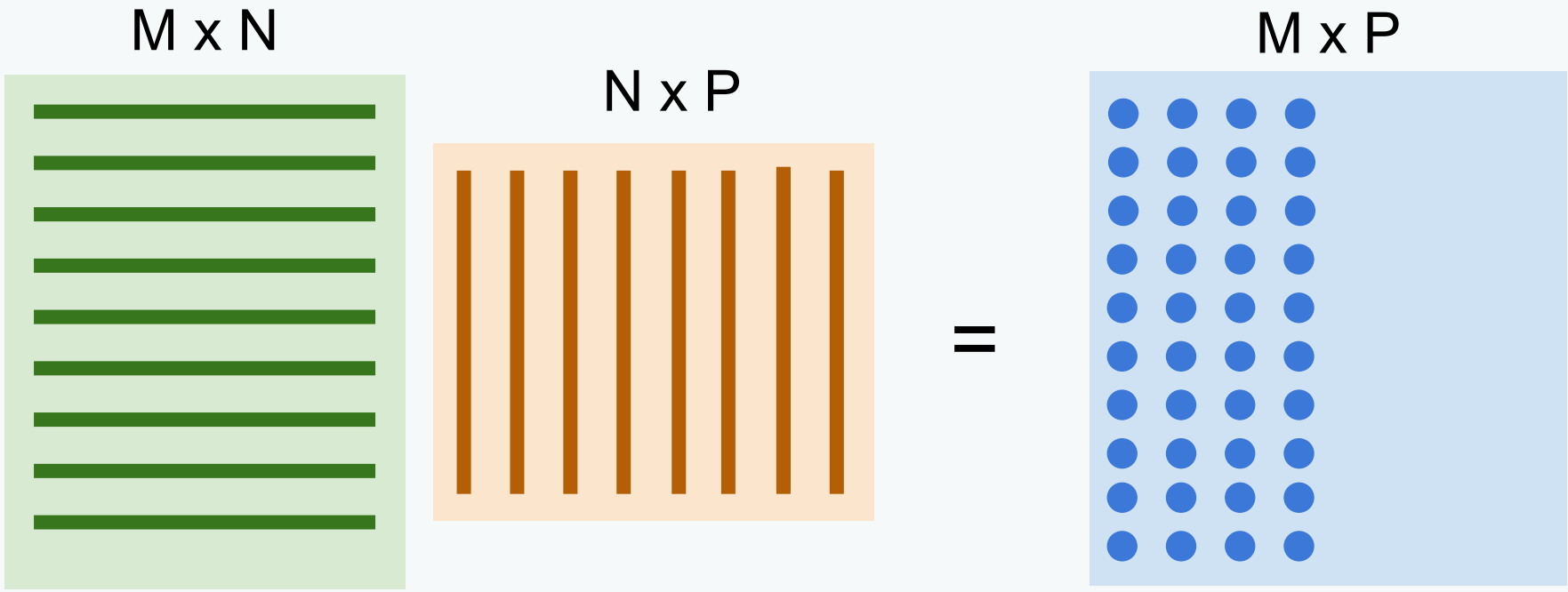
| | # Cores | Clock Speed | Memory | Price | Speed |
|---|---|---|---|---|---|
| **CPU** (Intel Core i7-7700k) | 4 (8 threads with hyperthreading) | 4.2 GHz | System RAM | $339 | ~540 GFLOPs FP32 |
| **GPU** (NVIDIA Titan Xp) | 3584 | 1.6 GHz | 11 GB GDDR5 X | $699 | ~11.4 TFLOPs FP32 |
| **TPU** NVIDIA TITAN V | 5120 CUDA, 640 Tensor | 1.5 GHz | 12 GB HBM2 | $2999 | ~14 TFLOPs FP32 ~112 TFLOPs FP16 |
| **TPU** 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

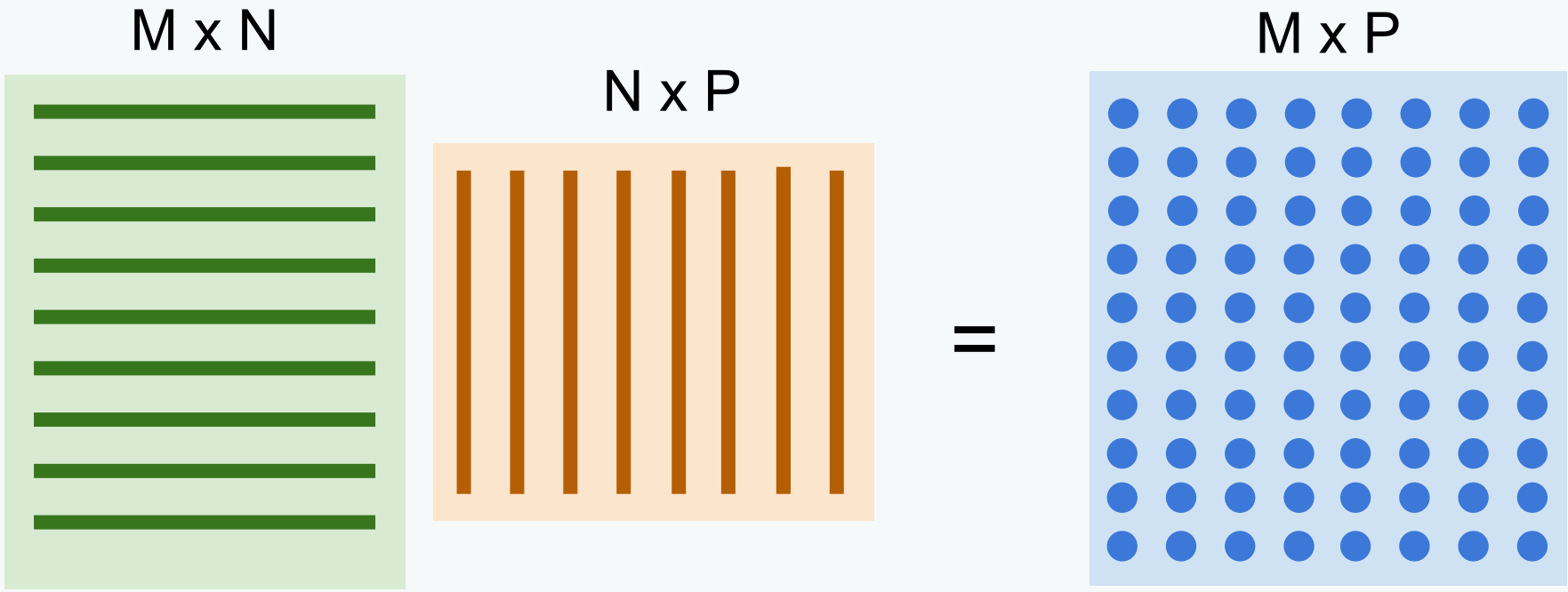
**GPU**: More cores, but each core is much slower and "dumber"; great for parallel tasks

