

Machine Learning With TensorFlow

X433.7 (2 semester units in COMPSCI)

Instructor Alexander I. Iliev, Ph.D.

Course Content Outline

- **Machine Learning With TensorFlow®**
 - Introduction, Python - pros and cons
 - Python modules, DL packages and scientific blocks
 - Working with the shell, IPython and the editor
 - Installing the environment with core packages
 - Writing “Hello World”
 - Ecosystem, Use Cases, Competition, UsersHW1 (5pts)
- **Tensorflow and TensorBoard basics (cont.)**
 - TF 1.13+ vs. 2.0+ some differences
 - Linear algebra recap
 - Data types in Numpy and Tensorflow
 - Basic operations in Tensorflow
 - Graph models and structures with TensorboardHW2 (5pts)
- **TensorFlow operations**
 - Sessions, graphs, variables, placeholders
 - Overloaded operators, Using Aliases
 - Eager execution, Data Sets, Tensorboard
 - Distributed StrategyHW3 (5pts)
- **TF 2.0 vs. 1.X comparison**
 - Upgrading, Migrating 1.x to 2.0
 - Control statements, Reductions
 - Name scopes
- **TF 1.x vs. 2.0, Machine Learning concepts**
 - Machine Learning Models, k-Means, Linear RegressionHW4 (5pts)

Course Content Outline

- **Machine Learning**
 - Querunners
 - Logistic Regression
 - Softmax classification
 - Splitting Data and Stratification
 - Single-Layer and Multi-layer Neural Networks
 - Gradient descent and Backpropagation
 - **Neural Networks**
 - Optimizers
 - Object recognition with Convolutional Neural Network (CNN)
 - Kernels, Strides, Padding
 - Activation Functions: Sigmoid, Tanh, ReLU
 - Common layers: Conv2d, Dropout, Pooling Layers
 - CNN Overview
 - **Working with images**
 - Keras Applications and TensorFlow Hub
 - Normalization
 - Loading Images
 - Image formats and manipulation
 - Speech processing
 - **CNN Implementation**
 - Performance and Model Training
 - Recurrent Neural Network (RNN)
 - Long-Short Term Memory (LSTM)
 - Word Vector Embedding
 - Principal Component Analysis (PCA)
 - Digital Audio and Signal Processing
 - **CPU/GPU usage + Project Presentations**
 - Project Presentation
- Midterm (30pts)
- Project proposal due (4pts)
- Discussion of student proposals
- Final Project (36pts)

Course Content Outline

- **Methods of Instruction**

Lectures and in-class discussions will be the main tools of instruction. Homework problems will help students absorb the material and get to practice in their free time. Since this is a more specialized course that naturally delves into specific topics a midterm exam will be held to assess the performance of the students midway through the course.

- **Credit Requirements**

Students must complete all homework assignments and pass all exams. They also must receive a passing grade on the final exam to receive a passing grade in the course.

- **Course Grade Weighting**

1. Class participation: 10% -
2. Homework: 20% -
3. Midterm 30% -
4. Final Project 40% -

Course Content Outline

- **Optional Reading for the Course**

Title: Python for Data Analysis

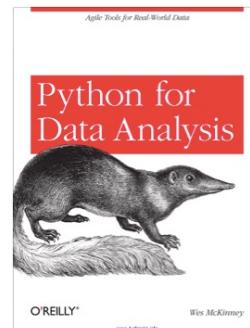
ISBN: 9781491957660

Author(s): Wes McKinney

Publisher: O'Reilly Media

Edition Number: 2nd edition

Publication Date: October, 2017



Title: Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow

Paperback : 856 pages

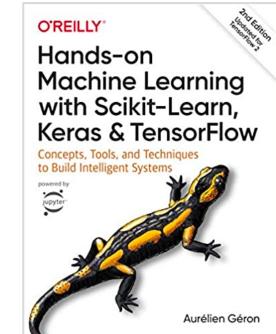
ISBN-10 : 1492032646

ISBN-13 : 978-1492032649

Publisher : O'Reilly Media

Edition: 2nd edition

Publication Date: October 15, 2019
kit-Learn, Keras, and TensorFlow



(optional reading)

Title: Data Mining

ISBN: 978-0-12-804291-5

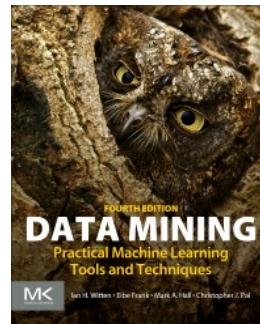
Author(s): Ian H. Witten, Eibe Frank, Mark A. Hall

Publisher: Elsevier

Edition Number: 4th edition

Publication Date: 2016

[Free Online copy for 3rd edition \(ResearchGate\)](#)



Title: Pattern Classification

ISBN: 111858600X, 9781118586006

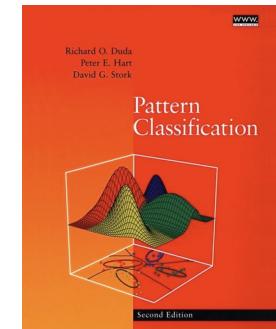
Author(s): Richard O. Duda

Peter E. Hart, David G. Stork

Publisher: John Wiley & Sons, 2012

Edition: 2nd edition

Publication Date: 2012



Data Mining

Class 1 ...

Python, TensorFlow, Versions, Installation, Use Cases and general concepts ...

Data

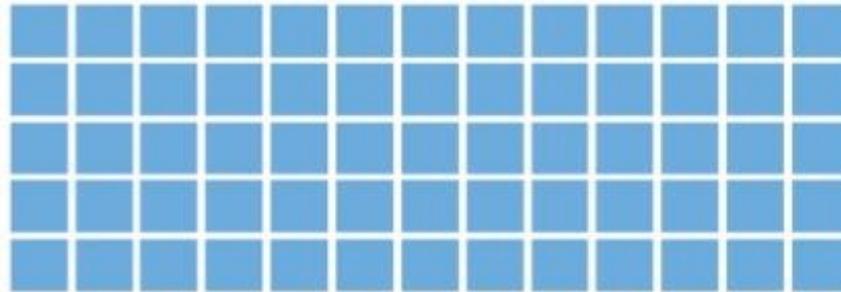
- The world's digital content and media is **growing rapidly** at a never stopping rate



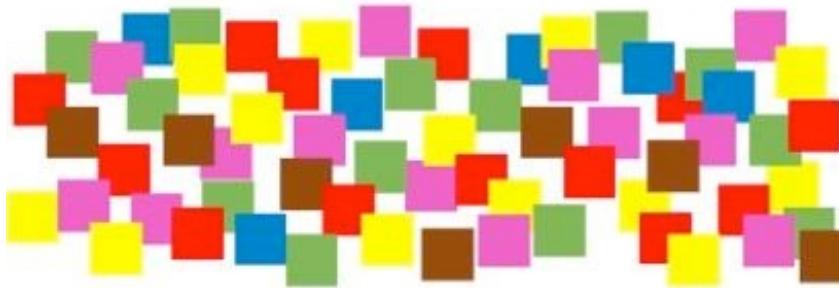
90% of the data in the world today has been created in the last two years alone, at 2.5 quintillion bytes of data a day

report from IBM Marketing Cloud in 2017

Data



Structured Data

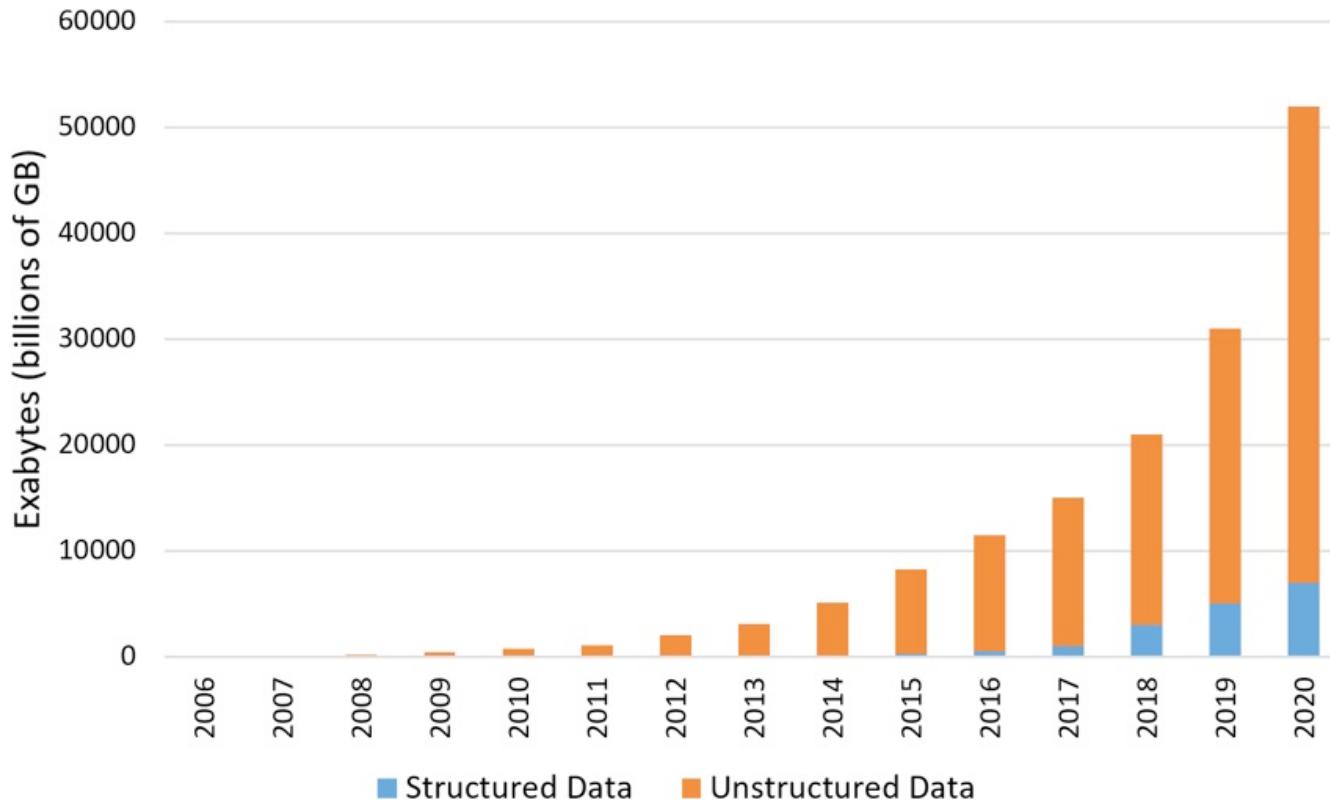


Unstructured Data

Graphical representations illustrate the difference between
structured and unstructured data

Data

The Cambrian Explosion...of Data



Data

Multiplying Factor	SI Prefix	Scientific Notation	Name
1 000 000 000 000 000 000 000 000	Yotta (Y)	10^{24}	1 septillion
1 000 000 000 000 000 000 000	Zetta (Z)	10^{21}	1 sextillion
1 000 000 000 000 000 000	Exa (E)	10^{18}	1 quintillion
1 000 000 000 000 000	Peta (P)	10^{15}	1 quadrillion
1 000 000 000 000	Tera (T)	10^{12}	1 trillion
1 000 000 000	Giga (G)	10^9	1 billion
1 000 000	Mega (M)	10^6	1 million
1 000	kilo (k)	10^3	1 thousand
0 001	milli (m)	10^{-3}	1 thousandth
0 000 001	micro (u)	10^{-6}	1 millionth
0 000 000 001	nano (n)	10^{-9}	1 billionth
0 000 000 000 001	pico (p)	10^{-12}	1 trillionth
0 000 000 000 000 001	femto (f)	10^{-15}	1 quadrillionth
0 000 000 000 000 000 001	atto (a)	10^{-18}	1 quintillionth
0 000 000 000 000 000 000 001	zepto (z)	10^{-21}	1 sextillionth
0 000 000 000 000 000 000 000 001	yocto (y)	10^{-24}	1 septillionth

Metric prefixes defined at the 19th General Conference on Weights and Measures in 1991

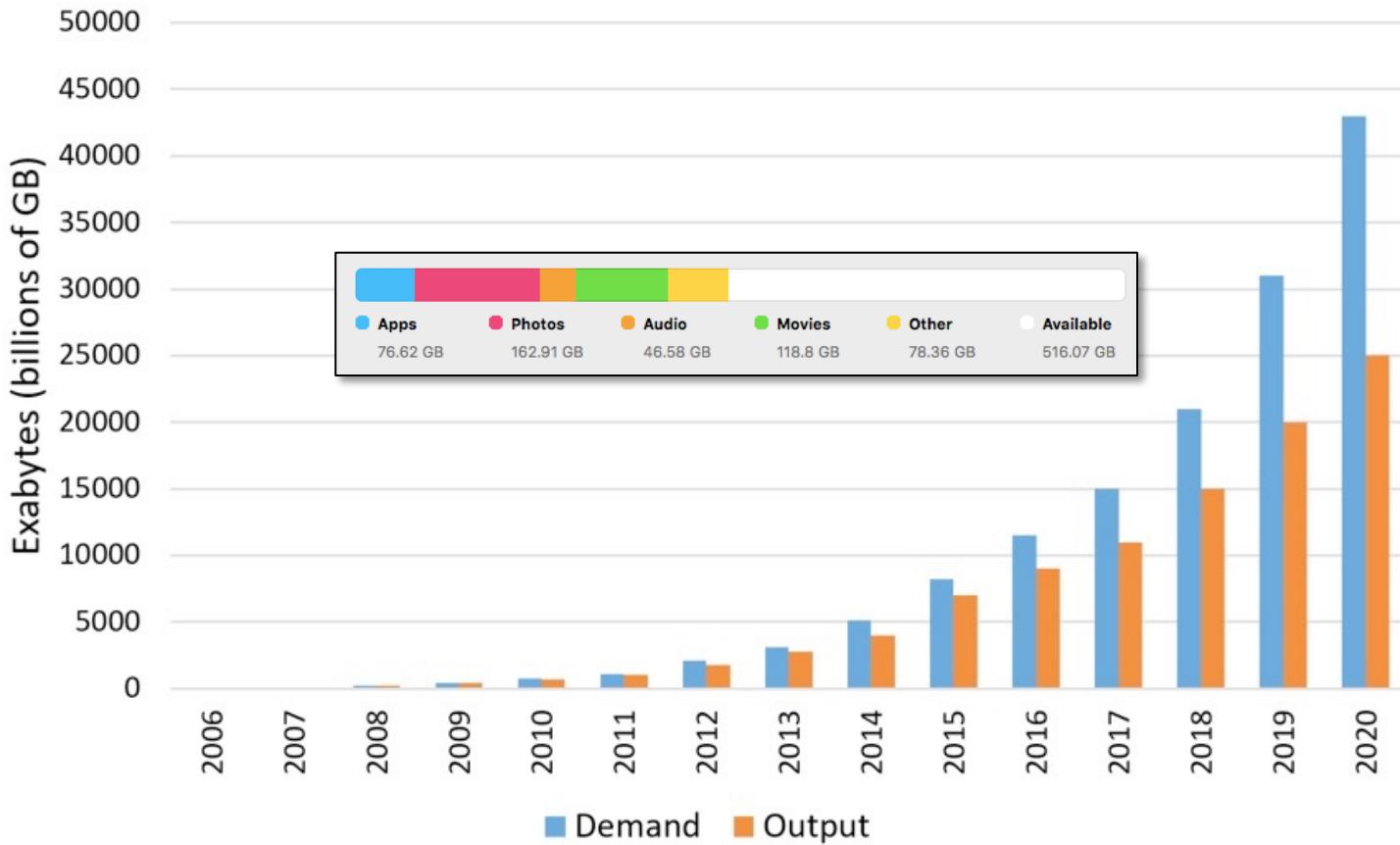
Data

Multiplying Factor	SI Prefix	Scientific Notation	Name
1 208 925 819 614 629 174 706 176	Yottabytes	2^{80}	1 septillion
1 180 591 620 717 411 303 424	Zettabytes	2^{70}	1 sextillion
1 152 921 504 606 846 976	Exabytes	2^{60}	1 quintillion
1 125 899 906 842 624	Petabytes	2^{50}	1 quadrillion
1 099 511 627 776	Terabytes	2^{40}	1 trillion
1 073 741 824	Gigabytes	2^{30}	1 billion
1 048 576	Megabytes	2^{20}	1 million
1 024	kilobytes	2^{10}	1 thousand

Examples of prefixes used to measure digital data with a binary system

Data

Storage Supply & Demand



Storage supply and demand growth over two decades

Machine Learning With TensorFlow

- Artificial Intelligence concept
 - It is a concept conceived in the mid 50s with the intention to construct complex machines (computers) that possessed the same characteristics of human intelligence
 - The main idea is to create technologies able to perform specific tasks better than humans can
 - Some of the first intelligent robots emerged in movies such as Star Wars – C-3PO
 - Artificial intelligence (AI) is already part of our everyday lives
 - Some examples of AI involve: image classification, face recognition, voice recognition, speaker recognition, emotion recognition, etc.