**TPU**: Specialized hardware for deep learning

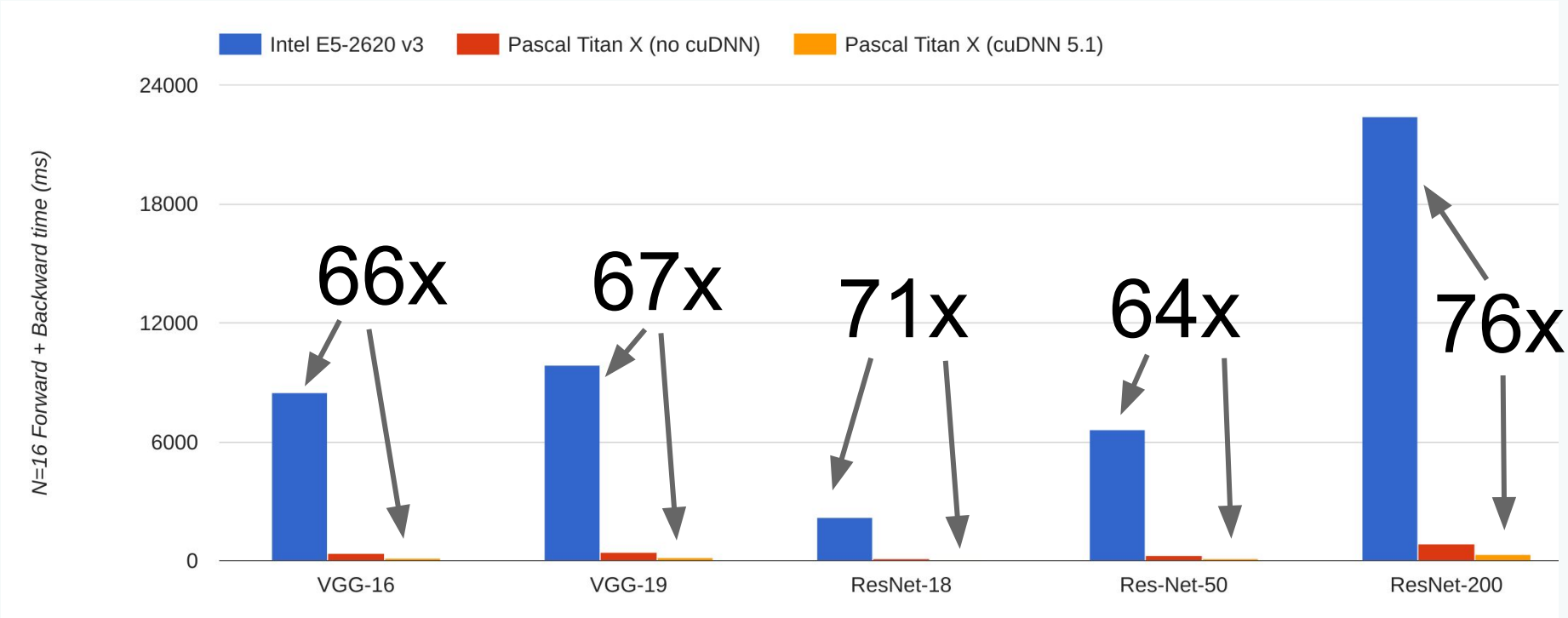
# Example: matrix multiplication



# Example: matrix multiplication



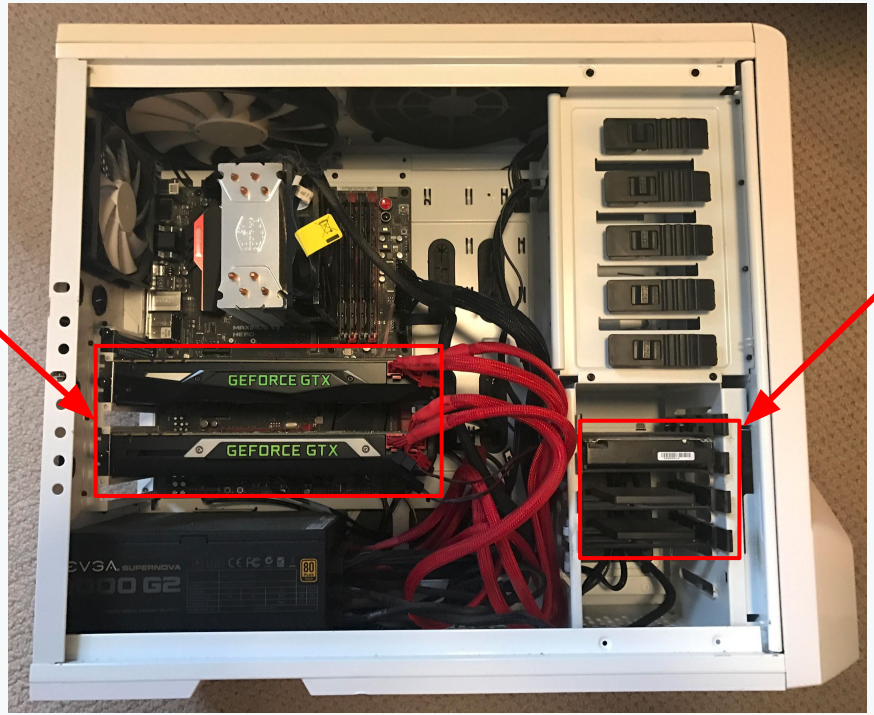
# Benchmark: CPU vs GPU



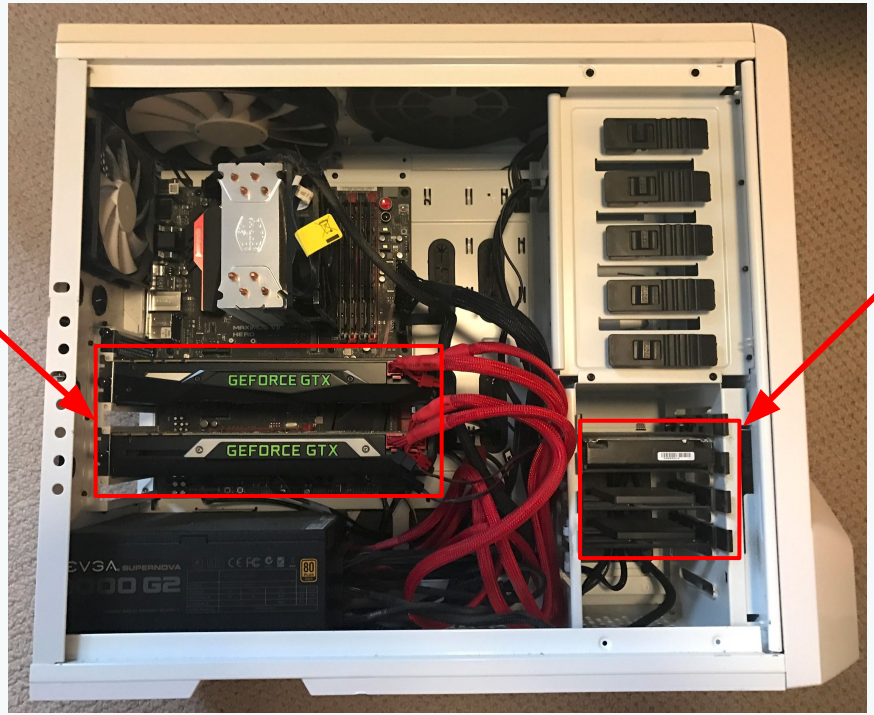

Data from https://github.com/jcjohnson/cnn-benchmarks

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

# CPU / GPU communication



# CPU / GPU communication




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

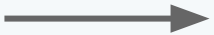
# Deep Learning Frameworks

# Deep Learning Frameworks: Overview

Caffe
(UC Berkeley)

Caffe2
(Facebook)

Torch
(NYU / Facebook)



Torch
(NYU / Facebook)

Theano
(U Montreal)

TensorFlow
(Google)

Keras
(Google)

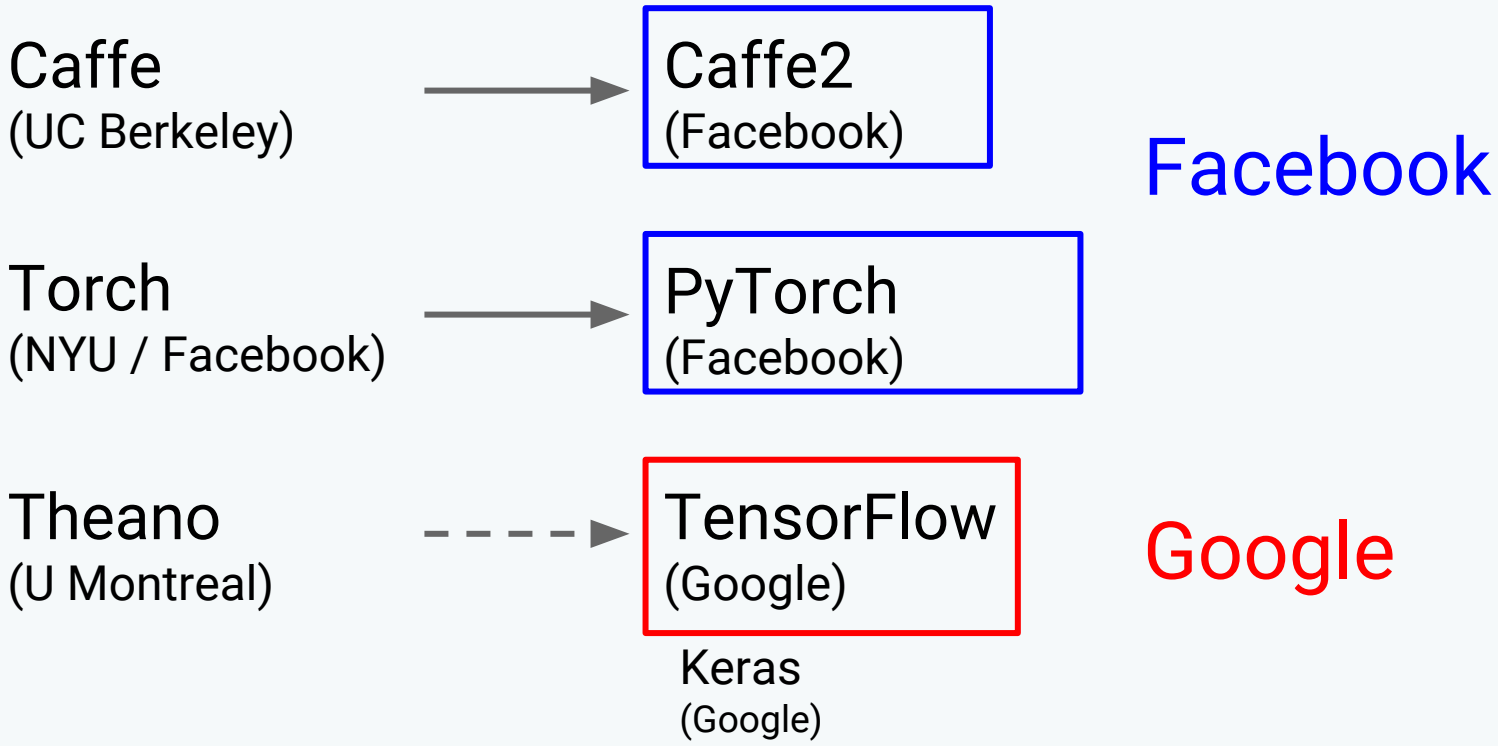
PaddlePaddle
(Baidu)

Core ML
(Apple)

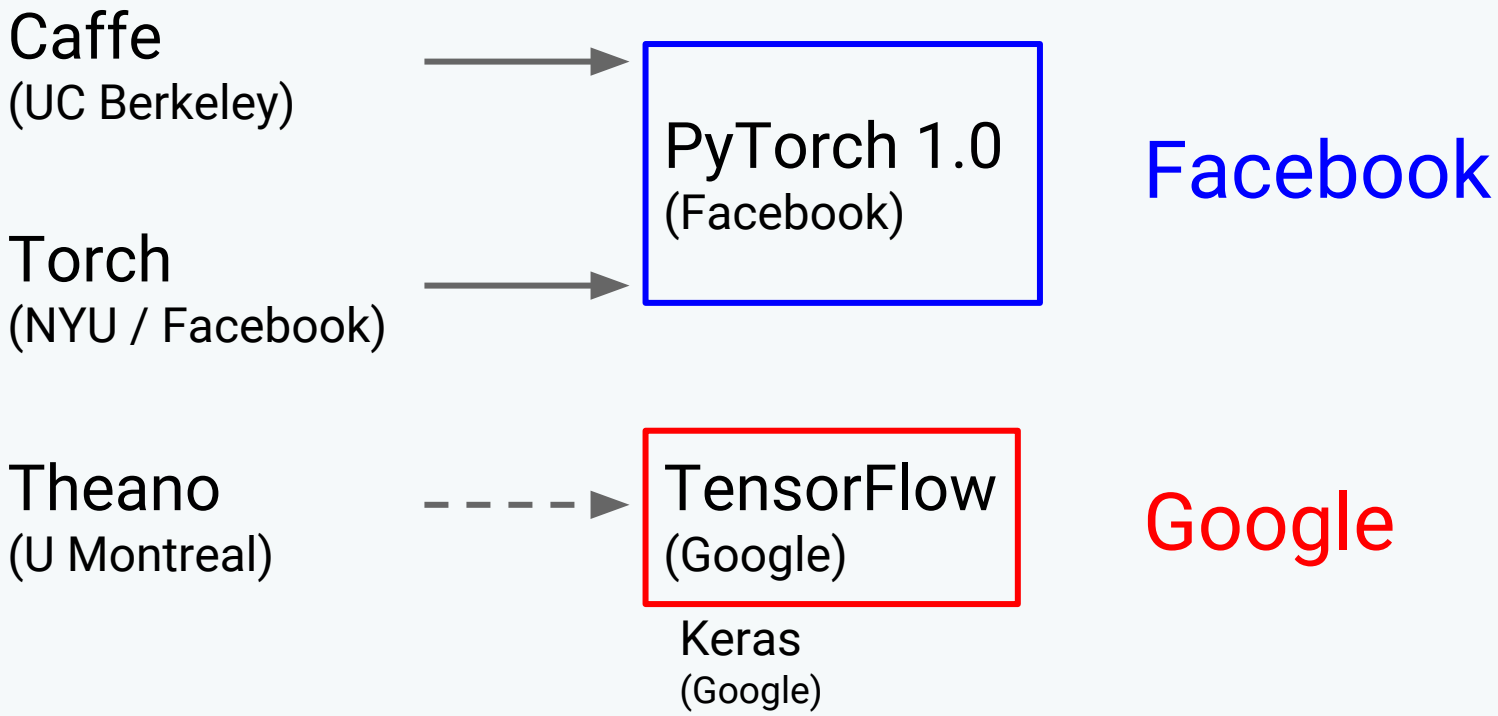
CNTK
(Microsoft)

MXNet
(Amazon)

# Deep Learning Frameworks: Overview



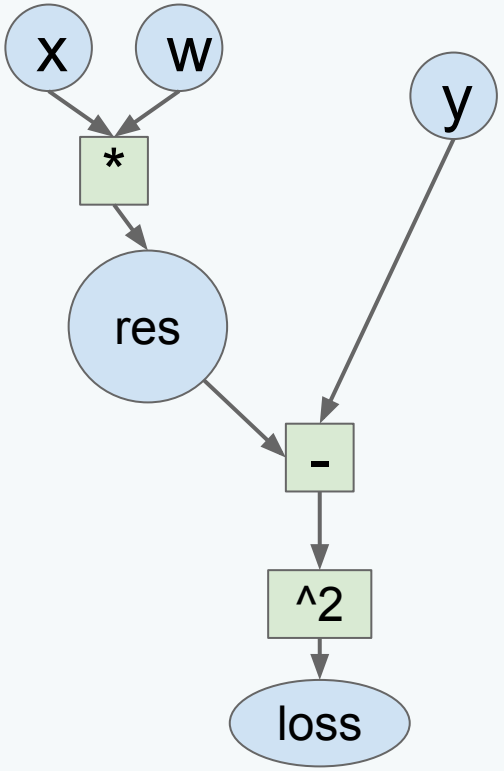
# Deep Learning Frameworks: Overview



# What is a deep learning framework

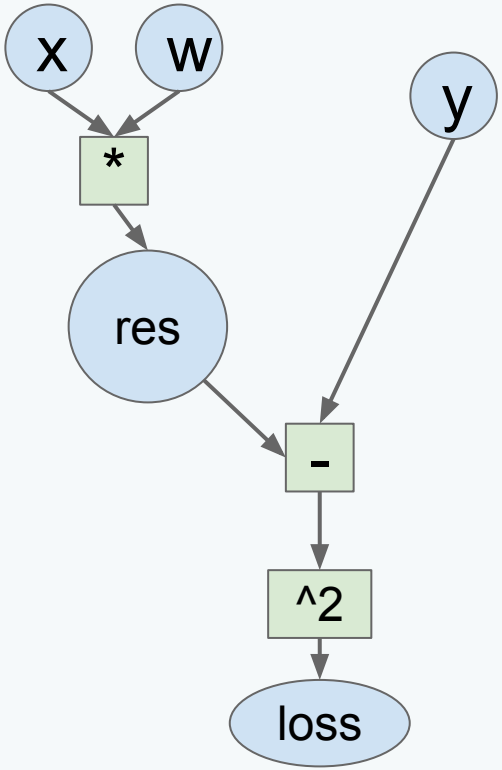
Computational graph

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



# 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

*http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture8.pdf*

# Computational graph

## Numpy

```python
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

```python
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 !

*http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture8.pdf*

# Computational graph

Free gradients !

## Numpy

```python
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)
```

*http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture8.pdf*

## PyTorch

```python
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)
```

*http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture8.pdf*

# 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

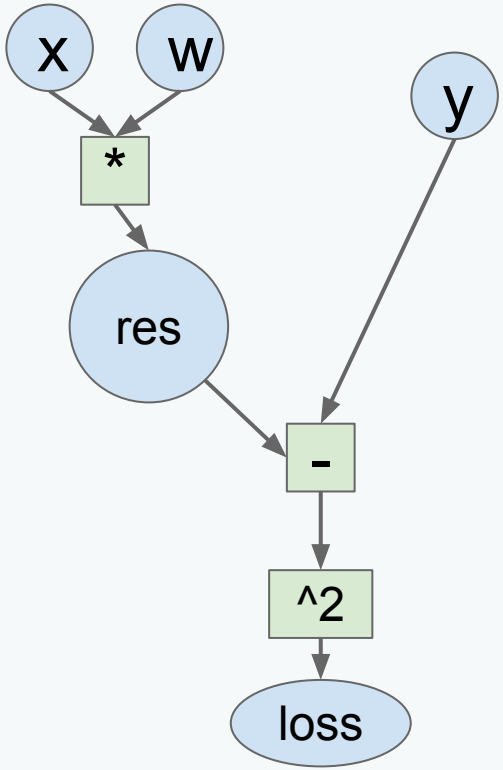
# Getting started

Read the official documentation:
https://www.tensorflow.org/get_started/

# 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))
```

https://www.tensorflow.org/programmers_guide/graphs

# 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))
```

https://www.tensorflow.org/programmers_guide/graphs

# 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

```
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))
```

Need to feed the values of the placeholders

# 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))
```

https://www.tensorflow.org/programmers_guide/variables

# 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))
```

https://www.tensorflow.org/programmers_guide/variables

# 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/programmers_guide/datasets
  - https://www.tensorflow.org/performance/datasets_performance