Machine Learning With TensorFlow

- **Data Mining** and Machine Learning concepts
 - In data mining, the data is stored electronically and the search is automated (or semi-automated)
 - It has been estimated that the amount of data stored in the world's databases doubles every 20 months ...
 - ... Thus the opportunities for data mining increase
 - **Data mining** is about solving problems by analyzing data already present in databases
 - **Data mining** finds patterns that can be analyzed to identify distinguishing characteristics

Machine Learning With TensorFlow

- Data Mining and Machine Learning concepts
 - Machine learning came directly from the early AI scientists. The algorithmic approaches over the years included: *decision tree learning, logic programming, clustering, reinforcement learning, Bayesian networks*, etc.
 - To “learn” by definition for us humans means to:
 - Obtain knowledge of something and being aware of something
 - Be informed, receive instructions, commit to memory
 - An operational definition for “learn” can be formulated like:
 - Things learn when they change their behavior in a way that makes them perform better in the future. The definition of “better” is given by us and means “performance” rather than “knowledge”

Machine Learning With TensorFlow

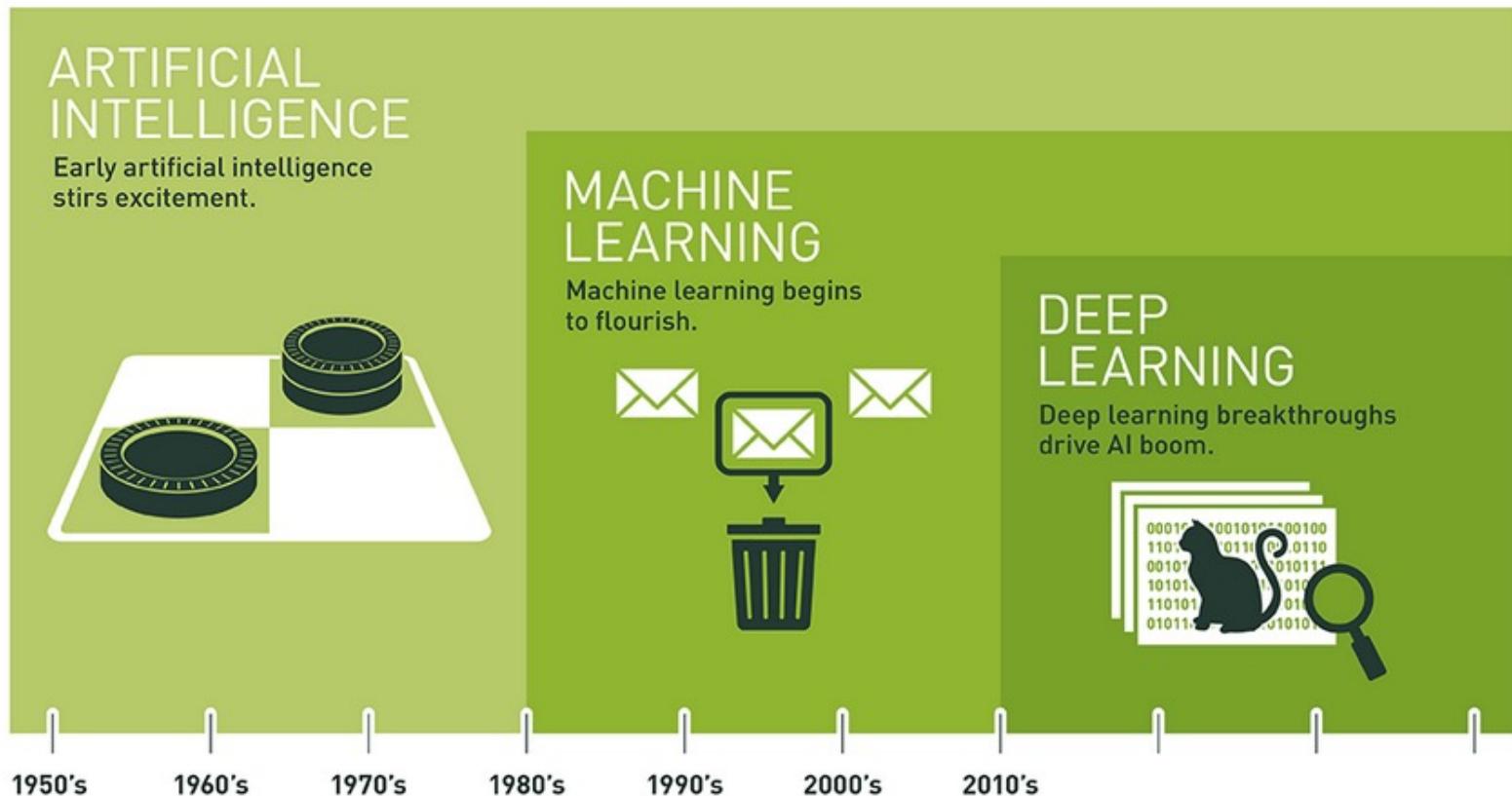
- Data Mining and Machine Learning concepts
 - Machine learning does not entail the conceptual limitations mentioned herein, and disregards any philosophical stance about what learning actually is
 - Machine learning is therefore a practical process connected to Data mining while using algorithms to parse data, learn from it, and then make a prediction
 - Performance:
 - the machine is “trained” using large amounts of data and algorithms that give it the ability to learn;
 - it is then “tested” on a new (usually unknown) set of data

Machine Learning With TensorFlow

- Deep Learning concept
 - Deep Learning is an area in Machine Learning that has been introduced with the objective of moving Machine Learning closer to its goal: Artificial Intelligence
 - Deep Learning has enabled many practical applications of Machine Learning and by extension the overall field of AI
 - Deep Learning breaks down tasks in ways that makes all kinds of machine assists seem possible
 - Areas of implementation are infinite, but some of them are: preventive healthcare, security systems, robotics, cars without drivers, better media recommendations, etc.

Machine Learning With TensorFlow

- Deep Learning concept



Machine Learning With TensorFlow

- Python environment for Deep Learning – why using it?
 - Because you can develop **proof-of-concept** solutions fairly easily and free
 - **Open-source** software that is widely spread
 - **Fast** development time
 - **Many scientific libraries**: NumPy, SciPy, Matplotlib, Theano, Scikit, Tensorflow, etc.
 - Mobile computing for **smartphone development** by using Kivy for Android and iOS
 - **Web-site** incorporation by using Django
 - Try it here: <https://www.pythonanywhere.com/try-ipython/>

Machine Learning With TensorFlow

- Python – pros and cons:

Pros

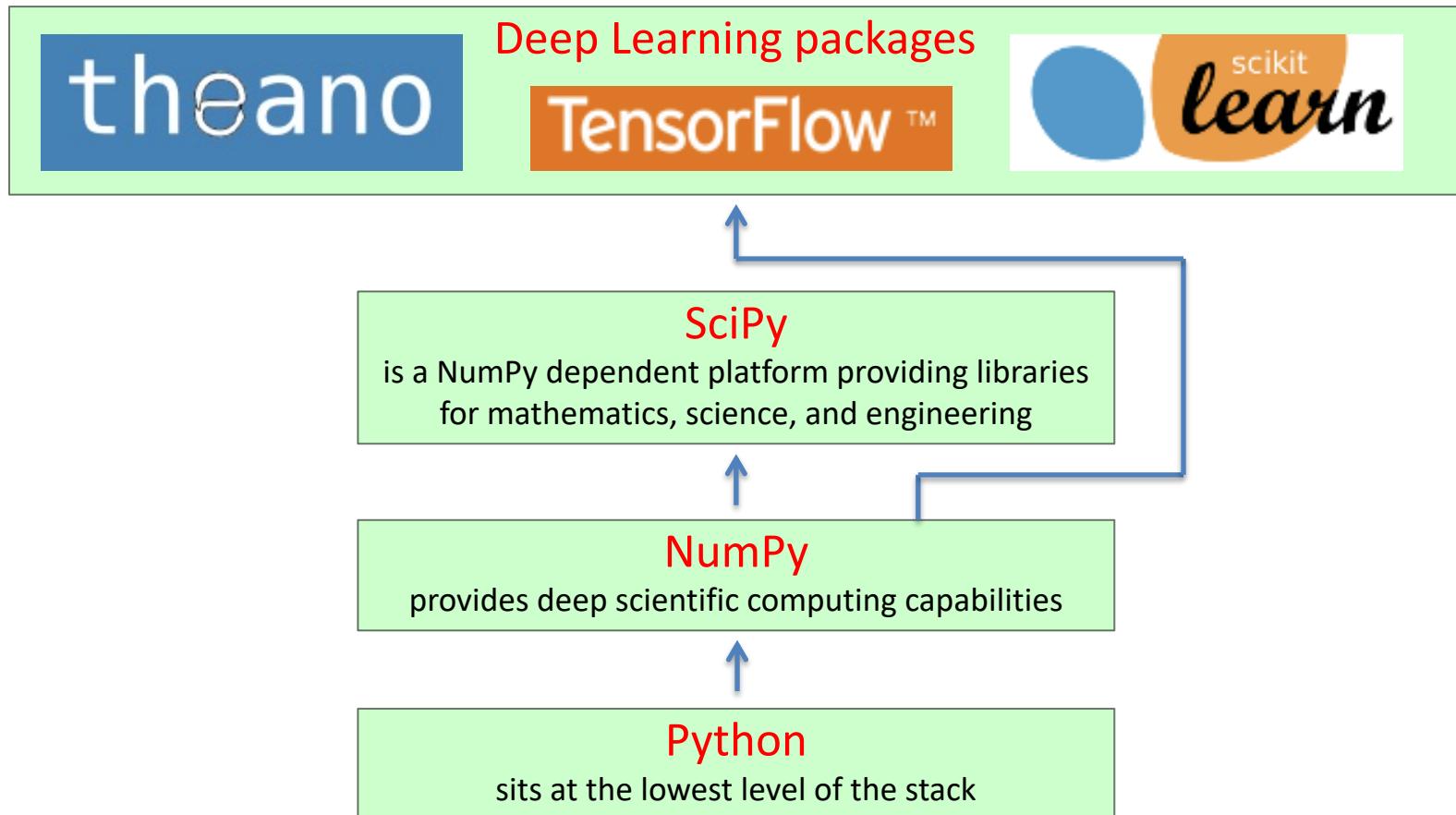
- High-level object-oriented language that is **easy to learn**
- **Good readability** of the code
- **Quick development time**
- Rich scientific libraries and many others like web server management, serial port access, etc.
- **Easy to share** and communicate results
- **Easy to read**, manipulate and process data
- A single environment for many tasks. No need to learn another language
- **Free and open-source** software that is widely spread

Cons

- **Speed of execution** since it is higher level language
- **Mobile computing** for smartphone development (unless you use Kivy: Linux, Windows, OS X, Android and iOS)
- It is **not included in Web browsers**, because it is hard to secure
- Design flaws because it is **dynamically typed language** and some errors show at runtime such as syntactic errors (mistyping variable names)
- Python can be both **interpreted (.py)** and **compiled (.pyc)**
- **Documentation** isn't at the level of PHP or Java

Machine Learning With TensorFlow

- Basic stack using NumPy, SciPy and Deep Learning in Python:



Machine Learning With TensorFlow

- Python + NumPy + SciPy + ?
 - ... is like Matlab + toolboxes + DL processing



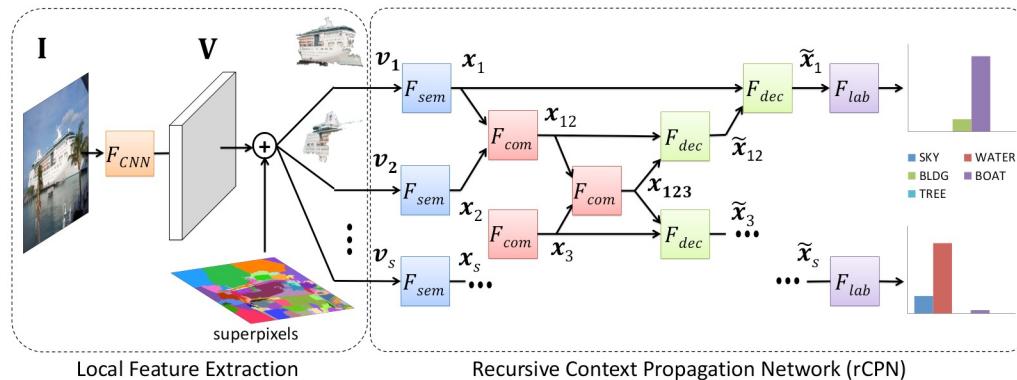
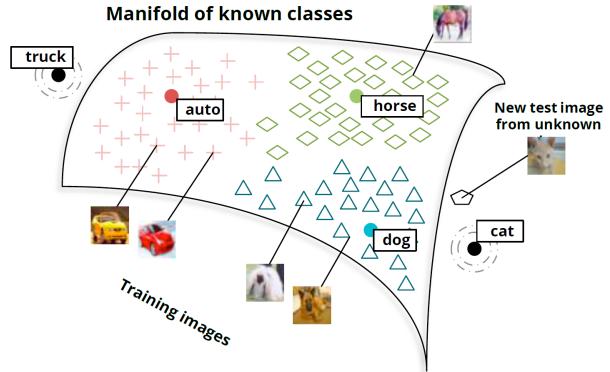
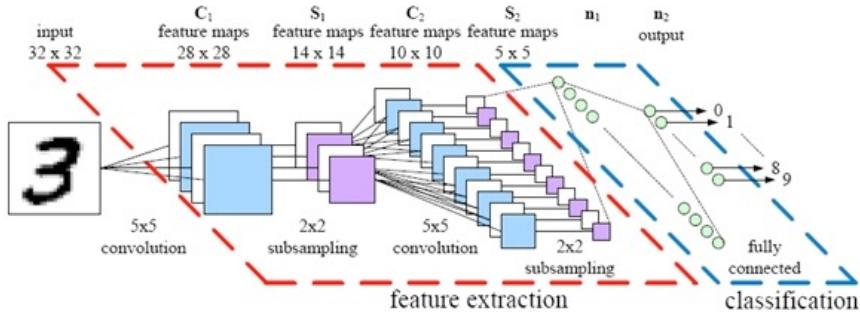
Machine Learning With TensorFlow

- Python + NumPy + SciPy + Theano + Scikit-Learn + Tensorflow
 - ... is like Matlab + toolboxes + DL processing



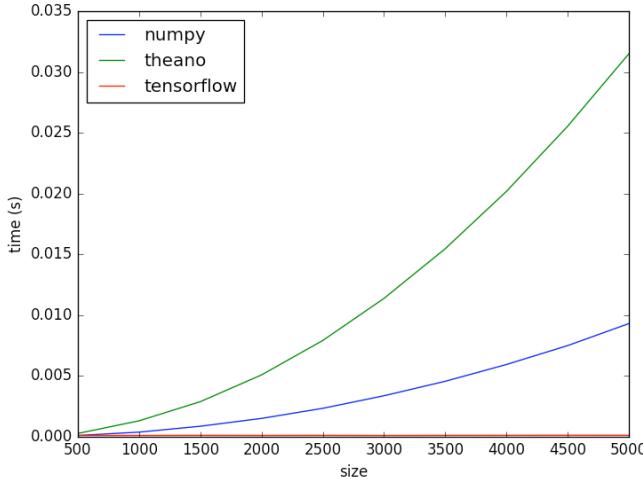
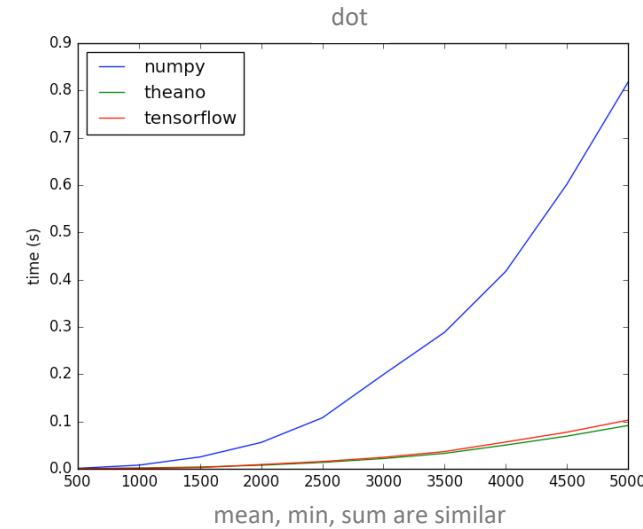
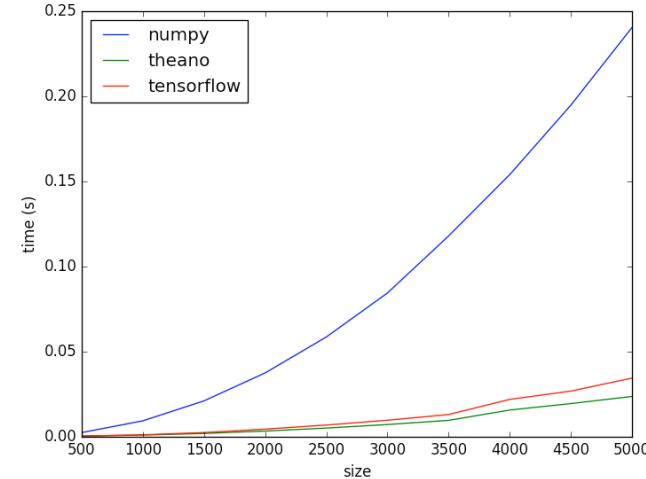
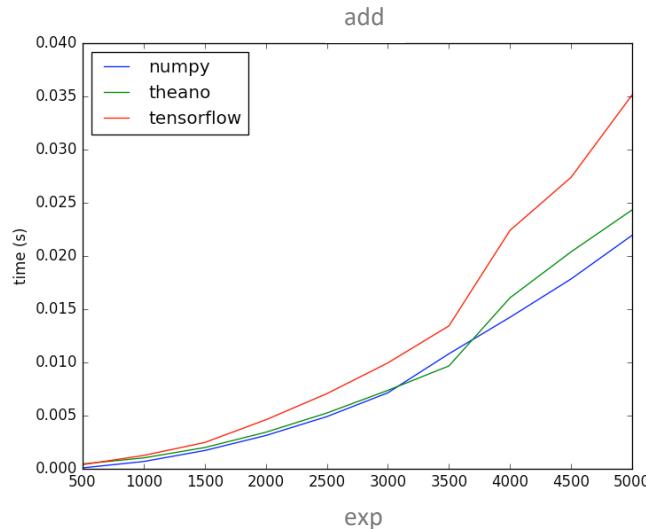
Machine Learning With TensorFlow

- Python + NumPy + SciPy + Theano + Scikit-Learn + Tensorflow
 - ... is like Matlab + toolboxes + DL processing



Machine Learning With TensorFlow

- Testing performance for NumPy + Theano + Tensorflow



Machine Learning With TensorFlow

Top Deep Learning Projects

A list of popular github projects related to deep learning (ranked by stars).



Project Name	Stars	Description
TensorFlow	29622	Computation using data flow graphs for scalable machine learning.
Caffe	11799	Caffe: a fast open framework for deep learning. CNN & image processing, written in C++
Theano	4286	Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. It can use GPUs and perform efficient symbolic differentiation.
Leaf	4281	Open Machine Intelligence Framework for Hackers.
Char RNN	3820	Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch.
Neural Talk	3694	NeuralTalk is a Python+numpy project for learning Multimodal Recurrent Neural Networks that describe images with sentences.
deeplearning4j	3673	Deep Learning for Java, Scala & Clojure on Hadoop, Spark.
TFLearn	3368	Deep learning library featuring a higher-level API for TensorFlow.
TensorFlow Playground	3352	Play with neural networks!
Scikit Neural Net	849	Deep neural networks without the learning cliff! Classifiers and regressors compatible with scikit-learn.

Source: <https://github.com/aymericdamien/TopDeepLearning>

Machine Learning With TensorFlow

Top Deep Learning Projects

According to medium.com

View 1+ more



TensorFlow



Caffe



Microsoft
Cognitive
Toolkit



PyTorch



Apache
MXNet



Chainer



Keras

8 Best Deep Learning Frameworks for Data Science enthusiasts

- 1. TensorFlow. TensorFlow is arguably one of the **best deep learning** frameworks and has been adopted by several giants such as Airbus, Twitter, IBM, and others mainly due to its highly flexible system architecture. ...
- 2. Caffe. ...
- 3. Microsoft Cognitive Toolkit/CNTK. ...
- 4. Torch/PyTorch. ...
- 5. MXNet. ...
- 6. Chainer. ...
- 7. Keras. ...
- 8. Deeplearning4j.

Apr 5, 2018

Source: <https://medium.com/the-mission/8-best-deep-learning-frameworks-for-data-science-enthusiasts-d72714157761>

Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow

Theano logo, the word "theano" in white lowercase letters on a blue rectangular background.

- Theano is a **Python library** that lets you to define, optimize, and evaluate mathematical expressions, especially ones with multi-dimensional arrays (`numpy.ndarray`)
- Speed is **rivaling C implementations** for problems involving large amounts of data
- It can surpass C on a CPU by many orders of magnitude by **taking advantage of GPUs**
- Theano combines aspects of a **computer algebra system** (CAS) with **optimized compiler**
- It can also **generate customized C code** for many mathematical operations
- Theano can minimize the amount of compilation/analysis overhead

Source: <http://deeplearning.net/software/theano/introduction.html>

Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow



- Theano's compiler applies many optimizations of varying complexity that include, but are not limited to:
 - use of GPU for computations
 - constant folding - evaluating **constant** expressions at compile time (not runtime)
 - merging of similar sub-graphs, to avoid redundant calculation
 - arithmetic simplification (e.g. $x*y/x \rightarrow y$, $-x \rightarrow x$)
 - inserting efficient **BLAS** operations (e.g. **GEMM**) in a variety of contexts
 - using memory aliasing to decrease calculations
 - using in place operations wherever it does not interfere with aliasing
 - loop fusion for elementwise sub-expressions
 - improvements to numerical stability (e.g. $\log(1 + \exp(x))$ and $\log(\sum_i \exp(x[i]))$)
 - for a complete list, see <http://deeplearning.net/software/theano/optimizations.html#optimizations>

Source: <http://deeplearning.net/software/theano/introduction.html>

Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow

Theano logo: the word "theano" in white lowercase letters on a blue rectangular background.