# Creating the model

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

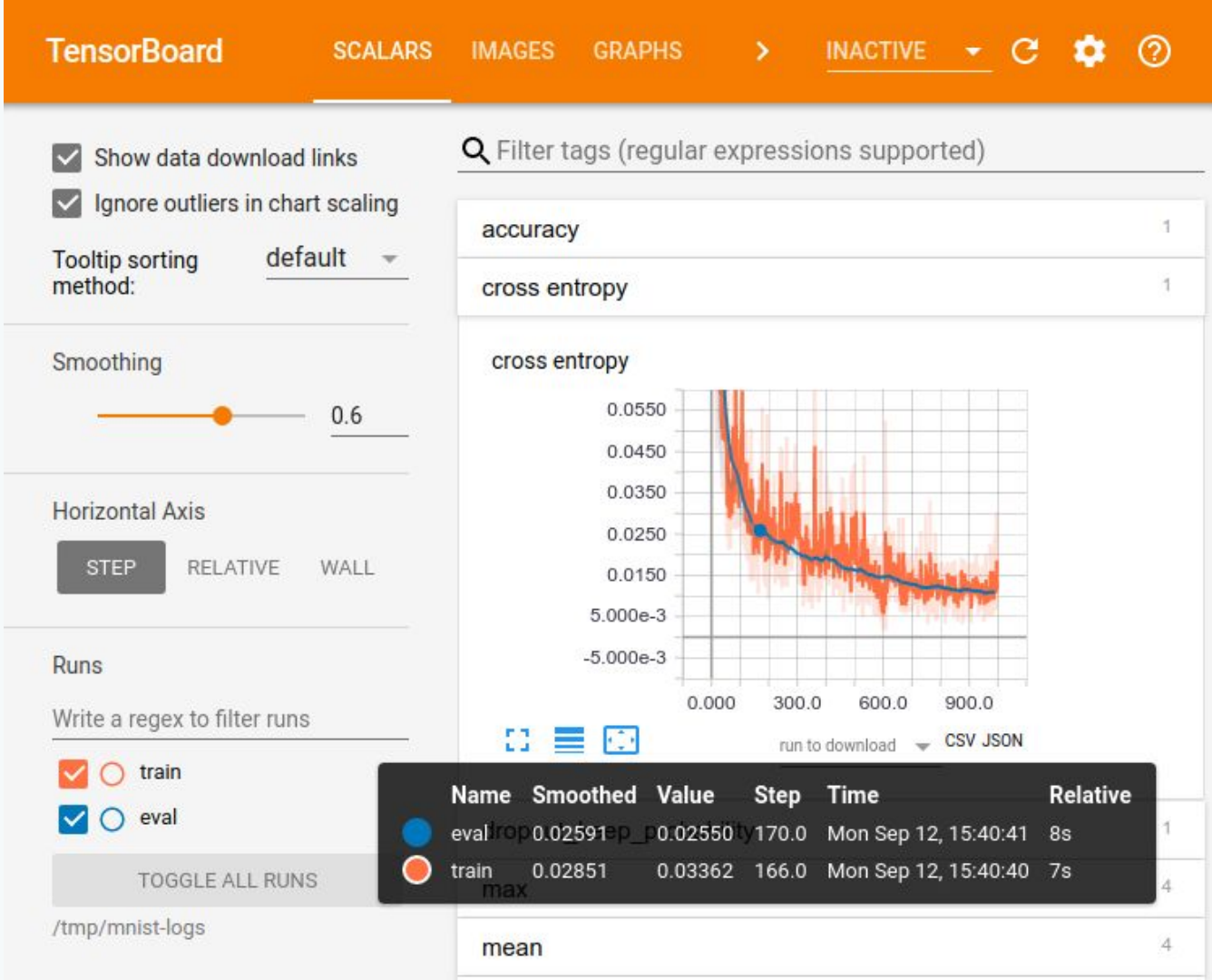
```
x = tf.placeholder(tf.float32, [N, D])
y = tf.placeholder(tf.float32, [N, 1])

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

TensorFlow

# Visualization with TensorBoard

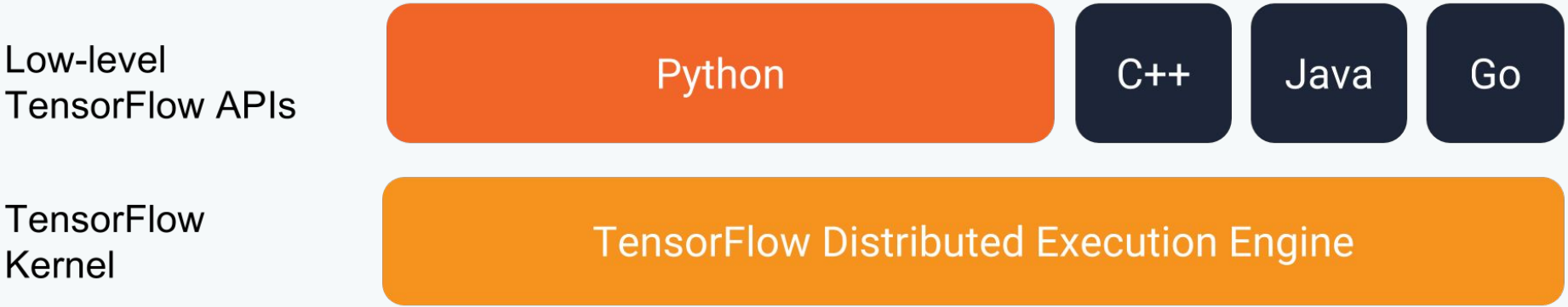
Online demo:
http://projector.tensorflow.org/



# Estimators: a high-level API

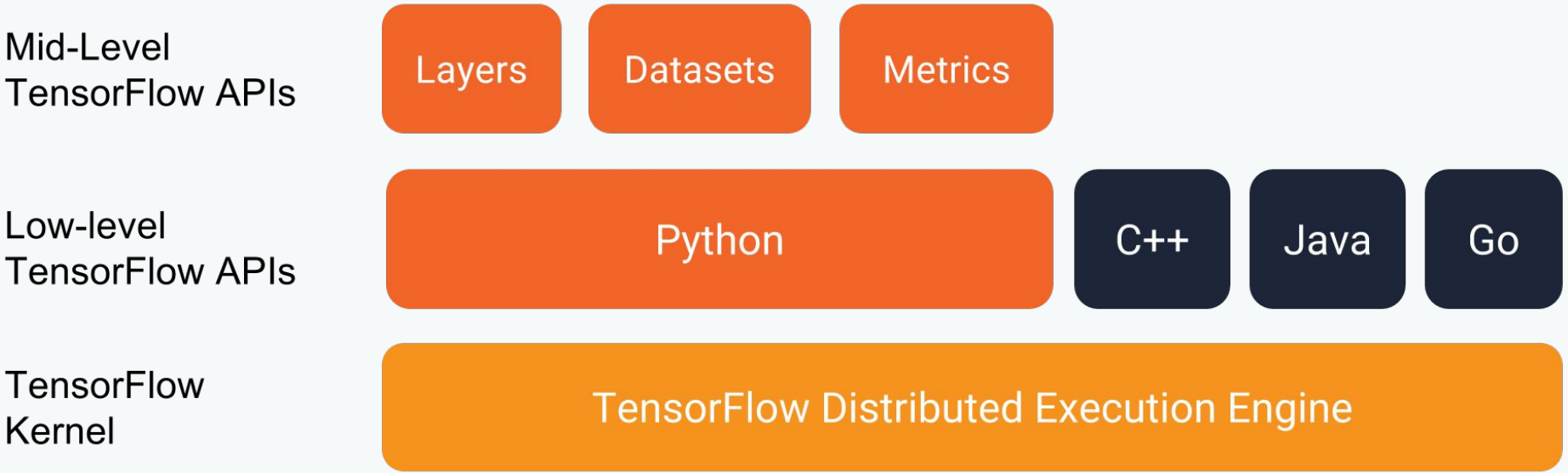
- Takes care of training, metrics, visualization…
- Premade or custom estimators
- Tutorials
  - https://www.tensorflow.org/get_started/custom_estimators
  - https://www.tensorflow.org/get_started/get_started_for_beginners
  - https://www.tensorflow.org/get_started/premade_estimators
  - https://www.tensorflow.org/programmers_guide/estimators

# TensorFlow Programming Stack



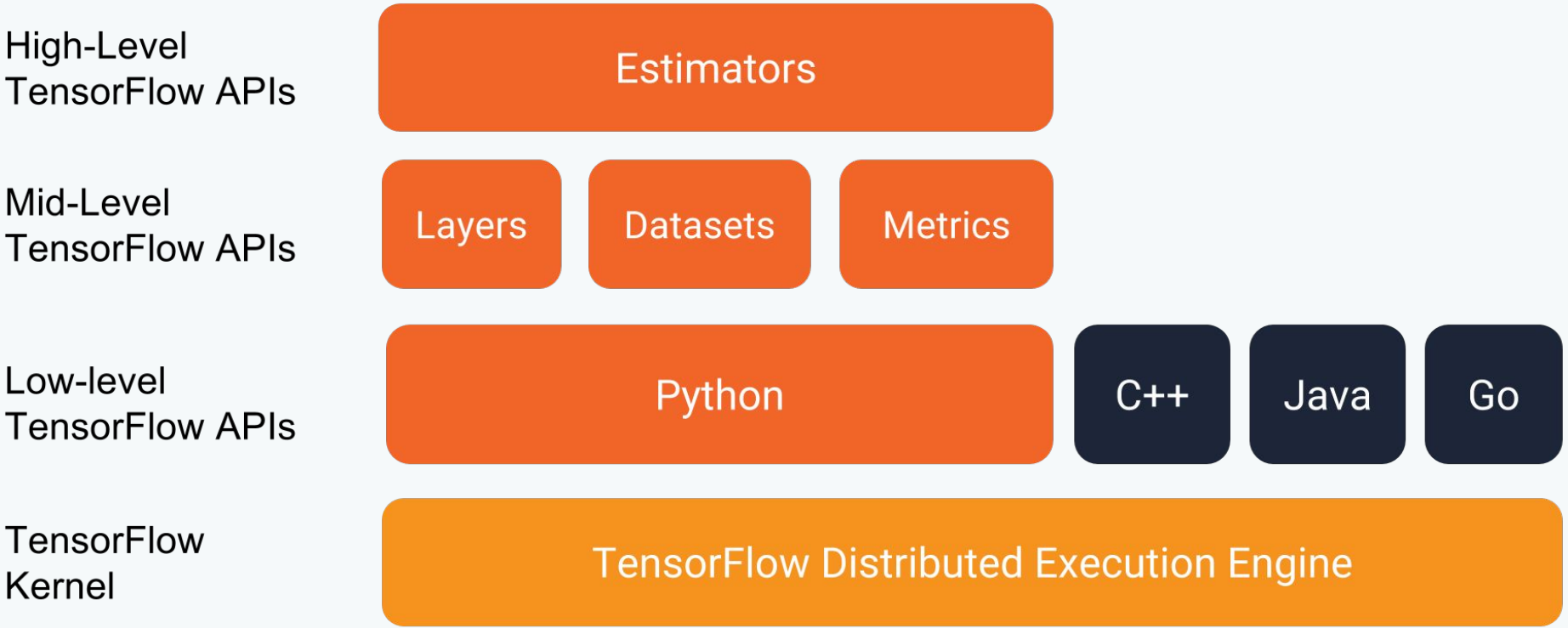

https://www.tensorflow.org/images/tensorflow_programming_environment.png

# TensorFlow Programming Stack




https://www.tensorflow.org/images/tensorflow_programming_environment.png

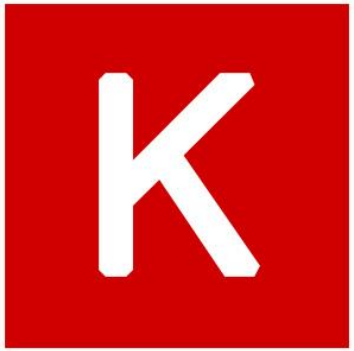
# TensorFlow Programming Stack




https://www.tensorflow.org/images/tensorflow_programming_environment.png

# 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

https://pythonclock.org/

# Python

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

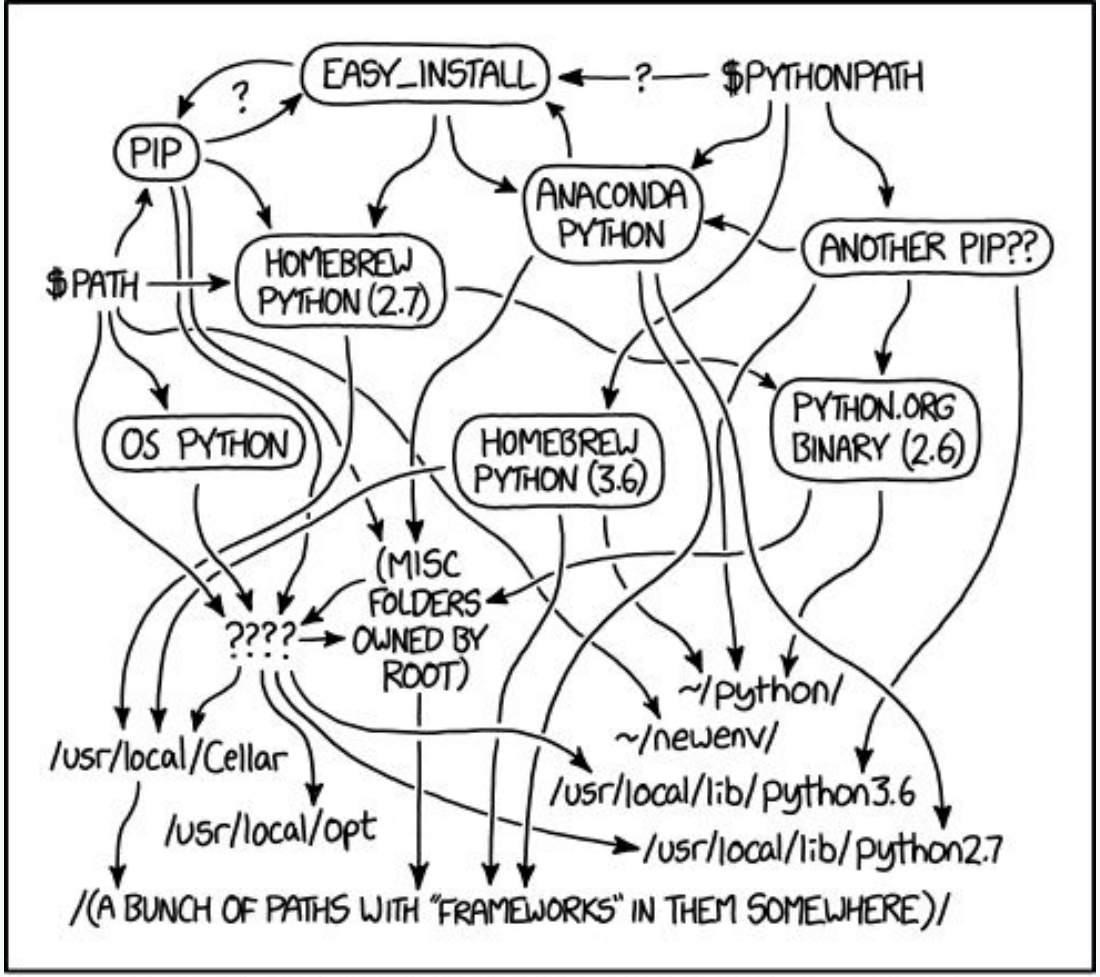
```
# Create project
mkdir project
cd project