- **Theano** is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving **multi-dimensional arrays efficiently**. It features:
 - **tight integration with NumPy** – Use `numpy.ndarray` in Theano-compiled functions
 - **transparent use of a GPU** – Perform data-intensive **calculations up to 140x faster** than with CPU.(float32 only)
 - **efficient symbolic differentiation** – Theano does your derivatives for function with one or many inputs
 - **speed and stability optimizations** – Get the right answer for $\log(1+x)$ even when x is really tiny
 - **dynamic C code generation** – Evaluate expressions faster
 - **extensive unit-testing and self-verification** – Detect and diagnose many types of errors

Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow



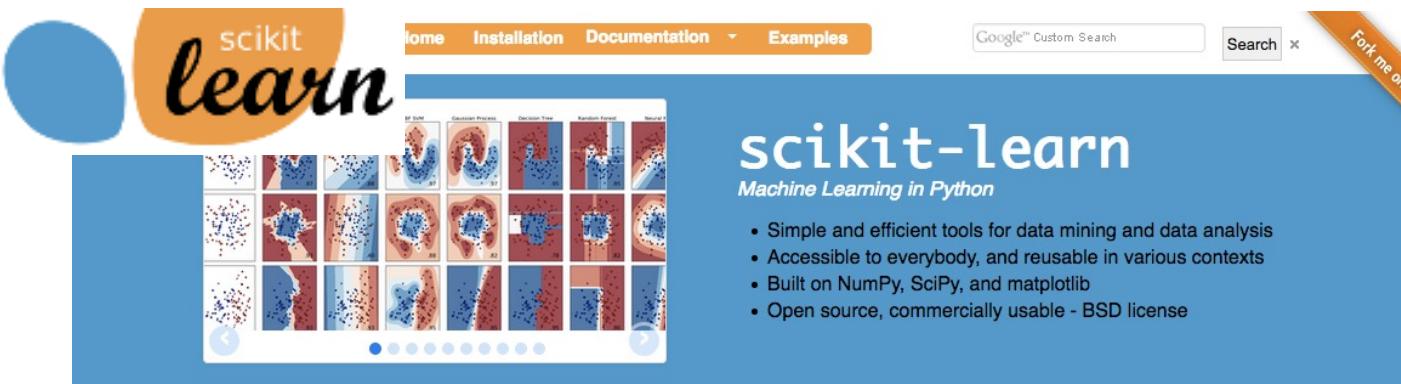
- Theano and TensorFlow are competing heavily for which one is the best platform
- Theano outperforms TensorFlow on a **single GPU**, but TensorFlow outperforms Theano for **parallel execution** on multiple GPUs
- Theano has better documentation in general than TensorFlow, although that changes quickly
- Theano has a **native Windows** support
- Both use Numpy arrays
- TensorFlow is more elegant conceptually cleaner than Theano

Machine Learning

- Working with Theano, Scikit-Learn and TensorFlow
 - **Classification:** Identifying to which category an object belongs to
 - **Regression:** Predicting a continuous-valued attribute associated with an object
 - **Clustering:** Automatic grouping of similar objects into sets
 - **Dimensionality reduction:** Reducing the number of random variables
 - **Model selection:** Comparing, validating and choosing parameters and models
 - **Preprocessing:** Feature extraction and normalization

Machine Learning

- Working with Theano, **Scikit-Learn** and TensorFlow



Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

— Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation,

Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

— Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics.

— Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

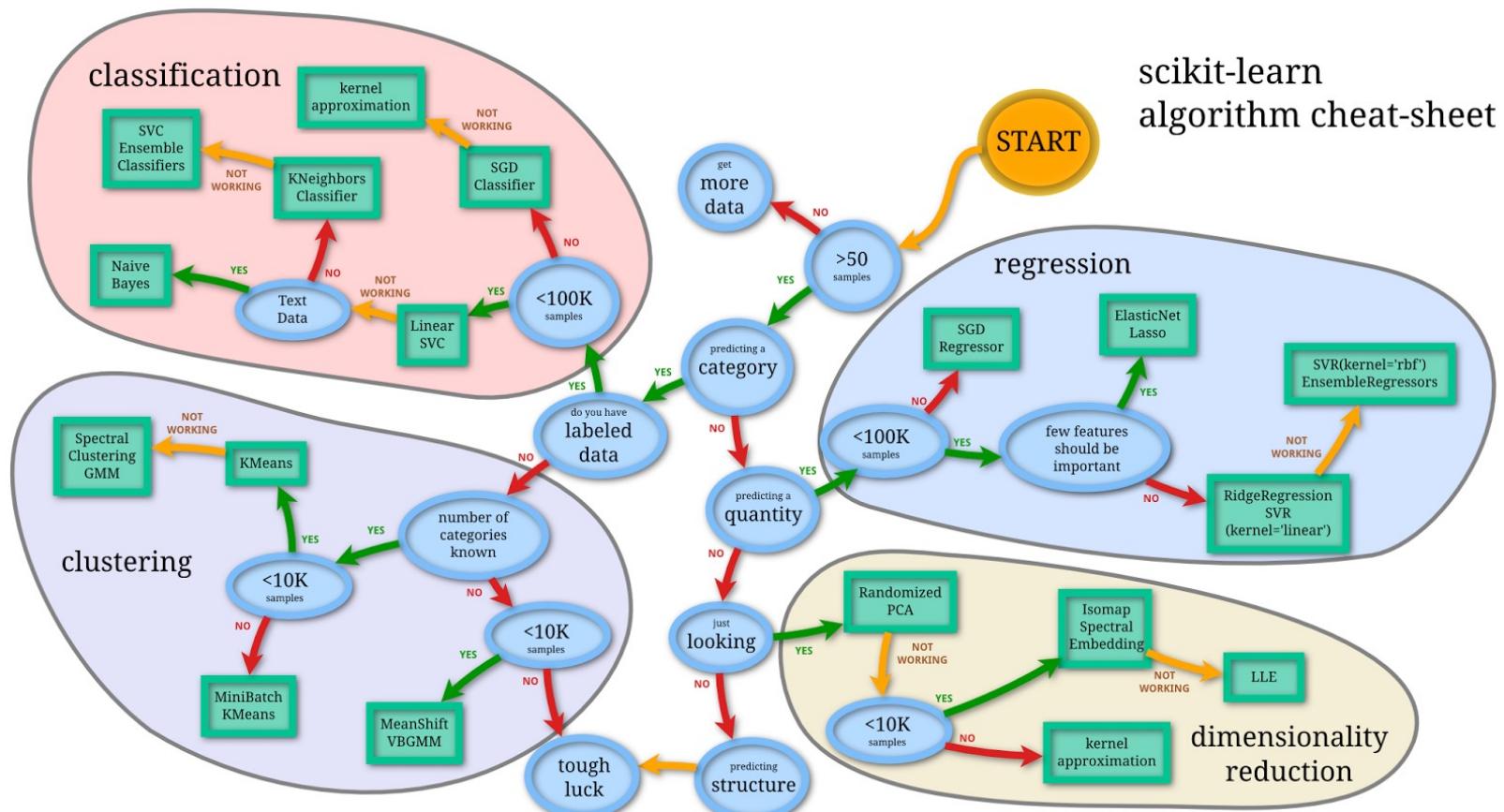
Modules: preprocessing, feature extraction.

— Examples

Source: <http://scikit-learn.org/stable/>

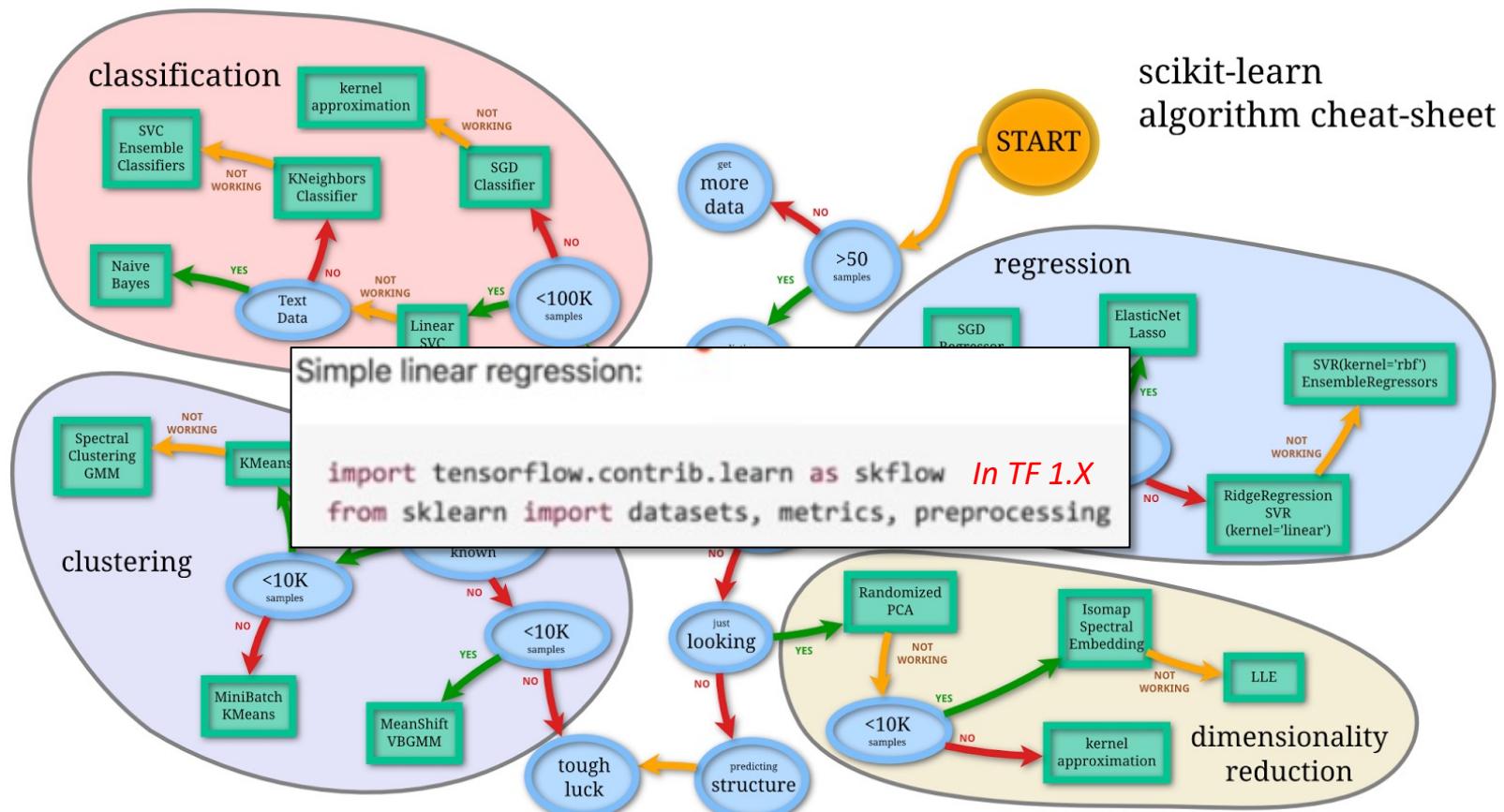
Machine Learning

- Working with Theano, Scikit-Learn and TensorFlow



Machine Learning

- Working with Theano, Scikit-Learn and TensorFlow



Tutorial 1

Instructor Alexander I. Iliev, Ph.D.

Installation

- TensorFlow is compatible with 4 operating Systems :
 - a) Windows OS
 - b) Mac OS
 - c) Ubuntu OS
 - d) Raspbian OS

TensorFlow Case Studies



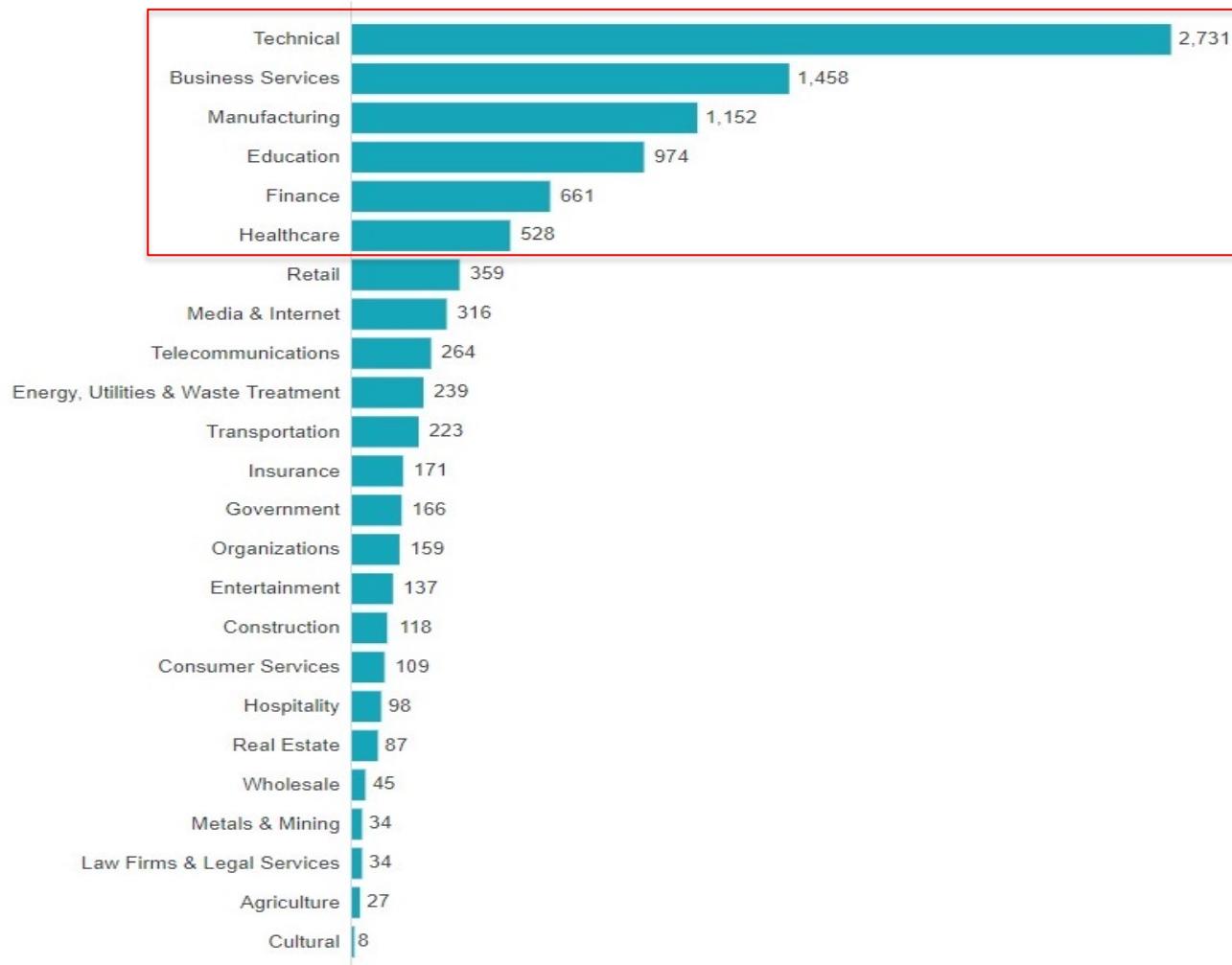
TensorFlow Case Studies

Lets have a quick look at:

- Who started using TensorFlow and Why?
- What are the different companies that are working with TensorFlow?
- How did they get benefited from it?

Companies using TensorFlow

- There are different types of companies that benefited from TensorFlow (creating quite a stir in various fields too)



Few companies that benefitted from using TensorFlow

- TensorFlow being an open package, it became the main paradigm that is being used for many solutions
- Some of the top companies that benefitted from TF are:
 - Airbnb
 - GE HealthCare
 - Airbus
 - Qualcomm

Few companies that benefitted from using TensorFlow

1)

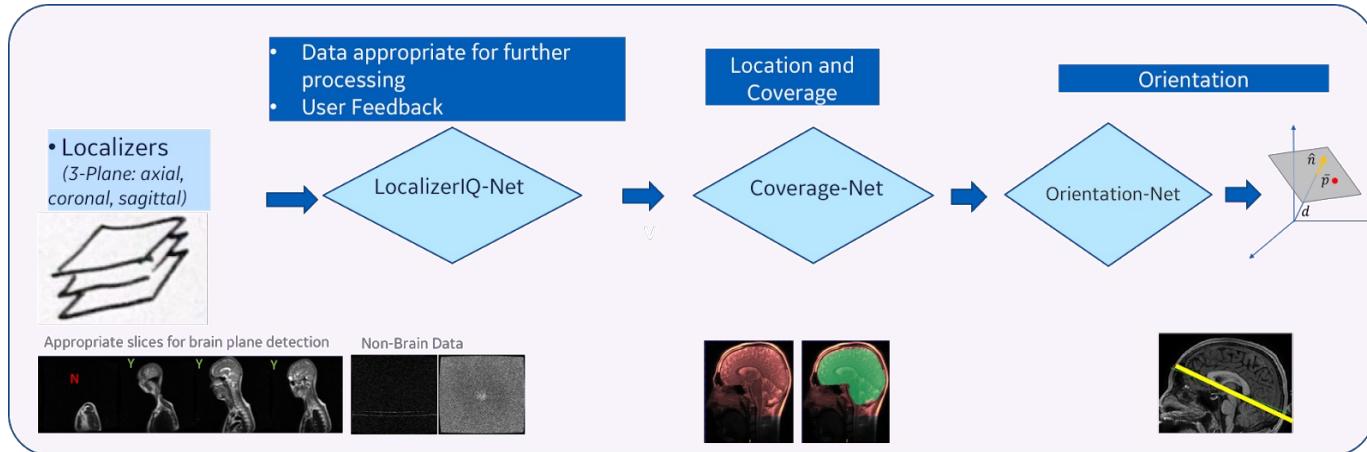


- Initially the team used to sort the images manually which consumed lots of time and energy
- A smarter way to over come such issues was to shift towards a better technology
- With TensorFlow the efficiency of their business was rapidly growing and this in turn gave the company a smoother experience and better results

Few companies that benefitted from using TensorFlow

2) GE Healthcare

- They used TensorFlow for Intelligent Scanning Using Deep Learning for MRI
- The images now have better clarity for better diagnosis



- This Company is working wonders already in the section of brain. They are looking forward to expand further more to knee and spine as well.