# 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
```

Best Practices

# Python




MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

# Python

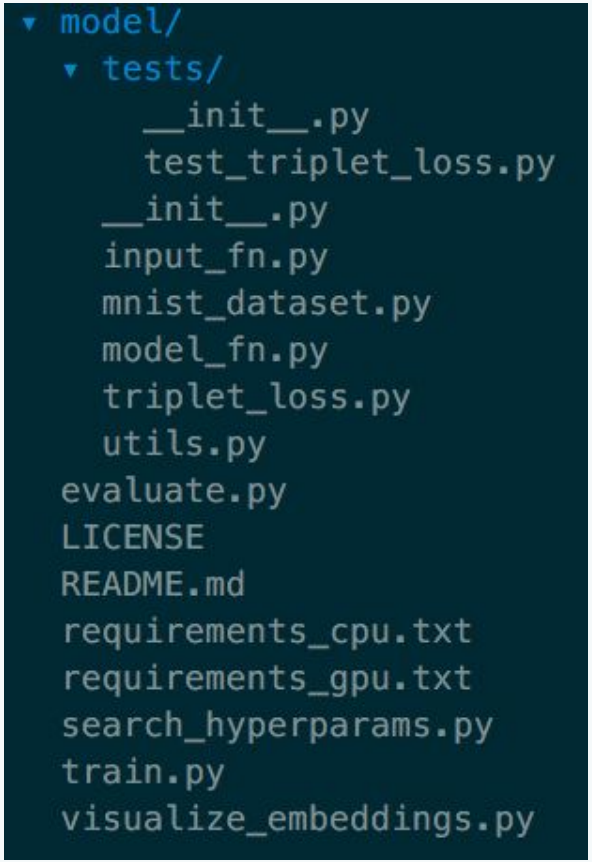
- Python is great
- Have a good install:
  http://docs.python-guide.org
- Coding style: PEP8, Pylint

```
❯ 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

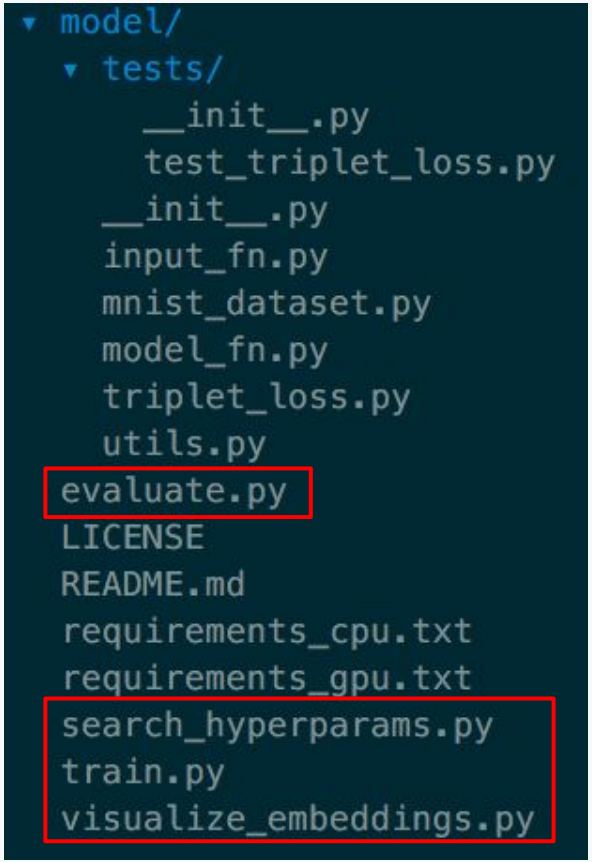
- Python is great
- Have a good install:
  http://docs.python-guide.org
- Coding style
  - PEP8
  - Pylint
- Code organization



# Python

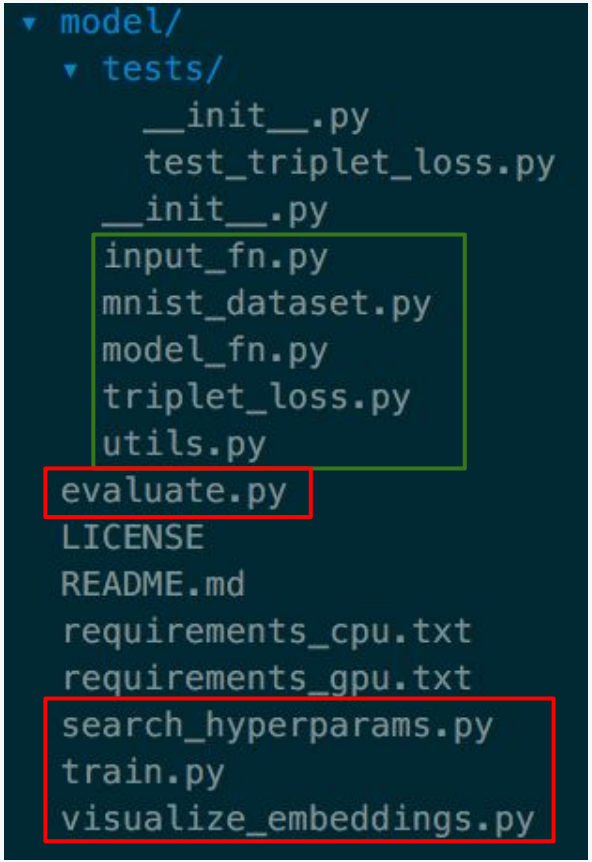
- Python is great
- Have a good install:
  http://docs.python-guide.org
- Coding style
  - PEP8
  - Pylint
- Code organization