Few companies that benefitted from using TensorFlow

3) AIRBUS DEFENCE & SPACE

- Airbus Defence and Space uses an algorithm to identify satellite images which is sometimes very difficult to determine, even for the human eye, whether an area on a satellite image is cloud or snow
- But it has an error rate of 11%, which has prompted Airbus to test the deep-learning platform TensorFlow
- TensorFlow analyses a large number of images, looks for recurring patterns, and uses this to learn how to identify objects by itself
- In the TensorFlow test phase, the error rate has improved to just 3%

Few companies that benefitted from using TensorFlow

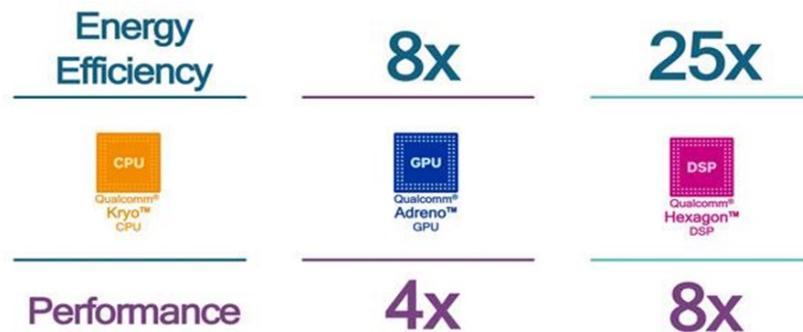
4) Qualcomm

- TensorFlow was designed to run on processing units inside of processors
- Snapdragon processors integrate a CPU, GPU and many technologies including Digital Signal Processor (DSP)
- The companies DSP architecture is designed to process certain audio and video features more quickly and at lower power than a CPU or GPU, which is why it's ideal to exploit its advantages

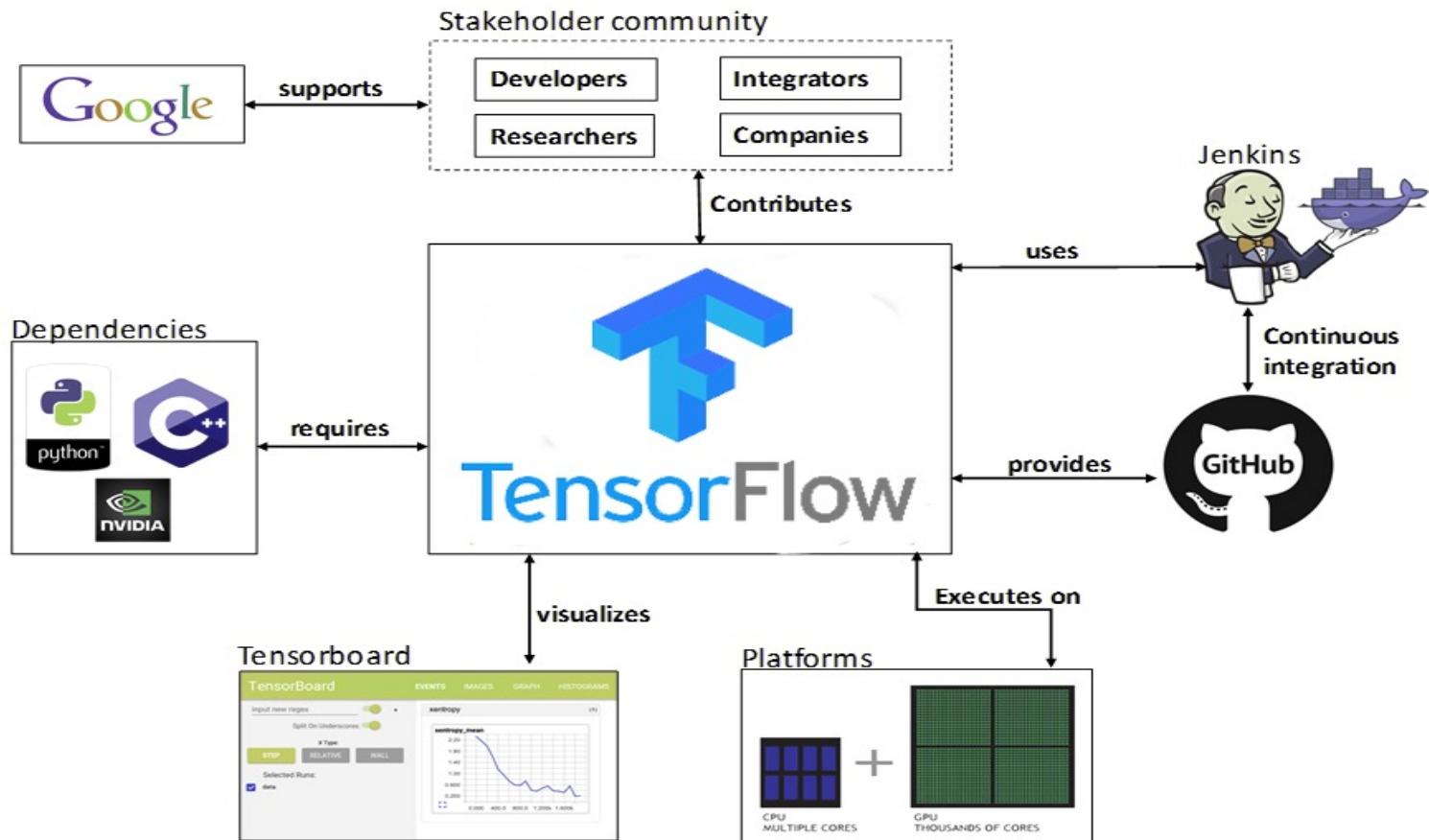
Few companies that benefitted from using TensorFlow

4) Qualcomm

- TensorFlow will run on the DSP so apps can run faster and more efficiently
- Qualcomm Technologies also offers **support for TensorFlow on Snapdragon** processors via the Qualcomm Snapdragon Neural Processing Engine SDK*
- The chart below illustrates the **benefits of apps optimized for the DSP**:



TensorFlow Ecosystem



TensorFlow Applications

- Tensorflow runs on a variety of platforms and the installation is Linux-only and more tedious than CPU-only installation.
- The applications go beyond deep learning to support other forms of machine learning:
 - reinforcement learning,
 - any goal-oriented tasks like winning video games or
 - helping a robot navigate an uneven landscape
- allows you to explore: sentiment analysis, google translate, text analysis and best of all image recognition



Machine Learning

- Working with Theano, Scikit-Learn and **TensorFlow**



- TensorFlow was originally created by Google as an **internal machine learning tool**
- An implementation of it was open-sourced under the Apache 2.0 **License in November 2015**
- TensorFlow is an interface for numerical computation as described in the TensorFlow white paper, and **Google still maintains its own internal implementation** of it
- Google is constantly **pushing internal improvements** to the public repository
- The open-source release contains the **same capabilities as Google's internal version**

Machine Learning

- Working with Theano, Scikit-Learn and **TensorFlow**



- TensorFlow™ is an **open-source** software library for numerical computation using data flow graphs
- TensorFlow was developed by researchers and engineers working on the **Google Brain Team** within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research
- **Nodes** in the graph represent **mathematical operations**, while **the graph edges** represent the multidimensional data arrays (**tensors**) communicated between them
- The **flexible architecture** allows you to deploy computation to one or more **CPUs** or **GPUs** in a desktop, server, or mobile device with a single API

Source: <https://www.tensorflow.org/>

Machine Learning

- Working with Theano, Scikit-Learn and **TensorFlow**



- TensorFlow's API is **most similar to Theano's**
- TensorFlow does contain a package, "learn" (AKA "Scikit Flow"), that emulates the one-line **modeling functionality of Scikit-Learn**
- TensorFlow provides an extensive suite of functions and classes that allow users to define **models from scratch mathematically**
- This allows users with the appropriate technical background to create customized, flexible models quickly and intuitively
- **TensorFlow is the best library to use for both Research and Production because it scales across multiple GPUs better than Theano does**

Why TensorFlow?

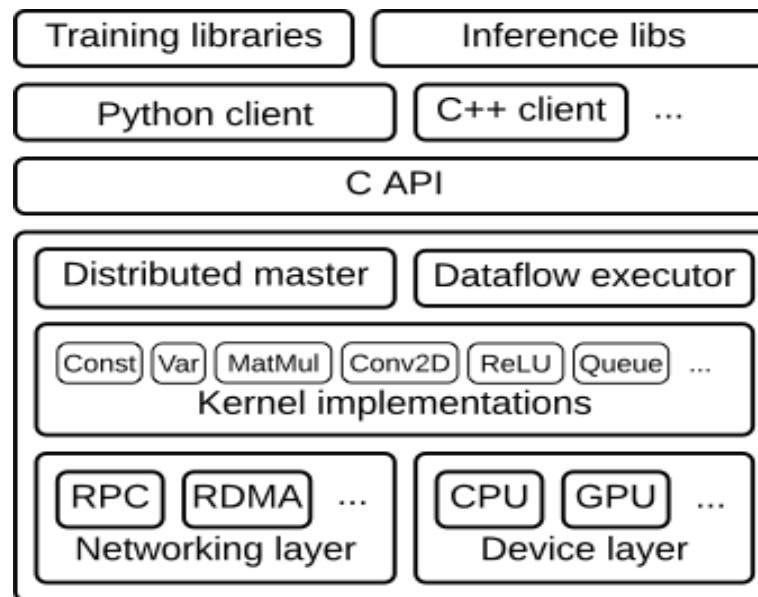
- TensorFlow provides **excellent functionalities and services** when compared to other popular deep learning frameworks.
- These **high-level operations** are essential for carrying out **complex parallel computations** and for building **advanced neural network** models.
- TensorFlow is a **low-level library** which provides **more flexibility**.
- Thus you can define your own functionalities or services for your models

Why TensorFlow?

- Distribute the training time of a neural network model over many servers to **reduce the training time**.
- TensorFlow's API is a complete package which is easier to use and read, plus provides **helpful operators**, **debugging** and **monitoring tools**, and **deployment** features.
- TensorFlow can run on multiple CPUs and GPUs.
- TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

TensorFlow Architecture

- The TensorFlow runtime is a **cross-platform library**. The figure below illustrates its general architecture.



Machine Learning With TensorFlow

- Distributions and Dependencies:



- Distributions: Python 2.6+, 3.3+
- Dependencies: Numpy 1.6.1+, Scipy 0.9+



- Distributions: Python 2.6+, 3.3+
- Dependencies: Numpy 1.7.1+, Scipy 0.11+

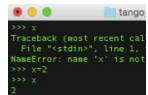


- Distributions: Python 2.7, 3.4, 3.5, 3.6, 3.7, 3.8
- Dependencies: Numpy, bazel, six, wheel

Machine Learning With TensorFlow

- Choosing your environment:

- Using the **shell** (primarily for Linux and OSX users):



- Obtaining Python with core packages:

<https://www.python.org/downloads/>

- Obtaining the NumPy & SciPy libraries:

<http://www.scipy.org/scipylib/download.html>

- GUI environments using an interactive editor and debuggers for Python:



- **Pyzo** – a free open-source environment based on Python:

<http://www.pyzo.org/downloads.html>

- **Anaconda** – Anaconda is the leading open data science platform powered by Python:

<https://www.continuum.io/>

- **Canopy** – Python based environment providing core scientific analytic and scientific Python packages, for scientific development and visualization:

<https://store.enthought.com/>

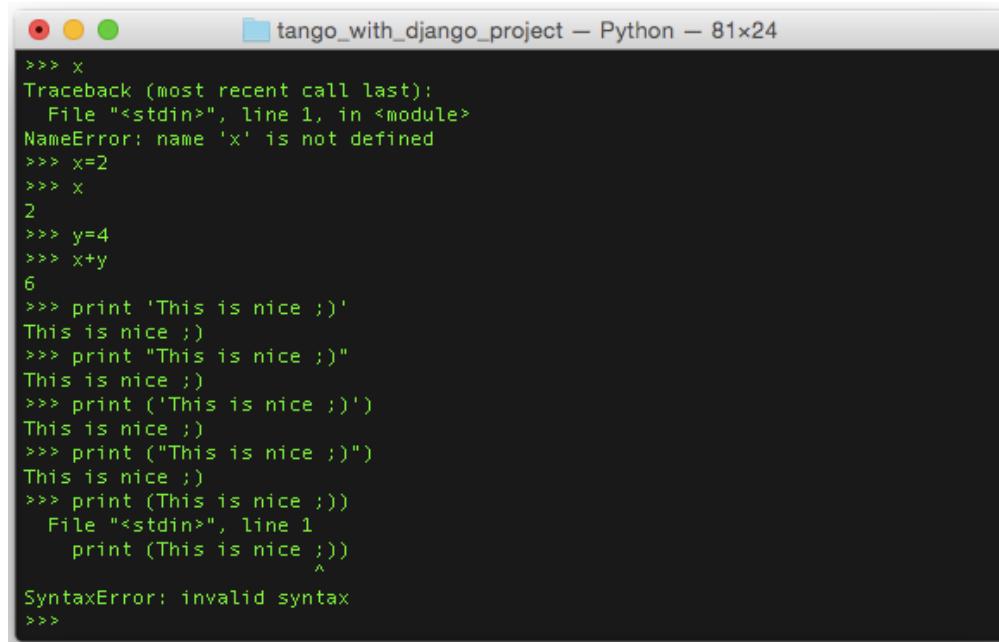


Machine Learning With TensorFlow

- Working with the shell

- The shell:

- present on OSX, Unix and Linux systems
 - provides a quick and easy to use plain text environment for swift execution of simple commands
 - it is flexible and lightweight
 - it has a Unix/Linux look and feel and is not used only for Python and its packages and modules



```
>>> x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
>>> x=2
>>> x
2
>>> y=4
>>> x+y
6
>>> print 'This is nice ;)'
This is nice ;)
>>> print "This is nice ;)"
This is nice ;)
>>> print ('This is nice ;)')
This is nice ;)
>>> print ("This is nice ;)")
This is nice ;)
>>> print (This is nice ;))
  File "<stdin>", line 1
    print (This is nice ;))
                           ^
SyntaxError: invalid syntax
>>>
```

Machine Learning With TensorFlow

- Working with the shell
 - Creating the **Virtualenv environment**:
 - to keep our dependencies nice and clean, we're going to be using **virtualenv** to create a virtual Python environment
 - first, we need to make sure that **Virtualenv** is installed along with **pip**, Python's package manager
 - run the following commands (on Mac OS X):

```
$ sudo easy_install pip  
$ sudo pip install --upgrade virtualenv
```

- create a directory to contain this environment:

```
$ sudo mkdir ~/env
```

Machine Learning With TensorFlow

- Working with the shell
 - Creating the **Virtualenv** environment:
 - now we'll create the environment using either the:
 - `virtualenv` command in Python 2 or
 - `venv` module built in Python 3, both located in `~/env/tensorflow`

```
# Python 2.7
$ virtualenv --system-site-packages ~/env/tensorflow
# Python 3
$ sudo python3 -m venv --system-site-packages ~/env/tensorflow
```

- Once it has been created, we can activate the environment using the `source` command:

```
$ source ~/env/tensorflow/bin/activate
# Notice that your prompt now has a '(tensorflow)' indicator
(tensorflow)$
```

Machine Learning With TensorFlow

- Working with the shell
 - Creating the **Virtualenv** environment:
 - make sure that the **environment is active** when we install anything with pip, since that is how **Virtualenv** keeps track of various dependencies
 - when done, shut it off by using the **deactivate** command:
`(tensorflow)$ deactivate`
 - create a shortcut for activating it by creating **alias** to your `~/.bashrc` file:
`$ sudo printf '\nalias tensorflow="source ~/env/tensorflow/bin/activate"' >> ~/.bashrc`

- restart your bash: `:~ alex$. ~/.bashrc`
- now test it:

```
$ tensorflow
# The prompt should change, as before
(tensorflow)$
```

Machine Learning With TensorFlow

- Working with the shell
 - Installing **TensorFlow**:
 - make sure that your **Virtualenv** environment from the previous section **is active and run the following** command corresponding to your operating system (Mac OS X) and version of Python:

```
# Mac OS X, Python 2.7:  
(tensorflow)$ pip install --upgrade https://storage.googleapis.com/tensorflow/mac/tensorflow-0.9.0-py2-none-any.whl  
  
# Mac OS X, Python 3.4+  
(tensorflow)$ pip3 install --upgrade https://storage.googleapis.com/tensorflow/mac/tensorflow-0.9.0-py3-none-any.whl
```

use any other version

pip3 install --upgrade
https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.15.0-py3-none-any.whl ... or ... / 2.0 or ... / 2.2

Machine Learning With TensorFlow

- Working with the shell
 - Running TensorFlow on **iPython**:
 - in case you'd like to run TensorFlow in iPython you may have to check where TensorFlow is running from:

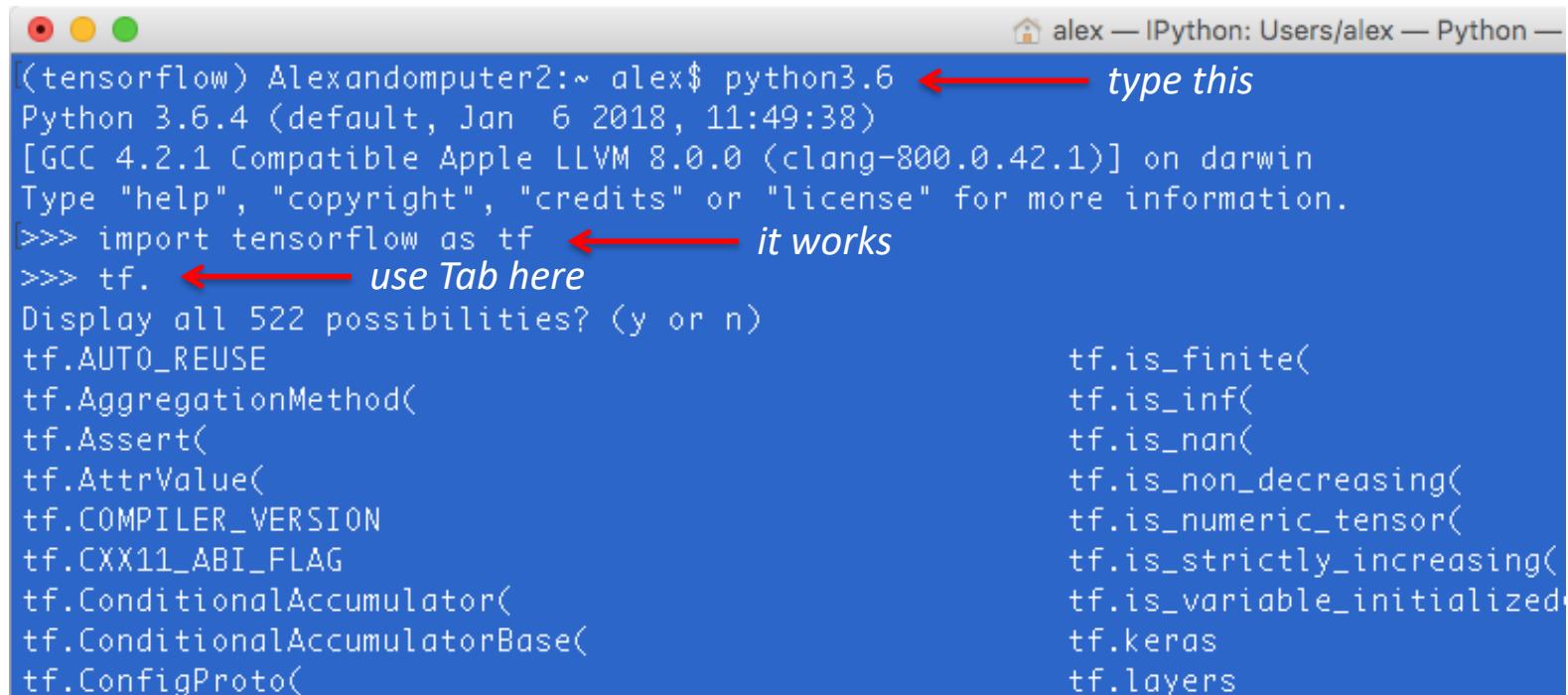
```
(tensorflow) Alexandomputer2:~ alex$ which python  
→ /Users/alex/env/tensorflow/bin/python  
(tensorflow) Alexandomputer2:~ alex$ which ipython  
→ /usr/local/bin/ipython
```
 - Since they run from different locations, iPython may not have the path to run TensorFlow so it is best to reinstall iPython like this in **virtualenv**:

```
(tensorflow) Alexandomputer2:~ alex$ sudo pip3 install --ignore-installed ipython
```