Scripts



# Python

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




Module

Scripts

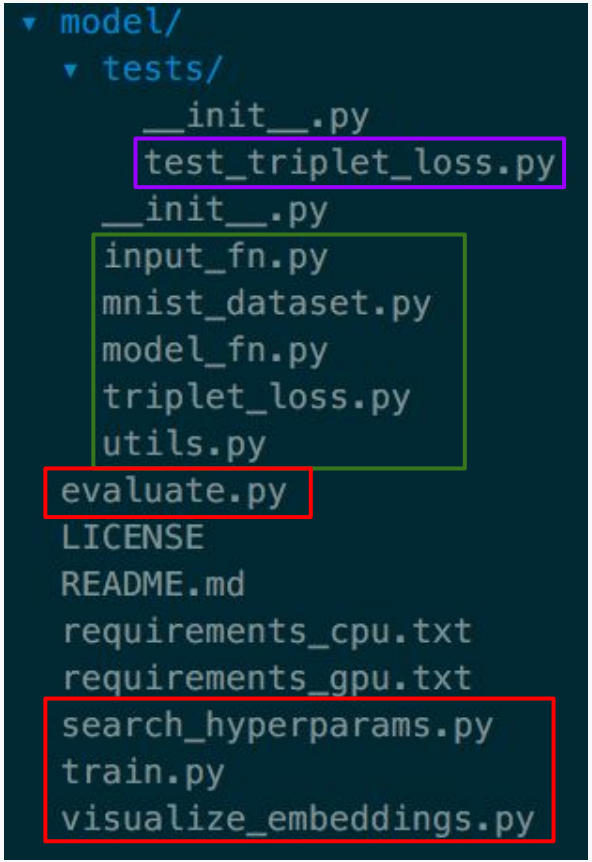
# Python

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

Tests

Module

Scripts



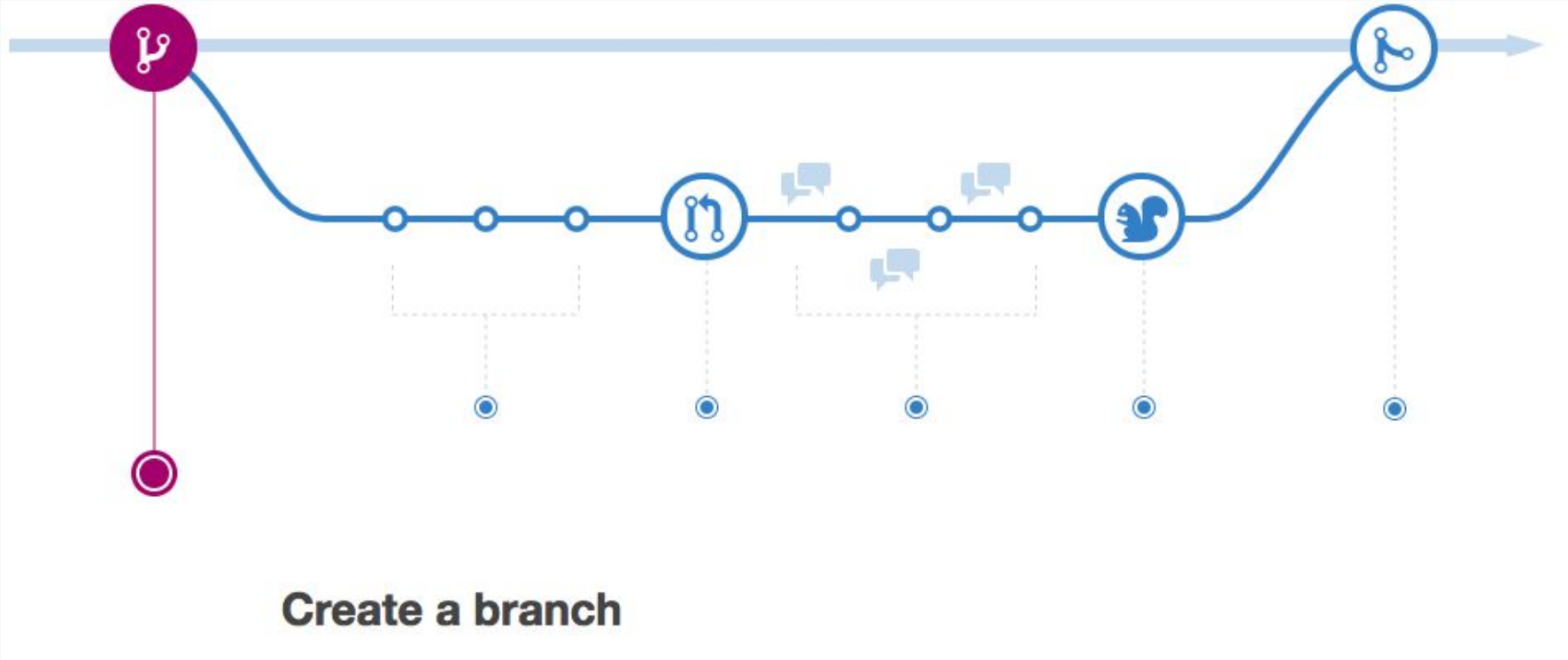
# Github

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


https://guides.github.com/activities/hello-world/

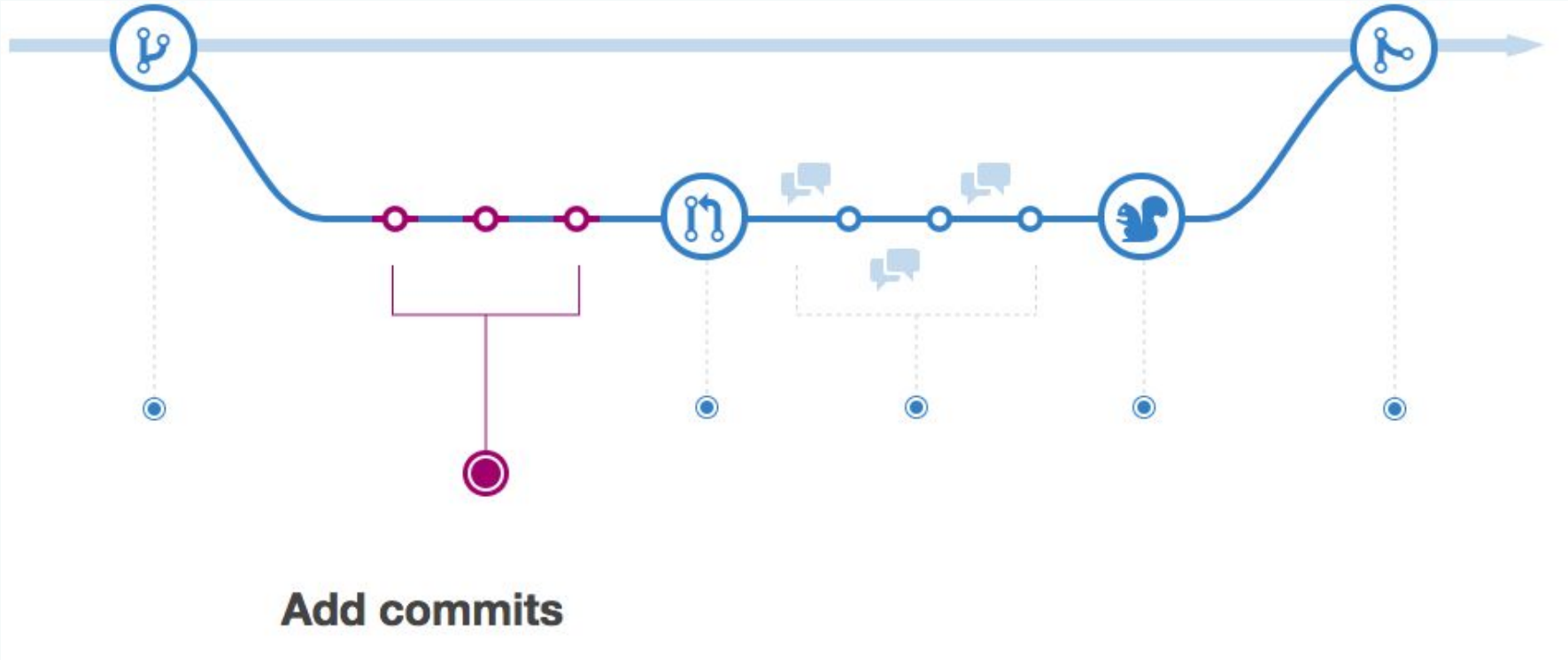
https://guides.github.com/introduction/flow/

# Github



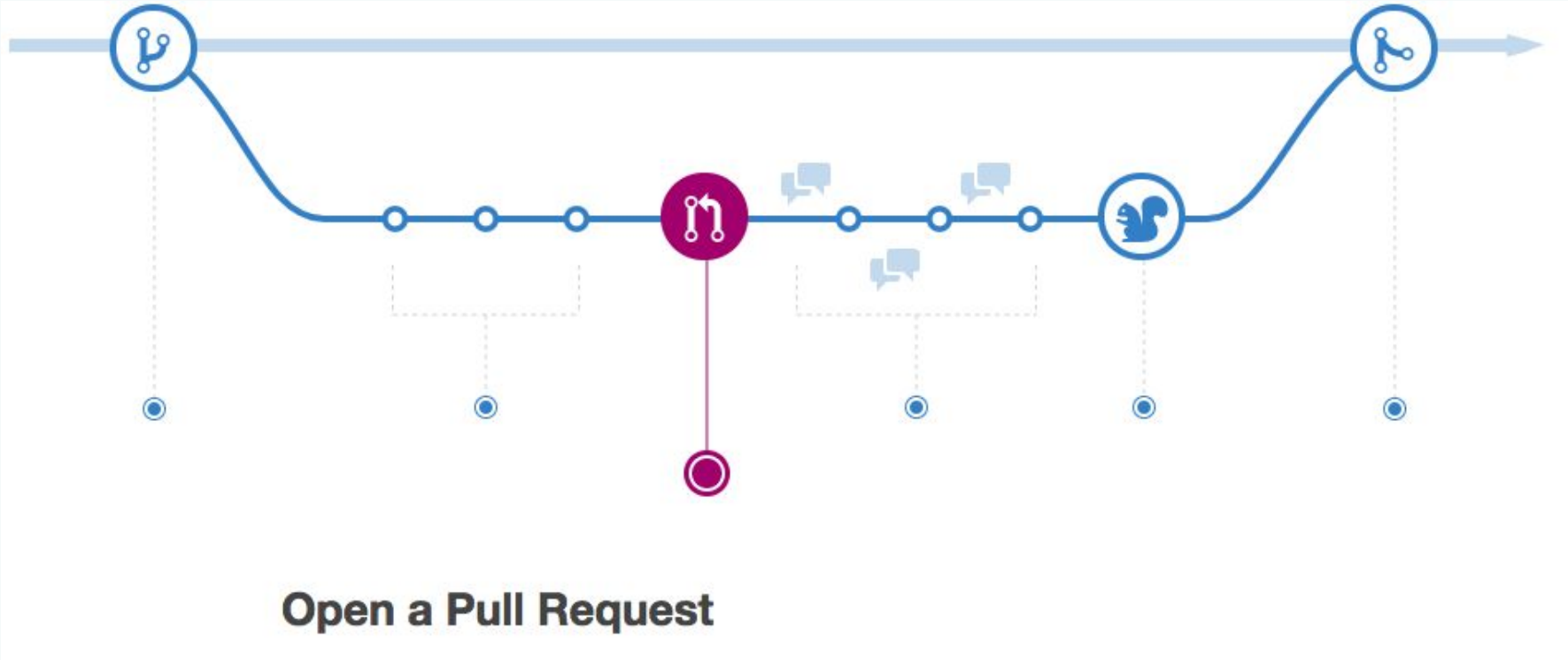

https://guides.github.com/introduction/flow/

# Github



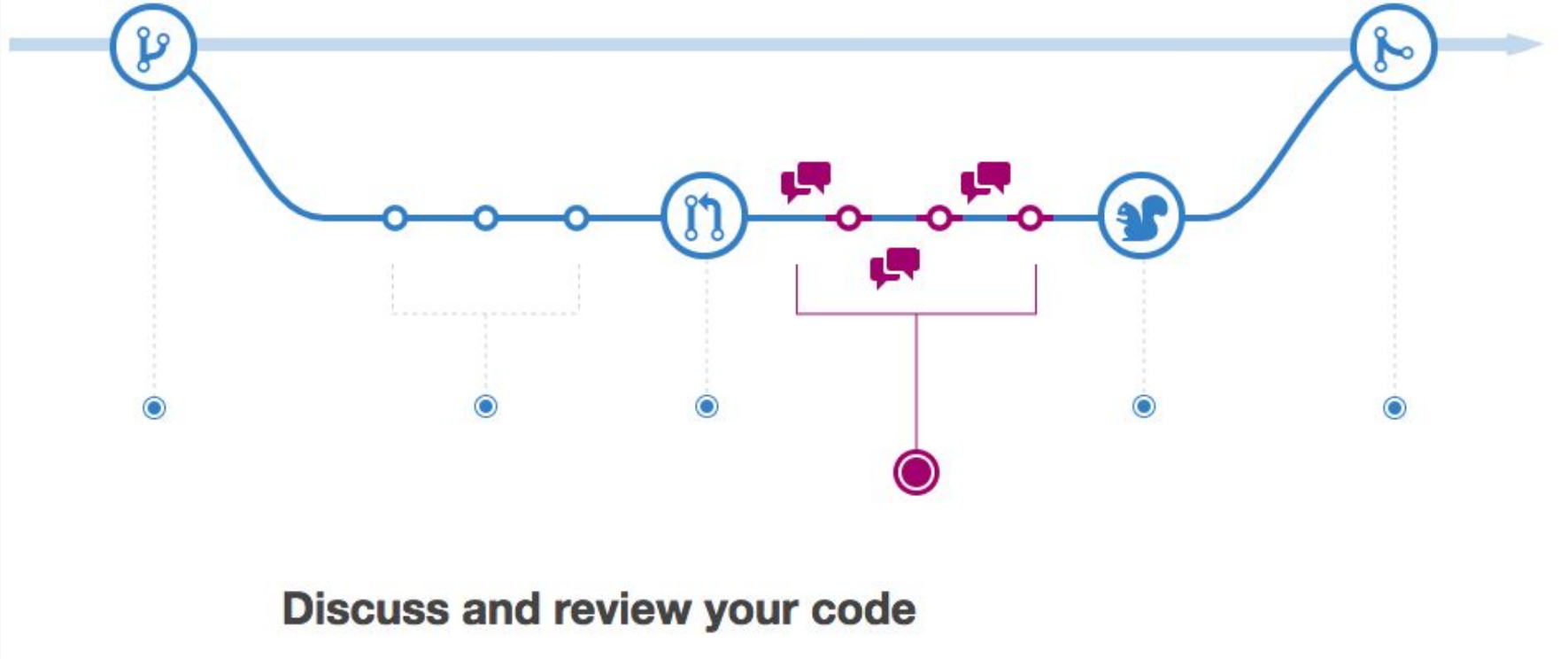

https://guides.github.com/introduction/flow/

# Github



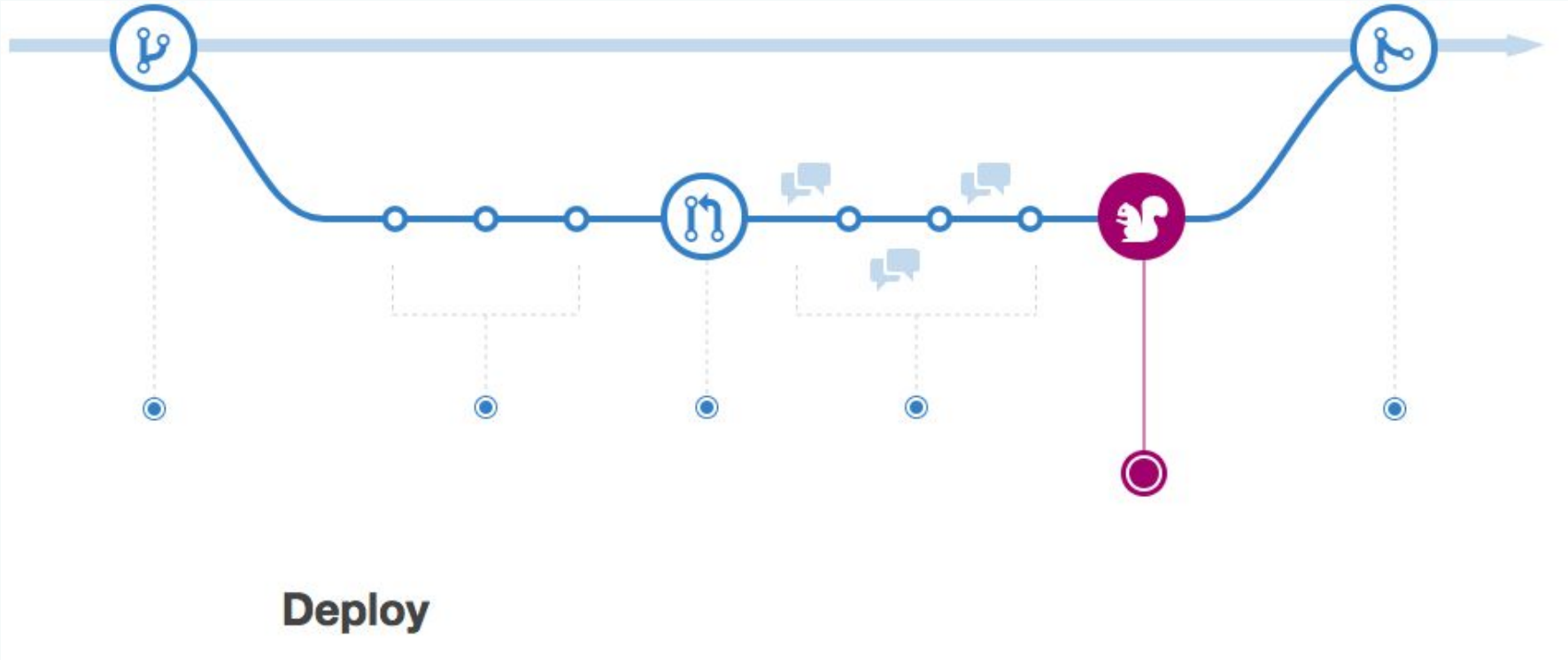

https://guides.github.com/introduction/flow/

# Github



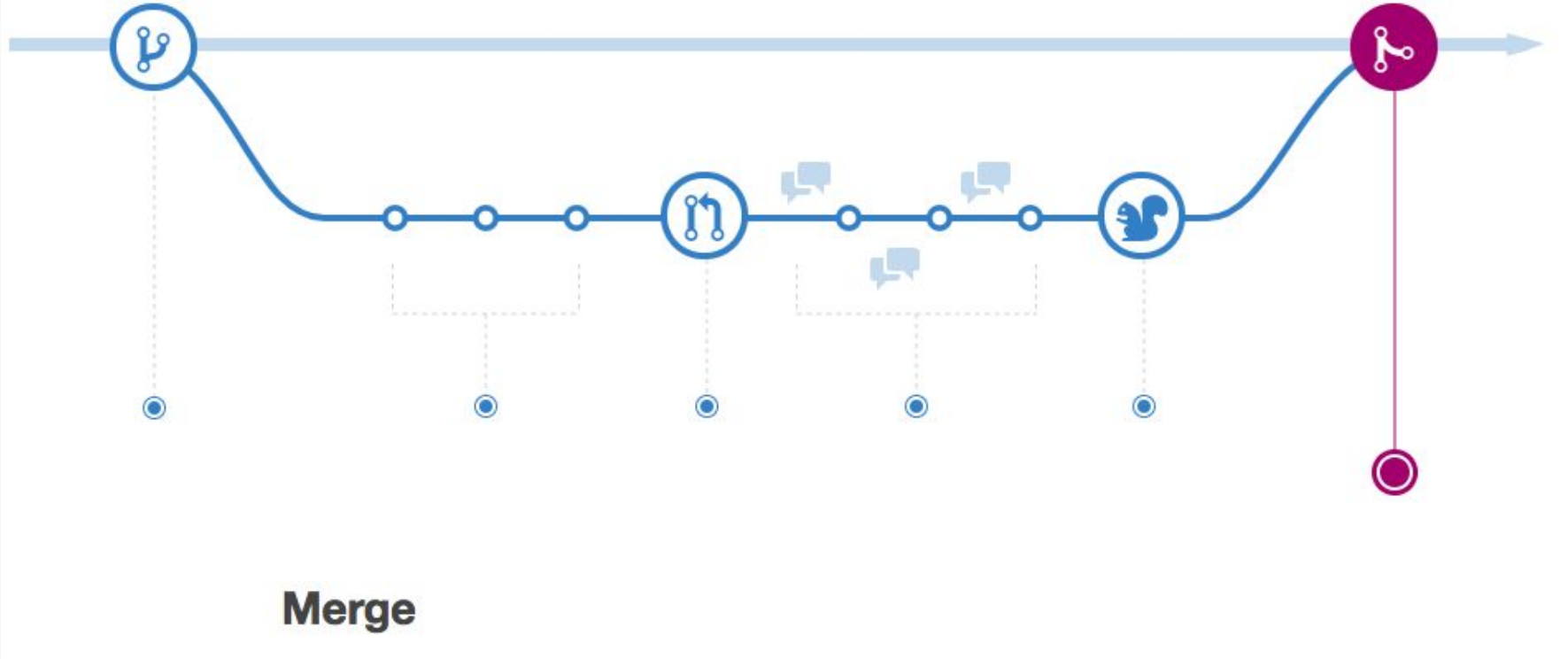

https://guides.github.com/introduction/flow/

# Github




https://guides.github.com/introduction/flow/

# Github




https://guides.github.com/introduction/flow/

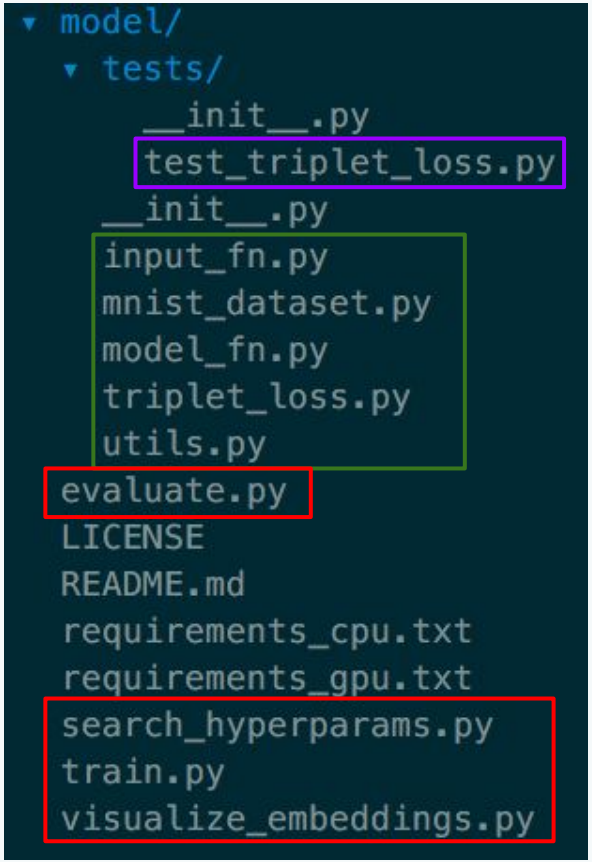
# Testing your code

- Writing tests = good practice
  - http://docs.python-guide.org/en/latest/writing/tests/

Tests

Module

Scripts