- You can also try to change the path

Machine Learning With TensorFlow

- Working with the shell
 - Try TensorFlow on [virtualenv](#) and [Python3.6](#):



A screenshot of a terminal window titled "alex — IPython: Users/alex — Python —". The window shows the following Python session:

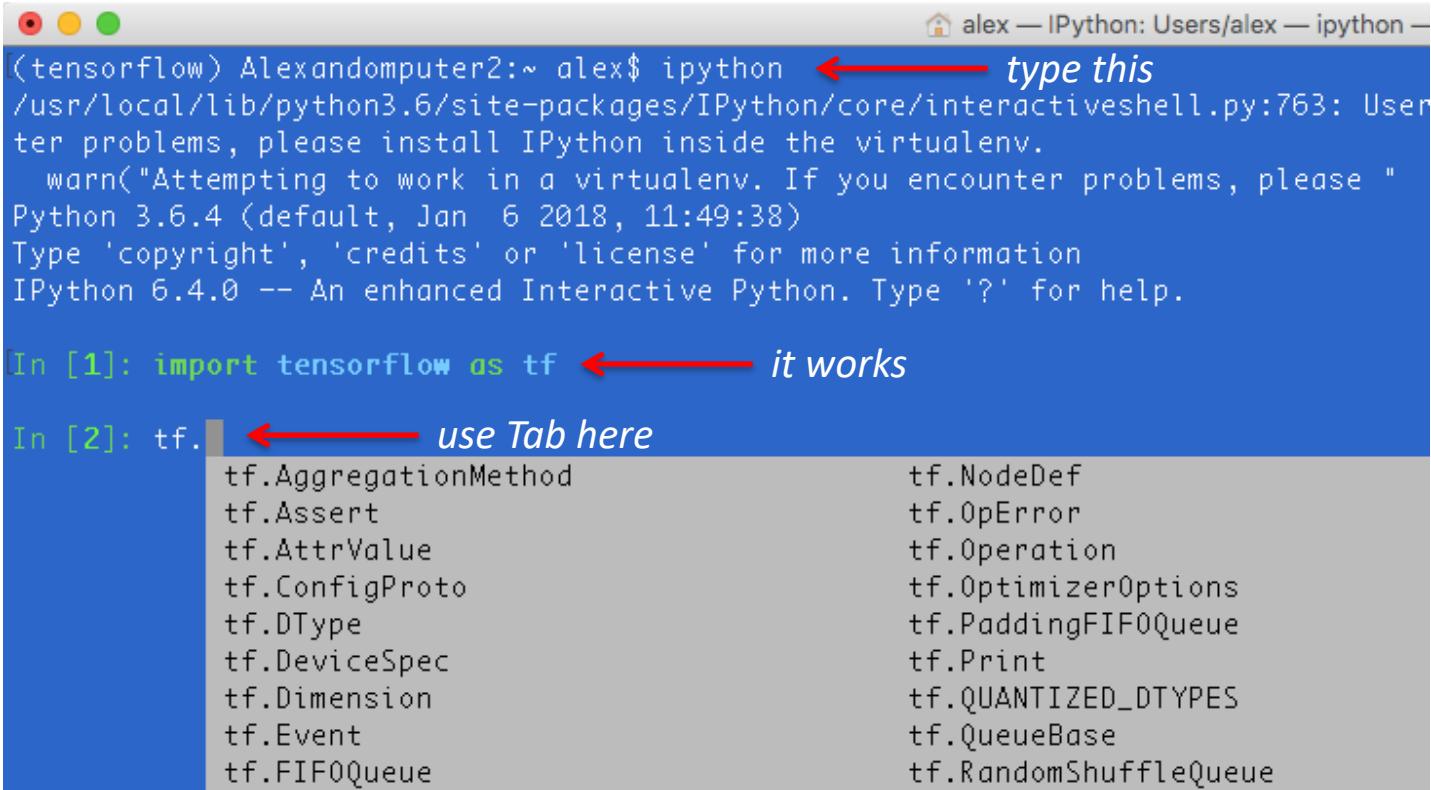
```
(tensorflow) Alexandromputer2:~ alex$ python3.6
Python 3.6.4 (default, Jan  6 2018, 11:49:38)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.
Display all 522 possibilities? (y or n)
tf.AUTO_REUSE
tf.AggregationMethod(
tf.Assert(
tf.AttrValue(
tf.COMPLIER_VERSION
tf.CXX11_ABI_FLAG
tf.ConditionalAccumulator(
tf.ConditionalAccumulatorBase(
tf.ConfigProto(
tf.is_finite(
tf.is_inf(
tf.is_nan(
tf.is_non_decreasing(
tf.is_numeric_tensor(
tf.is_strictly_increasing(
tf.is_variable_initialized(
tf.keras
tf.layers
```

Annotations with red arrows and text:

- An arrow points to "python3.6" with the text "type this".
- An arrow points to the first "tf." with the text "it works".
- An arrow points to the second "tf." with the text "use Tab here".

Machine Learning With TensorFlow

- Working with the shell
 - Try TensorFlow on **virtualenv** and **iPython**:



The screenshot shows a terminal window with the following content:

```
(tensorflow) Alexandomputer2:~ alex$ ipython ← type this
/usr/local/lib/python3.6/site-packages/IPython/core/interactiveshell.py:763: UserWarning: Attempting to work in a virtualenv. If you encounter problems, please "Python 3.6.4 (default, Jan 6 2018, 11:49:38)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.

[In [1]: import tensorflow as tf ← it works

In [2]: tf. ← use Tab here
          tf.AggregationMethod           tf.NodeDef
          tf.Assert                      tf.OpError
          tf.AttrValue                   tf.Operation
          tf.ConfigProto                 tf.OptimizerOptions
          tf.DType                       tf.PaddingFIFOQueue
          tf.DeviceSpec                  tf.Print
          tf.Dimension                  tf.QUANTIZED_DTYPES
          tf.Event                       tf.QueueBase
          tf.FIFOQueue                  tf.RandomShuffleQueue
```

Annotations with red arrows and text:

- A red arrow points to the command `ipython` with the text *type this*.
- A red arrow points to the command `import tensorflow as tf` with the text *it works*.
- A red arrow points to the beginning of the `tf.` prefix in the second input cell with the text *use Tab here*.

Machine Learning With TensorFlow

- Working with the shell
 - To see the available packages on `virtualenv` and `iPython`:

```
[In [2]: pip list ← this will not work, so we use 'help' ...]
```

The following command must be run outside of the IPython shell:

```
$ pip list
```

- To check all available modules we type either:

```
[In [3]: help('modules')]
```

Please wait a moment while I gather a list of all available modules...

we see ... `numpy` ... `matplotlib` ... `tensorflow` ... `tensorboard`

- Or:

```
In [4]: import ← this is a tab completion
```

IPython	alembic
abc	antigravity
absl	appnope
aifc	argparse

JUPYTER NOTEBOOK:

- Non-Profit, **open-source** project.
- **Supports** Data Science and **Scientific Computing** across all programming languages
- As all **open software**, Jupyter is developed by the Jupyter community (GitHub)
- **Python** language could easily be **availed to write a machine learning code on Jupyter** since it already comes with a pre-configured data science environment as it runs on the cloud or on our own hardware.

JUPYTER HUB:

- Without asking the users for installation or any other tasks, serves access to users for computational environment and resources.
- Everyone (students, researchers and scientists alike) get their own workspace which can be easily modified by the system administrators.

-----Installation-----

- Can be installed using CONDA or PiP.
 - `conda install -c conda-forge jupyterlab #CONDA`
 - `pip install jupyterlab #PiP`

Machine Learning With TensorFlow

- Working with the shell
 - Installing Jupyter notebook:
 - First, make sure you have installed iPython (on both Python 2 and 3):

```
# Python 2.7
$ sudo python2 -m pip install ipykernel
$ sudo python2 -m ipykernel install
# Python 3
$ sudo python3 -m pip install jupyterhub notebook ipykernel
$ sudo python3 -m ipykernel install
```

- Then install all dependencies:
- use pip to install the Jupyter Notebook (or pip3 for Python 3):

```
# For Python 2.7
$ sudo pip install jupyter
# For Python 3
$ sudo pip3 install jupyter
```

Jupyter

Machine Learning With TensorFlow

- Working with the shell
 - Installing **Matplotlib** and final check:
 - You should be able to see Jupyter in your environment:

```
In [3]: help('modules')
Please wait a moment while I gather a list of all available modules...
... jupyter
```

- Then install Matplotlib:

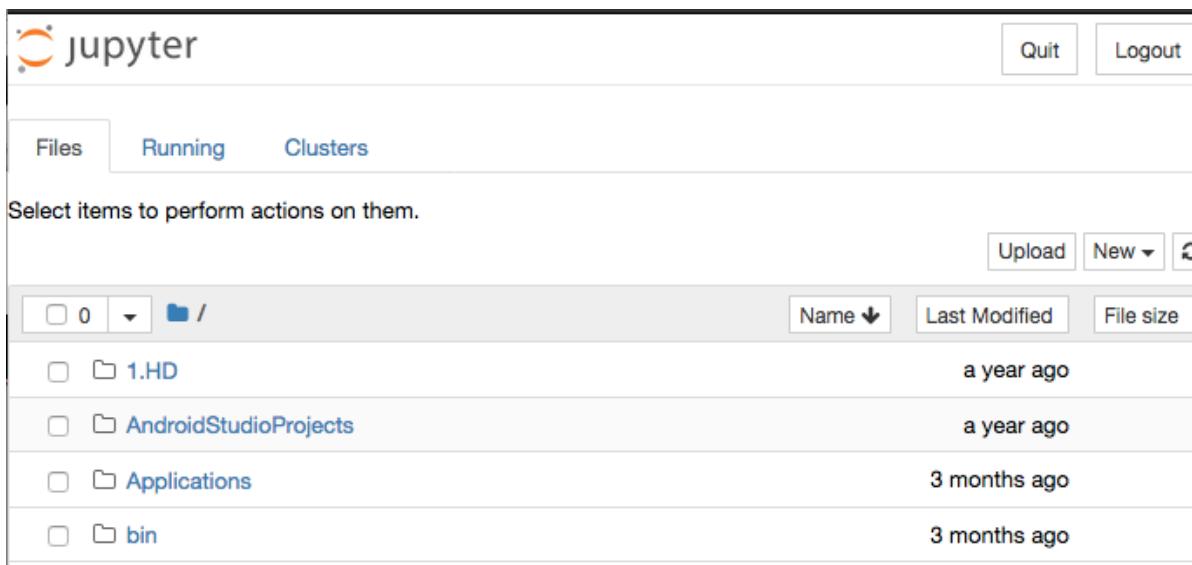
```
# Python 2.7
$ sudo apt-get build-dep python-matplotlib python-tk
# Python 3
$ sudo apt-get build-dep python3-matplotlib python3-tk
```

- Now check if it works:

```
(tensorflow)$ jupyter notebook
```

Machine Learning With TensorFlow

- Working with the shell
 - **Testing** your Jupyter environment:
 - You should get a page like this in your browser:

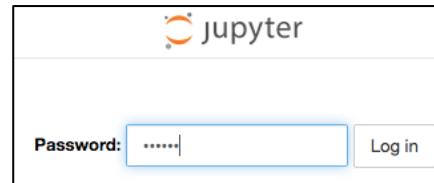


- With URL: <http://localhost:8888/tree>

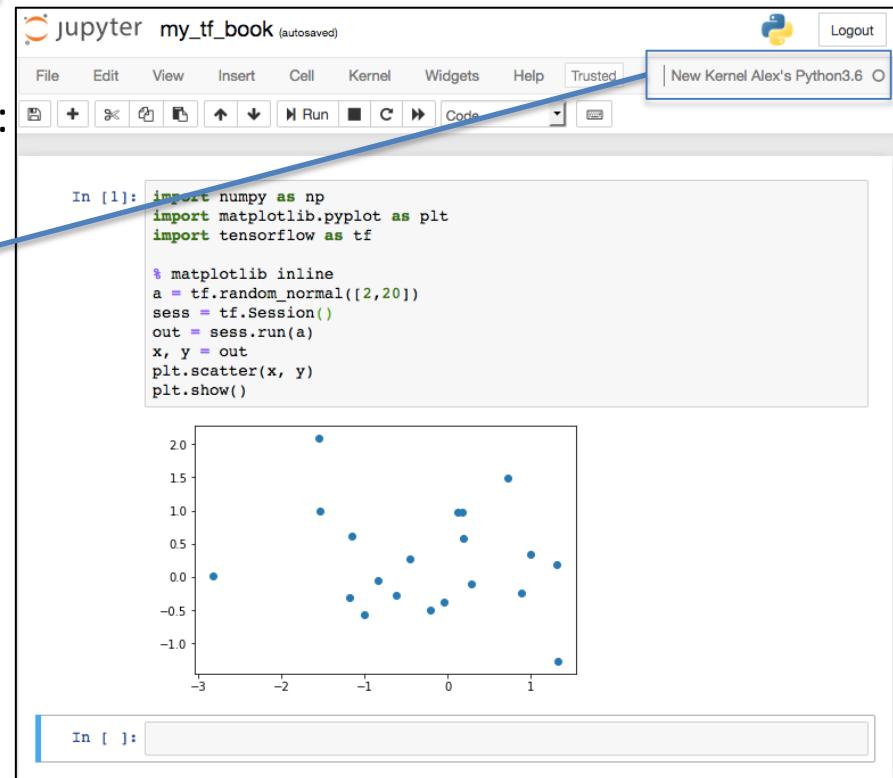
Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - You will be prompted to login:

```
alex$ jupyter notebook --NotebookApp.token=qwerty
```



- Then run any file you might have:



The screenshot shows a Jupyter Notebook interface with the following details:

- Kernel:** New Kernel Alex's Python3.6
- In [1]:**

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

%matplotlib inline
a = tf.random_normal([2,20])
sess = tf.Session()
out = sess.run(a)
x, y = out
plt.scatter(x, y)
plt.show()
```
- Output:** A scatter plot showing 20 data points in a 2D space. The x-axis ranges from -3 to 1, and the y-axis ranges from -1.0 to 2.0. The points are scattered randomly.
- Bottom Bar:** An empty input cell labeled "In []:".

... see next slide

Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - Here is how to customize your **preferred kernel**, its **source** and **name**:

```
(tensorflow) Alexandromputer2:~ alex$ nano ~/Library/Jupyter/kernels/python3.6/kernel.json
```

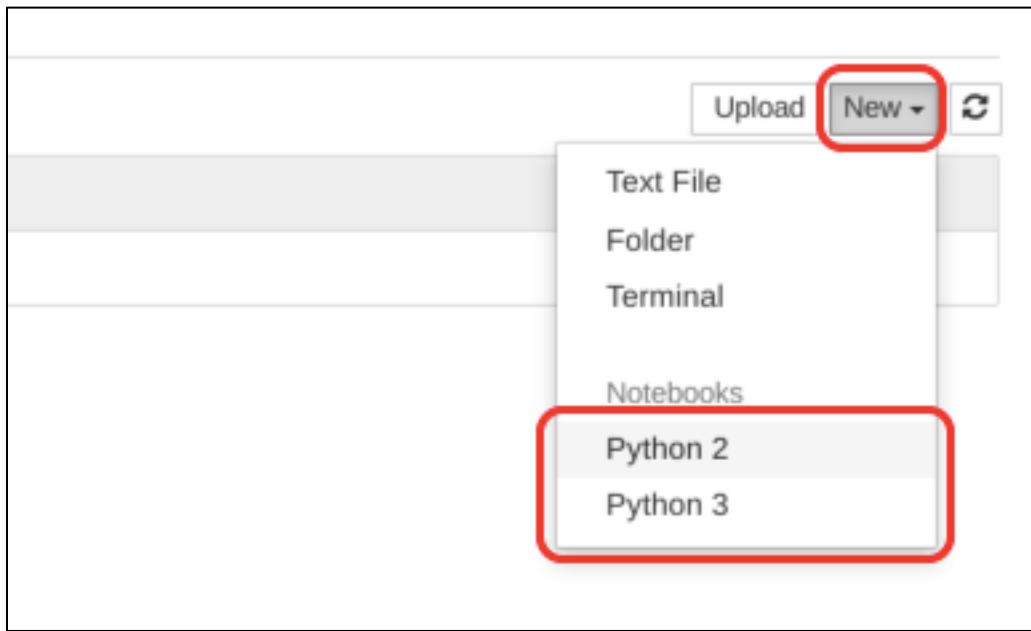


```
alex — nano ~/Library/Jupyter/kernels/python3.6/kernel.json
GNU nano 2.0.6 File: .../python3.6/kernel.json

{
  "display_name": "New Kernel Alex's Python3.6",
  "language": "python",
  "argv": [
    "/usr/local/bin/python3.6",
    "-m",
    "ipykernel",
    "-f",
    "{connection_file}"
  ]
}
```

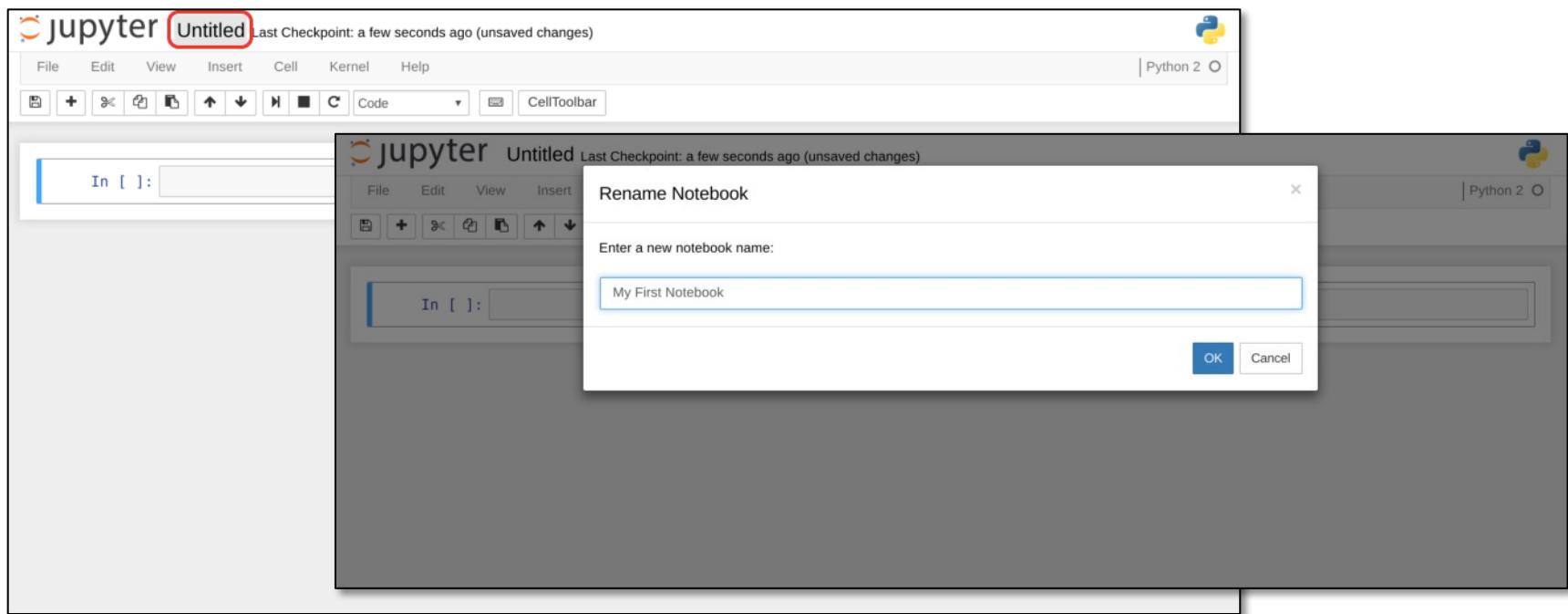
Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - You can start a new file using a different Python version (2 or 3):



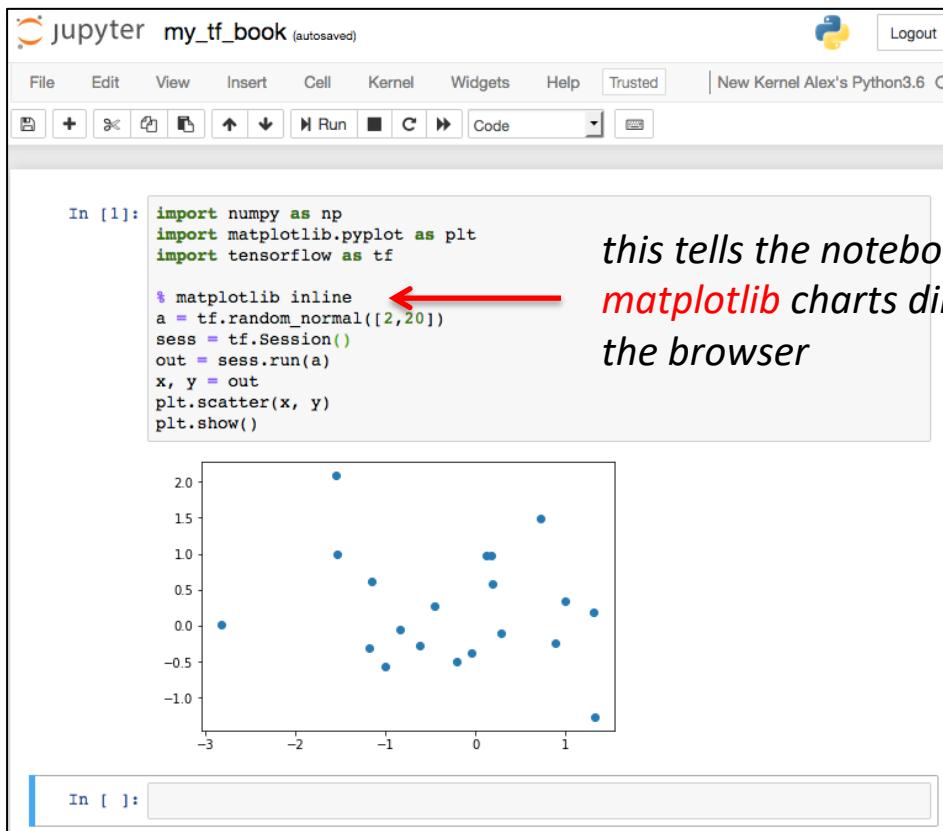
Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - Then you can begin working with it after giving it a name:



Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - You can type the code directly in the field or cut and paste it:

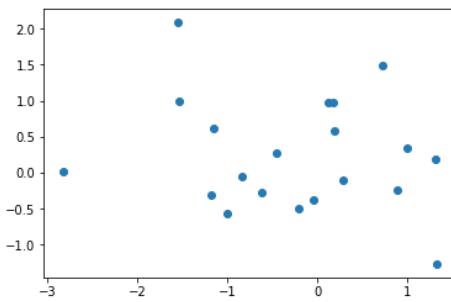


The screenshot shows a Jupyter Notebook interface with the title "jupyter my_tf_book (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Logout. A toolbar below the menu has icons for file operations and cell execution. The notebook content area shows a code cell labeled "In [1]:" containing Python code to generate a scatter plot. A red arrow points to the line "%matplotlib inline". The code imports numpy, matplotlib.pyplot, and tensorflow, generates random data, creates a session, runs the session, and plots the data. The resulting scatter plot is displayed below the code cell.

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

%matplotlib inline
a = tf.random_normal([2,20])
sess = tf.Session()
out = sess.run(a)
x, y = out
plt.scatter(x, y)
plt.show()
```

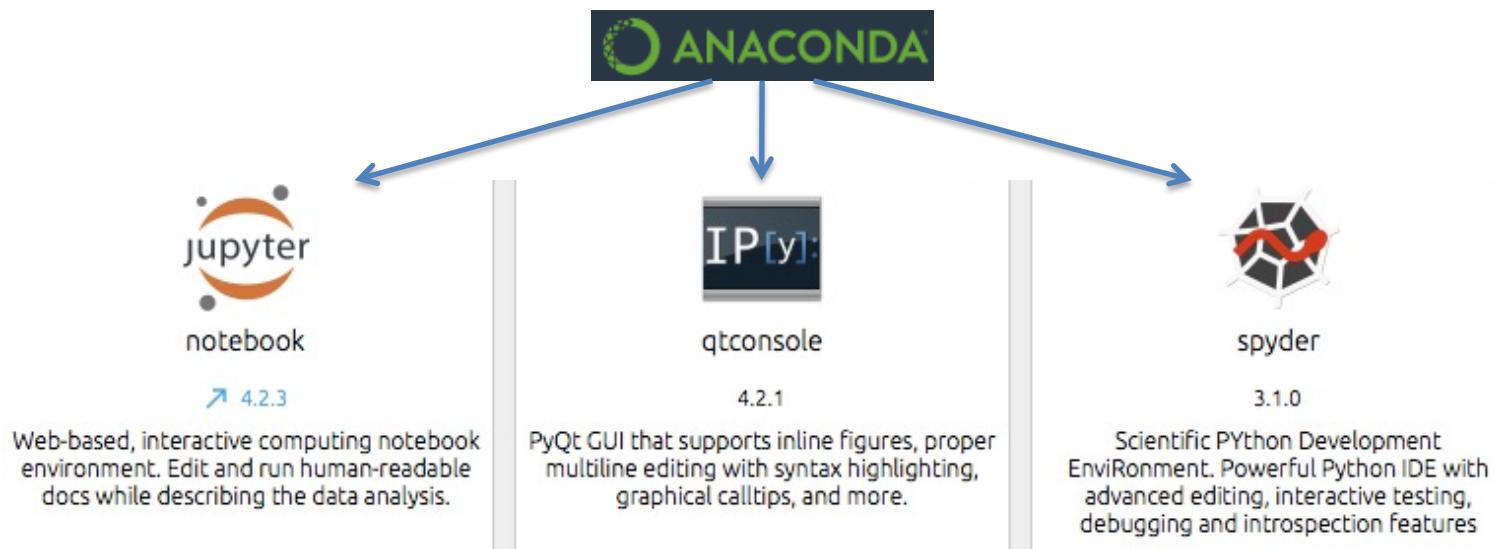
this tells the notebook to display matplotlib charts directly inside the browser



x	y
-2.8	0.0
-1.8	2.1
-1.5	1.0
-1.2	0.7
-1.0	-0.3
-0.8	-0.5
-0.5	0.2
-0.3	0.4
0.0	-0.4
0.2	0.9
0.5	0.6
0.8	1.5
1.0	0.4
1.2	-1.0

Machine Learning With TensorFlow

- Choosing your environment:
 - GUI environments using an interactive editor and debuggers for Python:
 - **Anaconda** – Anaconda is the leading open data science platform powered by Python:
<https://www.continuum.io/>



Machine Learning With TensorFlow

- Working with alternative environments:
 - Canopy – scientific and analytic Python package distribution. **has a free version, but limited**

It offers:

- object-oriented prog.
- MATLAB-like syntax
- many packages
- graphical debugger is **NOT** free

The screenshot shows the Enthought Canopy product page. At the top, there are tabs for 'For Individuals' (selected), 'For Enterprises', 'For Academics', and a link 'Already a Canopy subscriber? Download here'. Below this is a grid of four pricing plans: 'CANOPY EXPRESS' (FREE), 'CANOPY WITH PYTHON ESSENTIALS' (\$199 / year), 'CANOPY WITH PYTHON FOUNDATIONS ALL ACCESS' (\$678 / year), and 'CANOPY TRIPLE PLAY' (\$1,125 / year). A yellow button 'Contact us for discounts for 5+ users' is positioned between the middle two plans. Below the plans are buttons for 'Free Download' (blue) and 'Add to Cart' (green) for each plan. A section titled 'Pre-built and tested Python packages' compares '100+ Core' (light green) and '300+ Extended' (bright green). A link 'See included Python package details' with a plus sign icon is located at the bottom right of this section. The bottom part of the page is titled 'Integrated Analysis Environment' and lists features: 'Graphical Debugger', 'Integrated IPython', 'Advanced Code Editor', and 'Application Development Platform', each with a series of green dots indicating availability across the different plan levels.

Machine Learning With TensorFlow

- Working with alternative environments:
 - Canopy – scientific and analytic Python package distribution. **has a free version, but limited**

Package Manager:

- Packages are installed through Package Manager

The screenshot shows the Enthought Canopy Package Manager interface. At the top, there's a navigation bar with 'Refresh' and 'Not logged in.' status. A search bar contains the text 'seaborn'. Below the search bar, the main title is 'Package Manager' with the subtitle 'Install, update or remove your Python packages'. On the left, a sidebar has tabs for 'Installed' (0/106), 'Available' (1/494, highlighted in green), 'Updates' (0/0), 'History' (document icon), and 'Settings'. The main content area displays a table with columns 'Package Name' and 'Latest Available Version'. One row is shown for 'seaborn' with version '0.7.0-6'. At the bottom, a detailed view for 'seaborn' is shown with sections for 'Installed' (Currently not installed), 'Available on store' (0.7.0-6, entought/free), and a large blue 'Install v0.7.0-6' button. It also notes 'No summary available.' and 'No description available.'

Machine Learning With TensorFlow

- Working with alternative environments:
 - Canopy – scientific and analytic Python package distribution. **has a free version, but limited**

Using the browser

The screenshot shows a Jupyter Notebook interface with two code cells. The top cell, labeled 'In [*]', contains code to import various Python libraries and set up matplotlib. A blue arrow points from the text '[*] the line is in a process of executing' to the first line of this cell. The bottom cell, labeled 'In [15]', contains code to set better defaults for matplotlib, specifically setting the axes face color to white and the font size to 18. A red box highlights the warning message in the output of the top cell.

```
#Importing all the libraries needed
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import brewer2mpl
import seaborn as sns
from scipy import stats
from scipy import linalg

/Users/alex/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages/matplotlib/font_manager.py:273: UserWarning: Matplotlib is building the font cache using fc-list. This may take a moment.
    warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.')

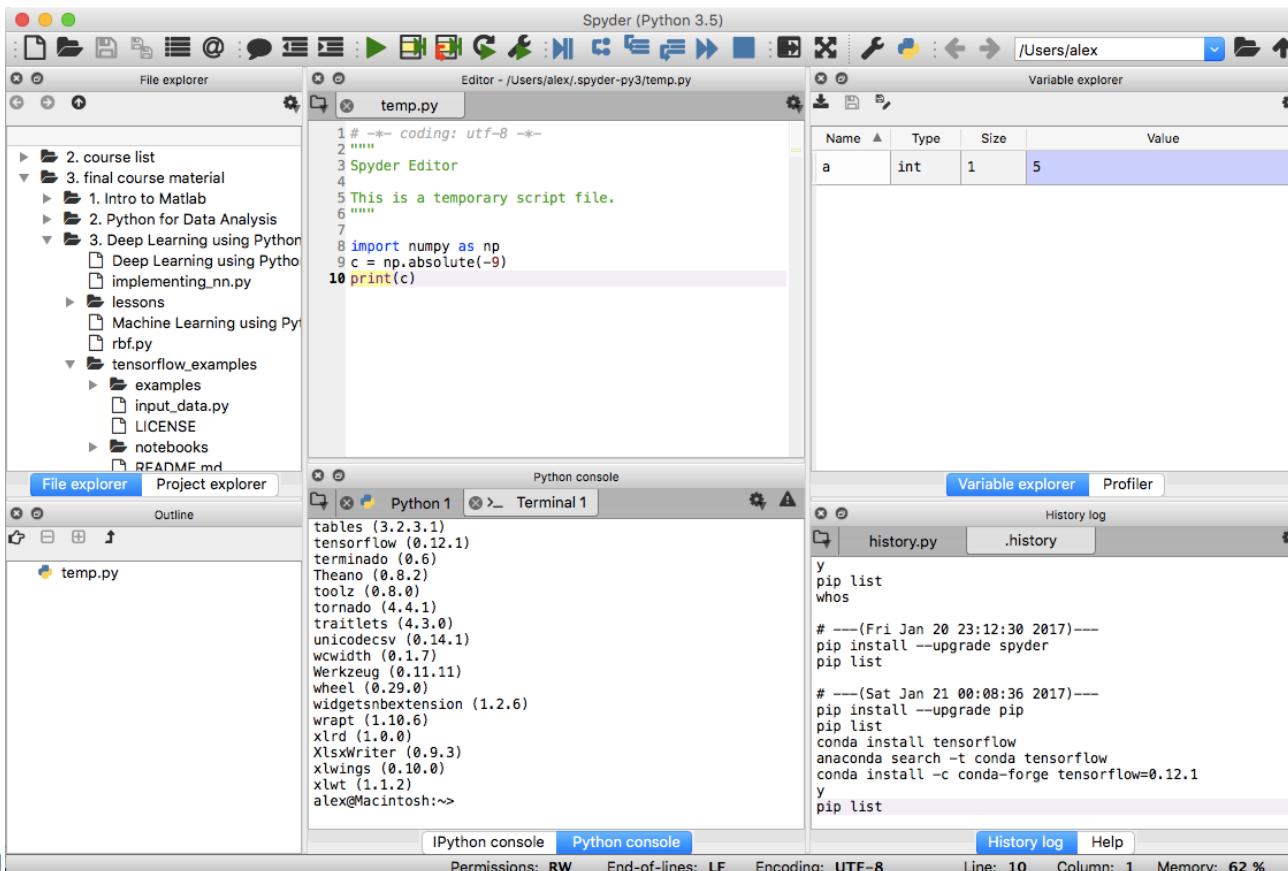
# Setting up better defaults for matplotlib
from matplotlib import rcParams

rcParams['axes.facecolor'] = 'white'
rcParams['font.size'] = 18

#colorbrewer2 Dark2 qualitative color table
dark2_colors = brewer2mpl.get_map('Dark2', 'Qualitative', 7).mpl_colors
```

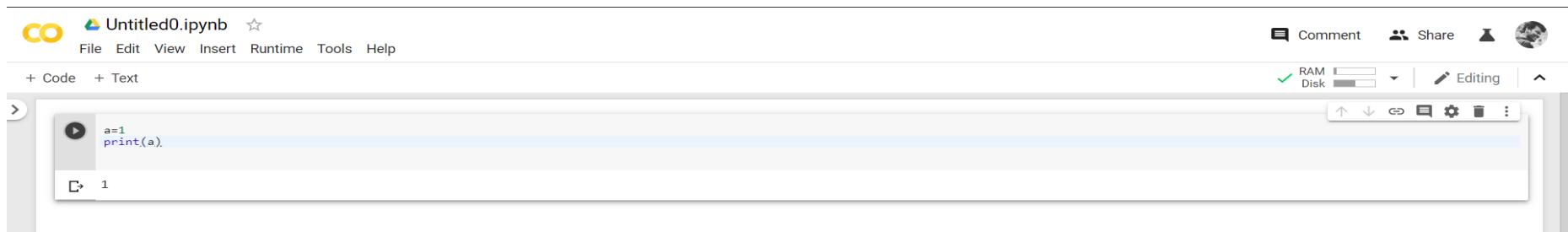
Machine Learning With TensorFlow

- Choosing your environment:
 - GUI environments using an interactive editor and debuggers for Python:
 - **Spyder** – a GUI environment for interactive Python use with shells (terminal- and Qt-based):



GOOGLE COLAB:

- Previously we learnt about installing Jupyter in our hardware, Google Colab gives us access to Jupyter Notebook on our browser
- Runs entirely on the cloud. No need for installation!
- It's easy and efficient for systems which do not have a lot of memory space.
- We can easily write a python code on it and run the shell script via the play button on the left of the code.



The screenshot shows the Google Colab interface. At the top, there is a toolbar with a logo, file navigation, and various menu options like File, Edit, View, Insert, Runtime, Tools, and Help. Below the toolbar, there are two tabs: 'Code' and 'Text'. A code cell is active, containing the following Python code:

```
a=1  
print(a)
```

At the bottom of the code cell, there is a small icon with a play button, indicating it is ready to be executed. To the right of the code cell, there are several status indicators: RAM (green checkmark), Disk (grey bar), and Editing (pencil icon). There are also buttons for Comment, Share, and other settings.

Machine Learning With TensorFlow

- Working with Anaconda and Canopy

- Useful magic commands:

Linux-like:

- pwd
- ls
- mkdir
- rmdir
- whos
- who()
- dir()
- cd
- history

iPython specific:

- edit *file*
- del(parameter)
- reset
- reset -f
- run *file*
- time
- dhist

How to create an alias:

- [35] %alias clc clear

Tips: 1. Using a magic command followed by ‘?’ will give you help for it. Example: ‘cd?’
2. Esc will get rid of the help menu

magic comprehensive list: <http://ipython.org/ipython-doc/3/interactive/magics.html>

Machine Learning

- Working with Anaconda Navigator:



- Scikit-learn **comes installed** with Anaconda distribution **by default**:

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Learning, and Community. The main area has a search bar at the top labeled "Search Environments". Below it, a tree view shows "root" expanded. To the right is a table of installed packages. A search bar in this section has "sciki" typed into it, with a blue arrow pointing to it from the text above. The table has columns for Name, Description, and Version. Two packages are listed: "scikit-image" (version 0.12.3) and "scikit-learn" (version 0.18.1). Blue arrows point from the text "comes installed" and "by default" to the "scikit-learn" row in the table.