# 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 ====================
```

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

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

https://pytest.org

# 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

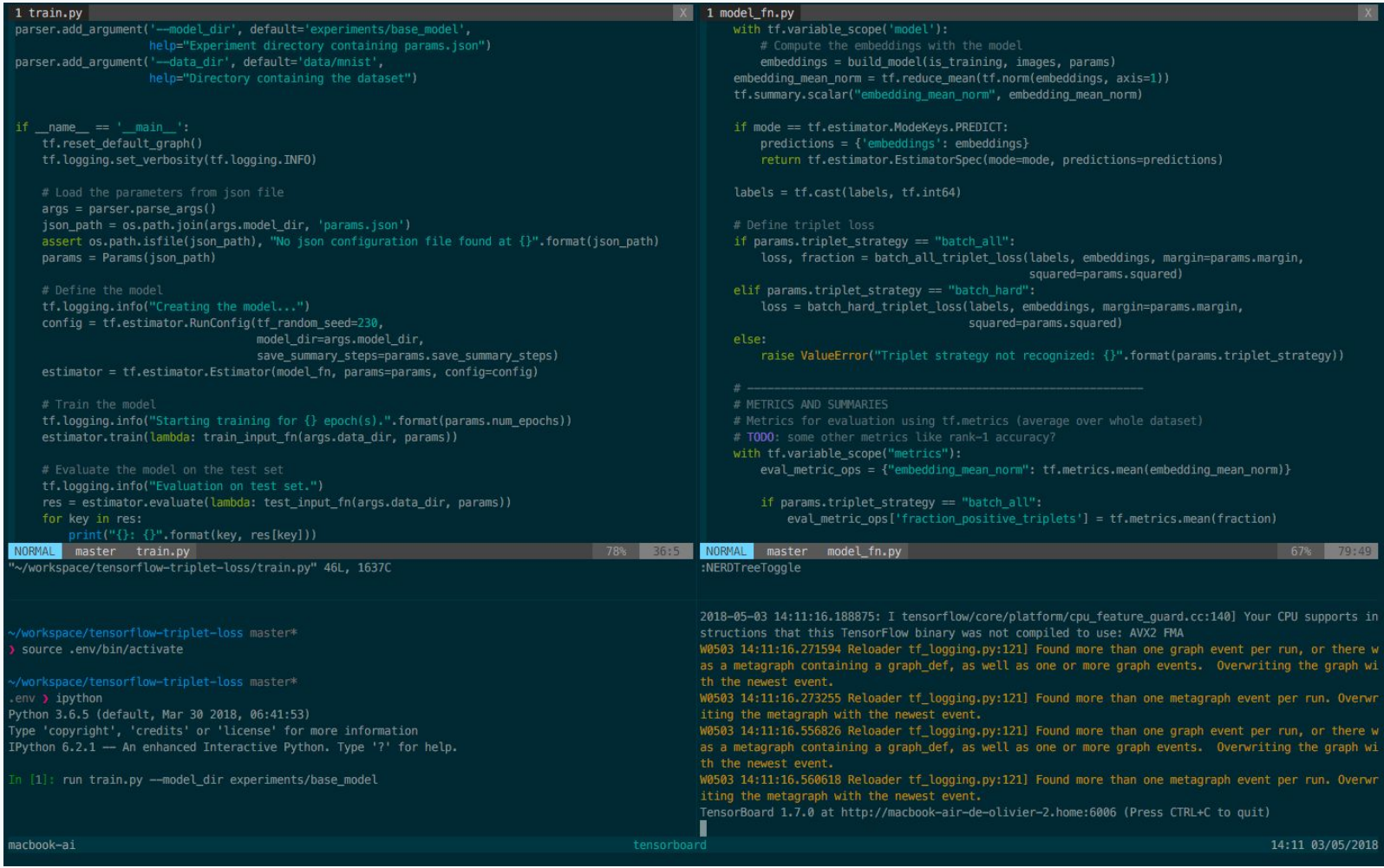
# Developing in python

- Python files
- Ipython notebooks
- Ipython in terminal

Best Practices

# Live demo

*(My work environment)*



# Daily life in deep learning

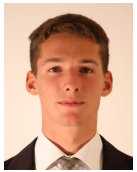
- Read papers
- Think about models on a whiteboard
- **Implement stuff**
  - Clean data, boilerplate around the model…
  - Small part for the model implementation
- Train models

# 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