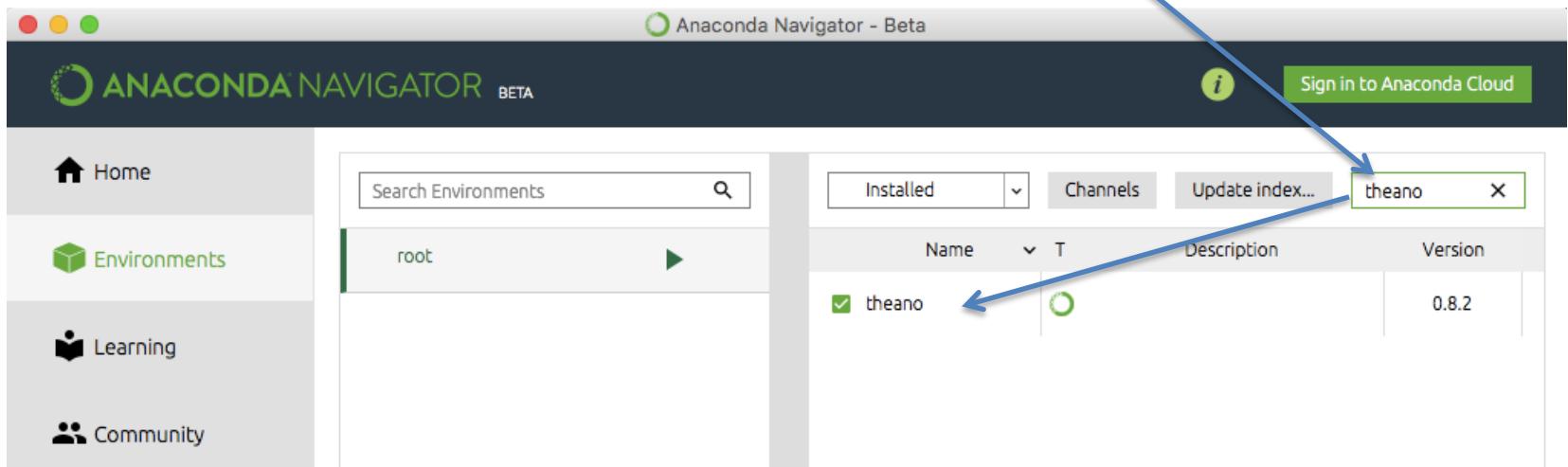
Name	Description	Version
scikit-image	Image processing routines for scipy	0.12.3
scikit-learn	Set of python modules for machine learning and data mining	0.18.1

Machine Learning

- Working with Anaconda Navigator:



- Theano is **not installed in Anaconda by default**, but is part of the standard repository and can be installed through the menu:

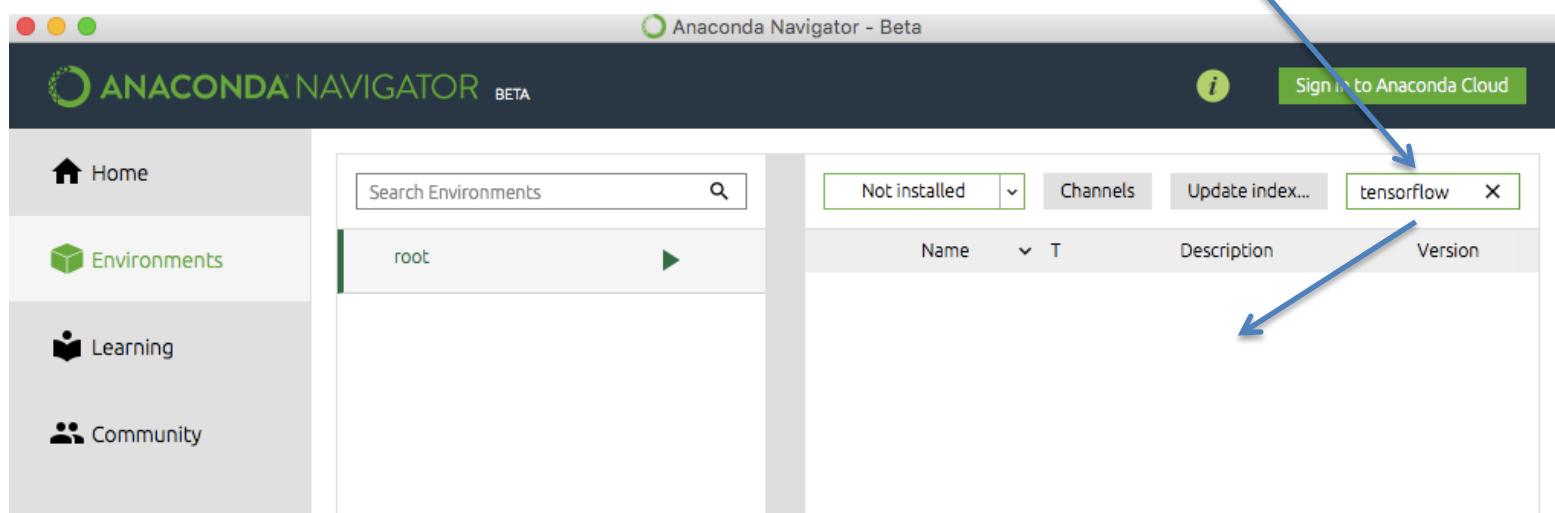


Machine Learning With TensorFlow

- TensorFlow in Anaconda:



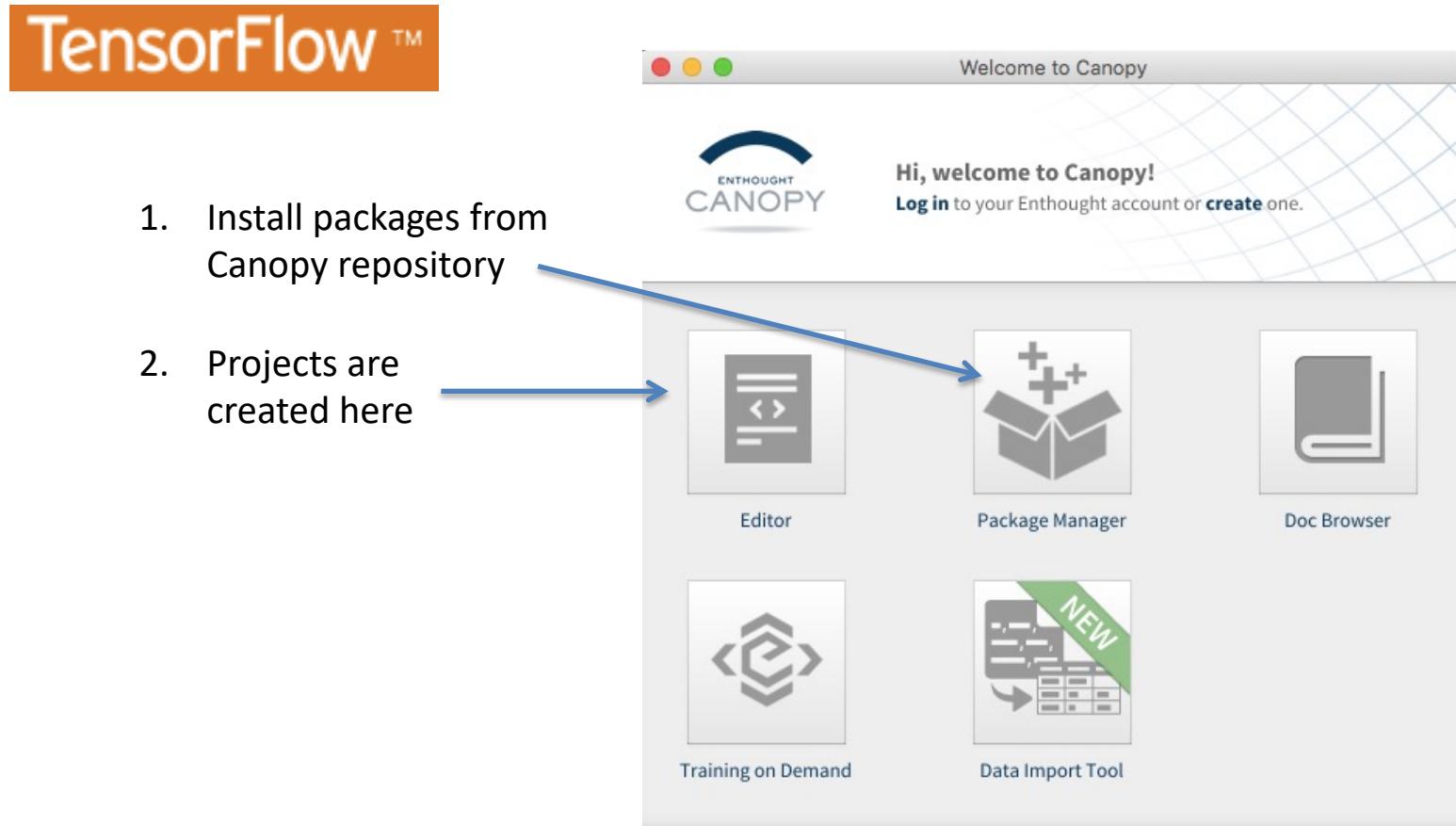
- TensorFlow **is not part of the standard Anaconda repository**:



- So you can **install it via Spyder's terminal** by typing:
`>>> conda install -c conda-forge tensorflow=1.15.0`

Machine Learning With TensorFlow

- TensorFlow in Canopy:



Machine Learning With TensorFlow

- TensorFlow in Canopy:

TensorFlow™

The screenshot shows the Canopy Package Manager interface. On the left, a sidebar has tabs for 'Installed' (7/137), 'Available' (9/564), 'Updates' (0/0), 'History', and 'Settings'. A blue arrow points from the 'Settings' tab to the 'tensorflow' row in the main table. The 'tensorflow' row is highlighted with a yellow background and circled in red. The table has columns for 'Package Name' and 'Installed Version'. The details for 'tensorflow' are shown below the table: 'Up-to-date. ✓', 'No summary available.', 'Installed: 1.3.0-2', 'Available on store: 1.3.0-2 (enthought/free)', 'Uninstall' button, and 'Re-install v1.3.0-2' button.

	Package Name	Installed Version
envisage	4.6.0-1	
nose	1.3.7-3	
protobuf	3.3.1-1	
pygments	2.1.3-1	
python_dateutil	2.6.0-1	
tensorflow	1.3.0-2	
widgetsnbextension	2.0.0-1	

tensorflow Up-to-date. ✓

No summary available.

Installed: 1.3.0-2

Available on store: 1.3.0-2 (enthought/free)

Uninstall

Re-install v1.3.0-2

Machine Learning With TensorFlow

- TensorFlow in Canopy:

TensorFlow™

→	numpy	1.13.3-1
→	matplotlib	2.0.0-5
→	scikit_learn	0.19.1-2
	scikits.image	0.13.0-5
	scikits.learn	0.8-2
→	scipy	1.0.0-1

Machine Learning With TensorFlow

- TensorFlow in Canopy:

The screenshot shows the Canopy IDE interface. On the left, the File Browser pane displays a list of recent files, including 'alex' and 'Recent Files' (with 'helloworld.py' selected). A blue arrow points from this pane to the main workspace. In the center, the Editor - Canopy window shows the code for 'helloworld.py'. The code imports tensorflow, creates a Constant op, starts a session, runs the op, and prints the result. A red box highlights the code area. To the right of the code is the Python console output, which includes a welcome message, version information (Python 2.7.11), help documentation for IPython 4.1.2, and a command history showing the execution of the code. A blue arrow points from the code area to the console output. A blue arrow also points from the top right corner of the slide towards the top right corner of the Canopy window.

```
''' HelloWorld example using TensorFlow library '''  
import tensorflow as tf  
# 1. Create a Constant op  
# The op is added as a node to the default graph.  
#  
# 2. The value returned by the constructor is the output of the Constant op.  
hello = tf.constant('Hello, TensorFlow!')  
# Start tf session  
sess = tf.Session()  
# Run the op  
print(sess.run(hello))
```

Welcome to Canopy's interactive data-analysis environment!
Type '?' for more information.
Python 2.7.11 | 64-bit | (default, Jun 11 2016, 03:41:56)
Type "copyright", "credits" or "license" for more information.
IPython 4.1.2 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

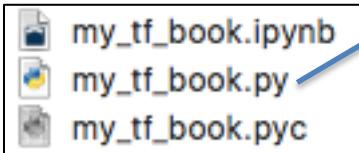
```
In [1]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction"  
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction  
In [2]:
```

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

1. as script file



The screenshot shows the Canopy IDE interface. On the left is a 'File Browser' window showing a directory structure under 'alex' with various Python files. A blue arrow points from the 'my_tf_book.ipynb' file in the browser to the code editor. The code editor contains a Python script named 'helloworld.py'. The terminal window at the bottom shows the user navigating through their workspace and attempting to run the script.

```
Editor - Canopy
File Browser
Recent Files
alex
my_tf_book.py
helloworld.py
main.py
Concept01_defining_tensors.ipynb
moving_avg.py
types.py
interactive_session.py
gradient.py
boston-marathon.py
Soumyendu-Sarkar-compsci-...
Soumyendu-Sarkar-compsci-...
FinalProject.ipynb
FinalProjectCombined-Copy1.ipynb
final project part-2 edit.ipynb
FinalProjectCombined-Copy1.ipynb
Titanic Berkeley Project V2.ipynb
double_pendulum.py
Single Electron Schrodinger E...
compsciX433_projectnc.ipynb
animation.py
final project.ipynb
Titanic Berkeley Project V2 - ...
Titanic V0.2b-2.ipynb

Editor - Canopy
helloworld.py
my_tf_book.py

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import tensorflow as tf
4
5 # This is only for Jupyter plotting in-line:
6 # %matplotlib inline
7
8 a = tf.random_normal([2,20])
9 sess = tf.Session()
10 out = sess.run(a)
11 x, y = out
12 plt.scatter(x, y)
13 plt.show()

Python ~1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction
In [1]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction"
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction

In [2]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/tensorflow_examples/alex_examples"
[Errno 2] No such file or directory: '/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/tensorflow_examples/alex_examples'
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction

In [3]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples"
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples

In [4]:
```

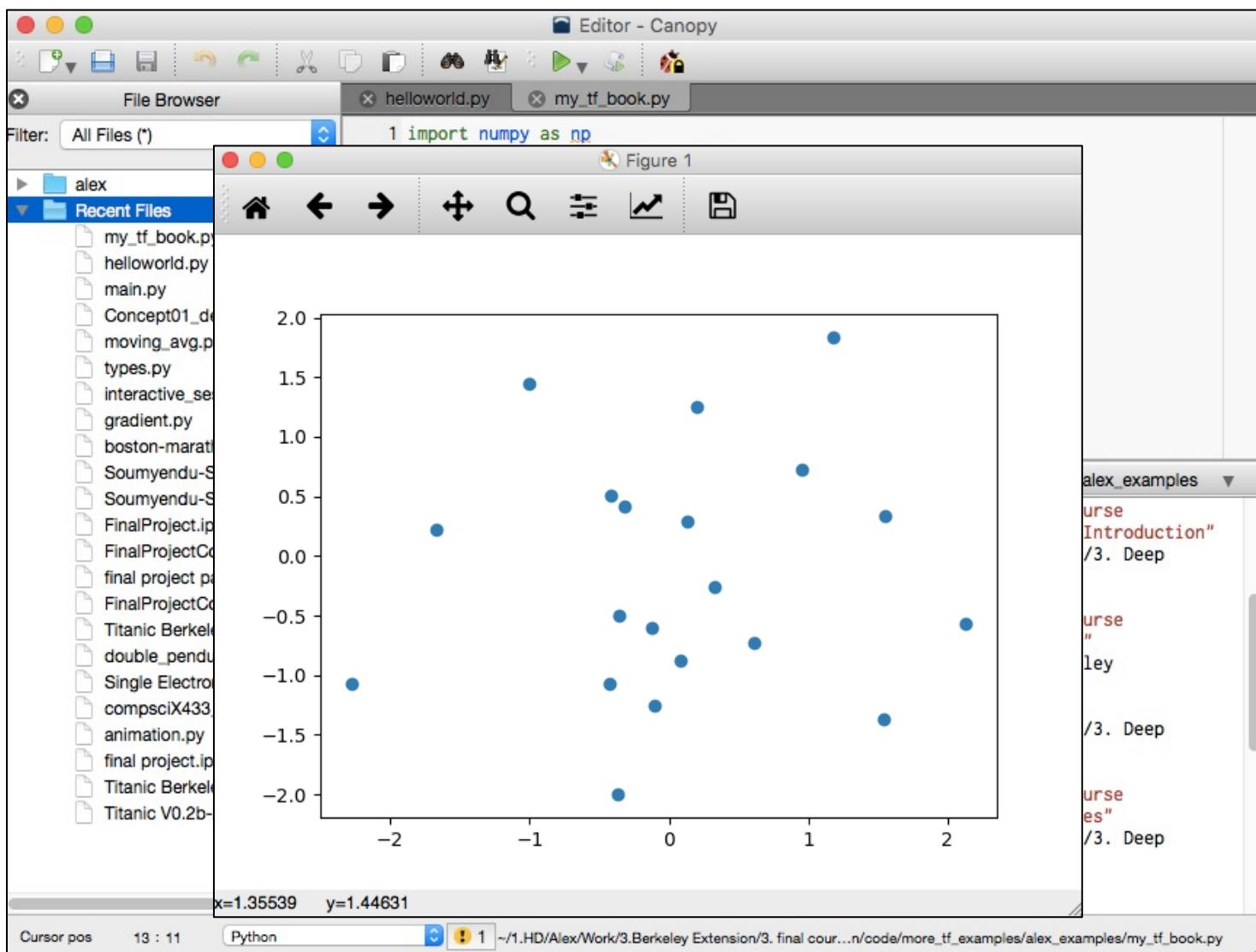
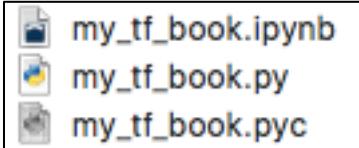
Cursor pos 13 : 11 Python 1 ~1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples/my_tf_book.py

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

1. as script file

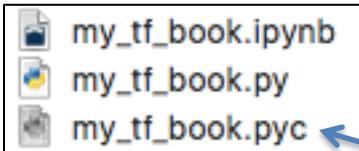


Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

2. as imported file
(creates compiled version .pyc)



The screenshot shows the Canopy IDE interface. On the left is a "File Browser" window listing various Python files. In the center is a code editor window titled "Editor - Canopy" containing the following code:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import tensorflow as tf
4
5 # This is only for Jupyter plotting in-line:
6 # %matplotlib inline
7
8 a = tf.random_normal([2,20])
9 sess = tf.Session()
10 out = sess.run(a)
11 x, y = out
12 plt.scatter(x, y)
13 plt.show()
```

On the right is a terminal window titled "Python" showing the following session:

```
In [6]: pwd
Out[6]: '/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples'

In [7]: ls
My First Notebook.ipynb    my_tf_book.py
my_tf_book.ipynb          my_tf_book.pyc

In [8]: import my_tf_book
In [9]: run my_tf_book
In [10]: |
```

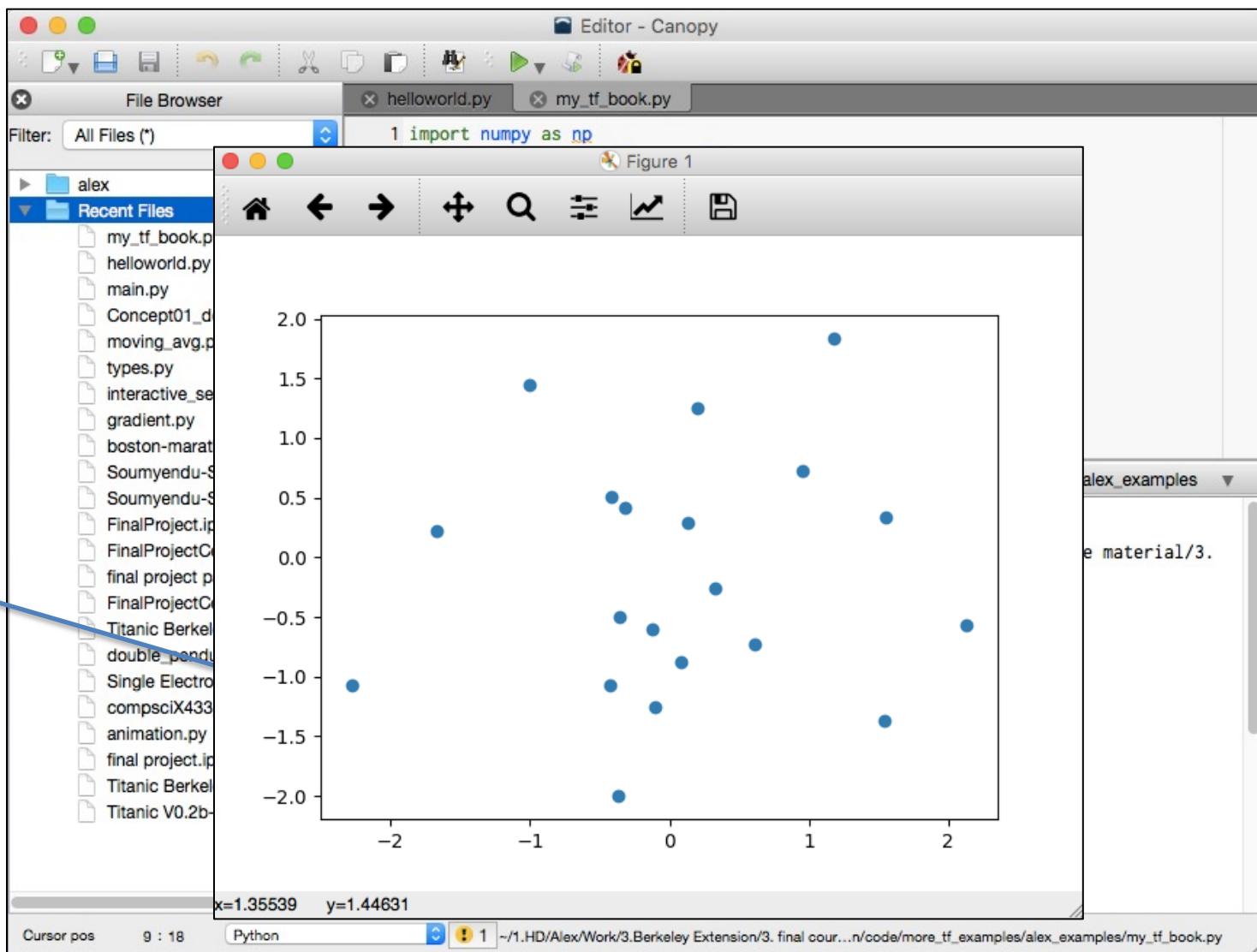
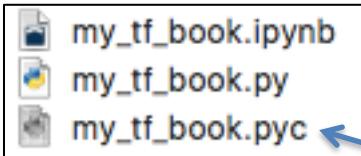
A red box highlights the command "import my_tf_book" in the terminal, and a blue arrow points from the "my_tf_book.py" file in the file browser to this highlighted command. Another blue arrow points from the "my_tf_book.pyc" file in the file browser to the "run my_tf_book" command in the terminal.

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

2. as imported file
(creates compiled version .pyc)

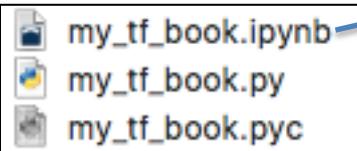


Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

3. in notebook
(it is a .ipynb file)



The screenshot shows the Canopy IDE interface. On the left is the 'File Browser' pane, which lists several Python files in the 'alex' directory, including 'my_tf_book.ipynb', 'helloworld.py', and 'main.py'. A blue arrow points from the 'my_tf_book.ipynb' icon in the bottom-left corner towards the code editor. The code editor window has tabs for 'helloworld.py' and 'my_tf_book.py'. The code in 'helloworld.py' is:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import tensorflow as tf
4
5 # This is only for Jupyter plotting in-line:
6 # %matplotlib inline
7
8 a = tf.random_normal([2,20])
9 sess = tf.Session()
10 out = sess.run(a)
11 x, y = out
12 plt.scatter(x, y)
13 plt.show()
```

A blue arrow points from the 'my_tf_book.ipynb' tab in the code editor towards the Python console. The Python console window shows the following session:

```
In [6]: pwd
Out[6]: u'/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples'

In [7]: ls
My First Notebook.ipynb  my_tf_book.py
my_tf_book.ipynb        my_tf_book.pyc

In [8]: import my_tf_book
In [9]: run my_tf_book
In [10]: |
```

The status bar at the bottom indicates 'Cursor pos 9 : 18' and 'Python'.

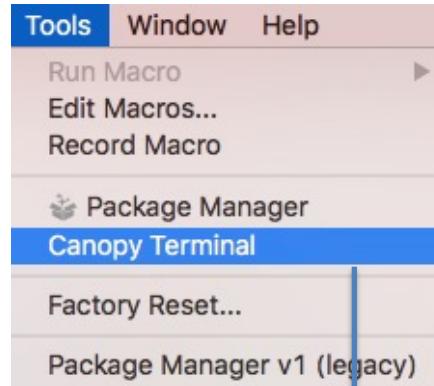
Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

3. in notebook

(it is a .ipynb file)



```
Hard Disk — Canopy Terminal — jupyter-notebook --NotebookApp.token=qwerty — 98x22

(Canopy 64bit) Alexandomputer2:/ alex$ jupyter notebook --NotebookApp.token=qwerty
[I 12:54:41.481 NotebookApp] The port 8888 is already in use, trying another port.
[I 12:54:41.552 NotebookApp] Serving notebooks from local directory: /Users/alex
[I 12:54:41.552 NotebookApp] 0 active kernels
[I 12:54:41.552 NotebookApp] The Jupyter Notebook is running at: http://localhost:8889/?token=...
[I 12:54:41.552 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to
skip confirmation).
[I 12:54:42.385 NotebookApp] 302 GET /tree (::1) 2.08ms
[I 12:54:45.776 NotebookApp] 302 POST /login?next=%2Ftree (::1) 2.72ms
```

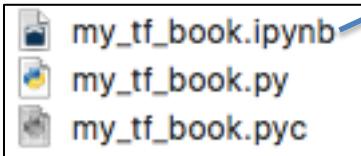
Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

3. in notebook

(it is a .ipynb file)



The screenshot shows a Jupyter Notebook interface titled 'jupyter my_tf_book'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Help, and a Python 2 option. The toolbar includes icons for file operations like Open, Save, and Cell. The code cell 'In [1]' contains the following Python code:

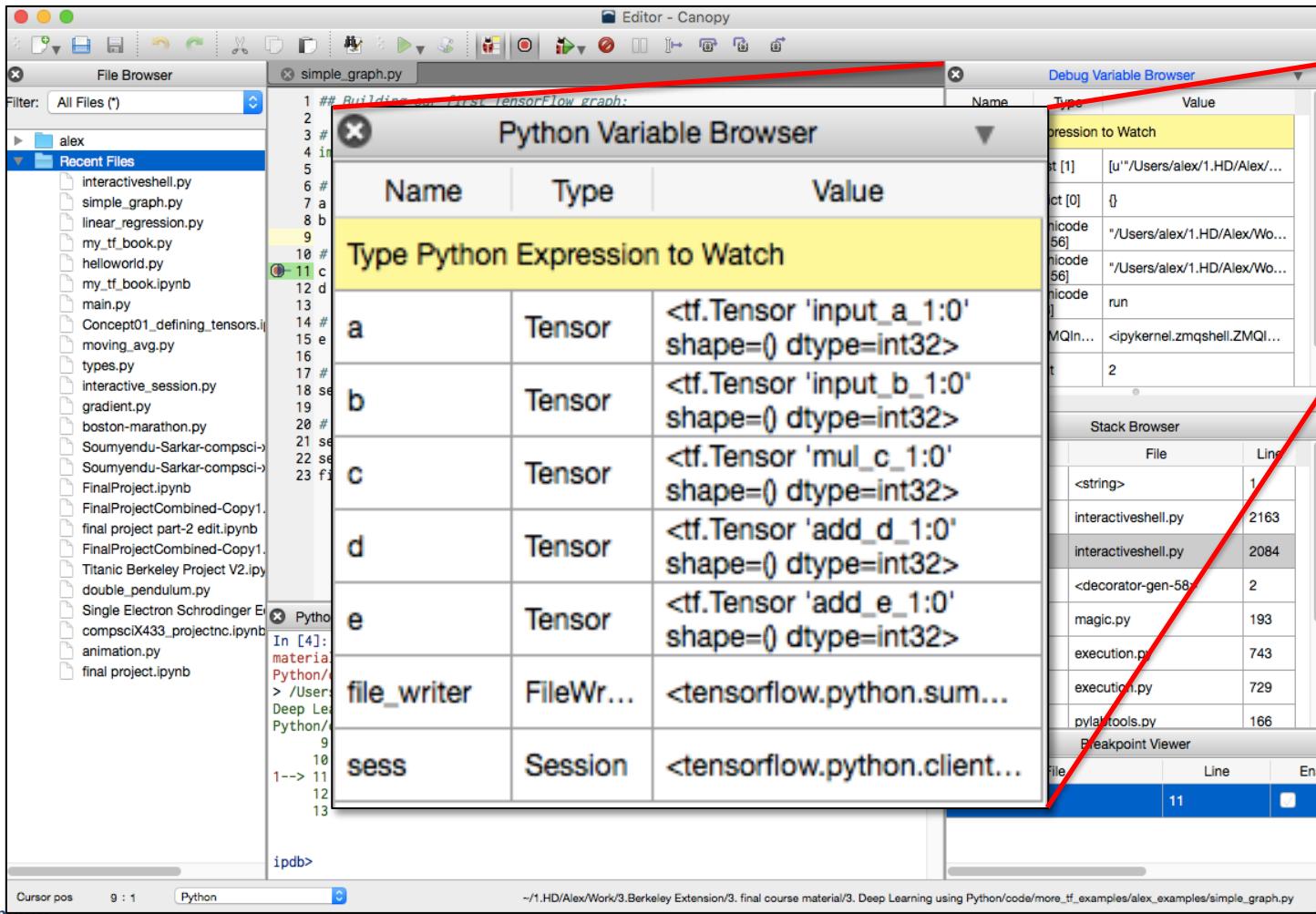
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

%matplotlib inline
a = tf.random_normal([2,20])
sess = tf.Session()
out = sess.run(a)
x, y = out
plt.scatter(x, y)
plt.show()
```

The output of the code is a scatter plot showing 20 data points in a 2D space. The x-axis ranges from approximately -1.0 to 1.5, and the y-axis ranges from -1.5 to 2.0. The points are scattered with some clustering, particularly around (0, 0).

Canopy debugger

- Running in debug mode:



Machine Learning With TensorFlow

Assignment:

Choose, install and configure your Python-Tensorflow environment

TensorFlow: how it works?

- Nodes and **tensors** in TensorFlow are Python objects, and TensorFlow applications are themselves Python applications
- The **libraries** of transformations that are available through TensorFlow are **written as high-performance C++ binaries**.
- Python just directs traffic between the pieces, and provides high-level programming abstractions to hook them together.
- TensorFlow applications can be run on most any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs.
- If you use Google's own cloud, you can run TensorFlow on Google's custom [TensorFlow Processing Unit \(TPU\)](#) silicon for further acceleration.

TensorFlow: how it works?

- TensorFlow 2.0, in beta as of June 2019, revamped the framework in many ways based on user feedback, to make it easier to work with and more performant
- Distributed training is easier to run thanks to a new API
- Support for TensorFlow Lite makes it possible to deploy models on a greater variety of platforms
- However, code written for earlier versions of TensorFlow must be rewritten—sometimes only slightly, sometimes significantly—to take maximum advantage of new TensorFlow 2.0 features.

TensorFlow v/s Competition

- TensorFlow competes with a slew of other machine learning frameworks
- PyTorch, CNTK, and MXNet are three major frameworks that address the same needs

1) PyTorch:

- In addition to being built with Python, and has many other similarities to TensorFlow:
 - Hardware-accelerated components under the hood
 - A highly interactive development model that allows for design-as-you-go work
 - A better choice for fast development of projects that need to be up and running in a short time, but TensorFlow wins on larger projects and more complex workflows

TensorFlow v/s Competition

- TensorFlow competes with a slew of other machine learning frameworks
- PyTorch, CNTK, and MXNet are three major frameworks that address the same needs

2) CNTK :

- The Microsoft Cognitive Toolkit (MCT)
- Uses a graph structure to describe dataflow
- Focuses mostly on creating deep learning neural networks
- Handles many neural network jobs faster
- But CNTK isn't as easy to learn or deploy as TensorFlow

TensorFlow v/s Competition

3) Apache Mxnet :

- Adopted by Amazon as the premier deep learning framework on AWS
- Can scale almost linearly across multiple GPUs and multiple machines.
- It also supports a broad range of language APIs—Python, C++, Scala, R, JavaScript, Julia, Perl, Go—although its native APIs aren't as pleasant to work with as TensorFlow's.

PROS

GRAPHS :

has better computational graph visualizations, which are indigenous when compared to other libraries like Torch and Theano

LIBRARY MANAGEMENT:

Backed by Google, TensorFlow has the advantage of the seamless performance, quick updates and frequent new releases with new features

DEBUGGING :

lets you execute subparts of a graph which gives it an upper-hand as you can introduce and retrieve discrete data onto an edge

SCALABILITY :

The libraries can be deployed on a gamut of hardware machines, starting from cellular devices to computers with complex setups

1

2

3

4

TensorFlow

CONS

MISSING SYMBOLIC LOOPS:

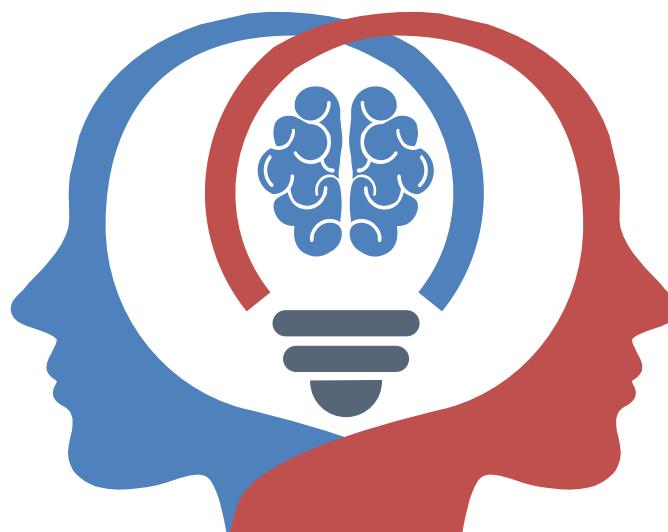
TensorFlow does not offer this feature, but there is a workaround using finite unfolding (bucketing).

LIMITED SUPPORT FOR WIN:

There is still a wide variety of users who are comfortable with a windows environment rather than a [Linux](#) in their systems and TensorFlow does not assuage these users.

BENCHMARK TESTS :

On small scale TensorFlow lacks behind in both speed and usage when compared to its competitors (not the case for large computations!)



1

2

3



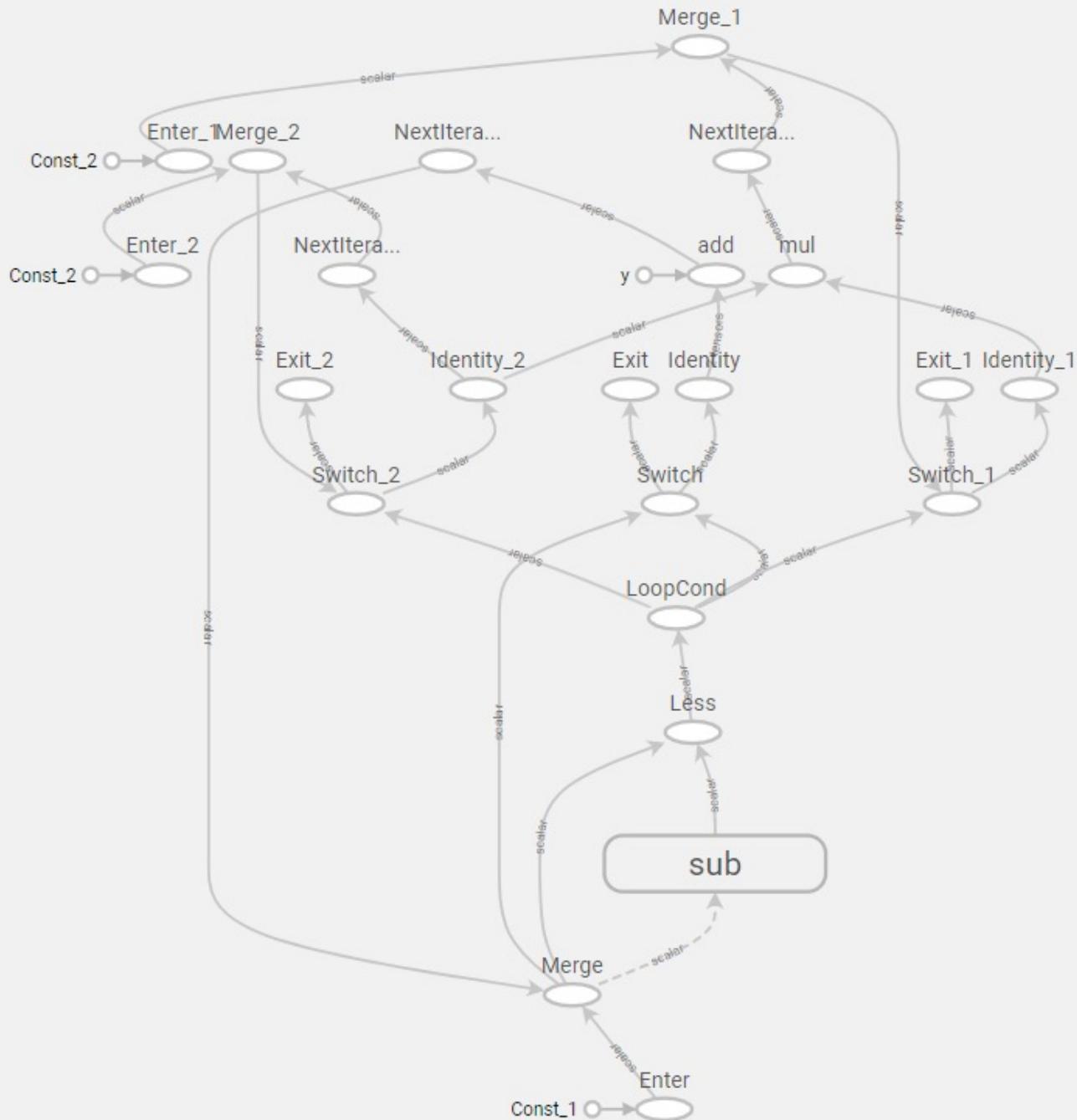
```
import tensorflow as tf

N = tf.constant(100)
i = tf.constant(0)
x = tf.constant(1.5)

def body(i, x, x0):
    return i + 1, x * x0, x0

output = tf.while_loop( lambda i,x,x0: i < N-1, body, [i, x, x] );
# a `Session` encapsulates the environment in which `Operations` are executed, and `Tensors` are evaluated
with tf.Session() as sess:
    writer = tf.summary.FileWriter("while_loop_example")
    writer.add_graph(sess.graph)
    print(sess.run(output))
```

while



Comparison Chart 1/3

	Platform	Cost	Written in language	Algorithms or Features
Scikit Learn	Linux, Mac OS, Windows	Free.	Python, Cython, C, C++	Classification Regression Clustering Preprocessing Model Selection Dimensionality reduction.
PyTorch	Linux, Mac OS, Windows	Free	Python, C++, CUDA	Autograd Module Optim Module nn Module
TensorFlow	Linux, Mac OS, Windows	Free	Python, C++, CUDA	Provides a library for dataflow programming.
Weka	Linux, Mac OS, Windows	Free	Java	Data preparation Classification Regression Clustering Visualization Association rules mining

Comparison Chart 2/3

KNIME	Linux, Mac OS, Windows	Free	Java	Can work with large data volume. Supports text mining & image mining through plugins
Colab	Cloud Service	Free	-	Supports libraries of PyTorch, Keras, TensorFlow, and OpenCV
Apache Mahout	Cross-platform	Free	Java Scala	Preprocessors Regression Clustering Recommenders Distributed Linear Algebra.
Accors.Net	Cross-platform	Free	C#	Classification Regression Distribution Clustering Hypothesis Tests & Kernel Methods Image, Audio & Signal. & Vision
Shogun	Windows Linux UNIX Mac OS	Free	C++	Regression Classification Clustering Support vector machines. Dimensionality reduction

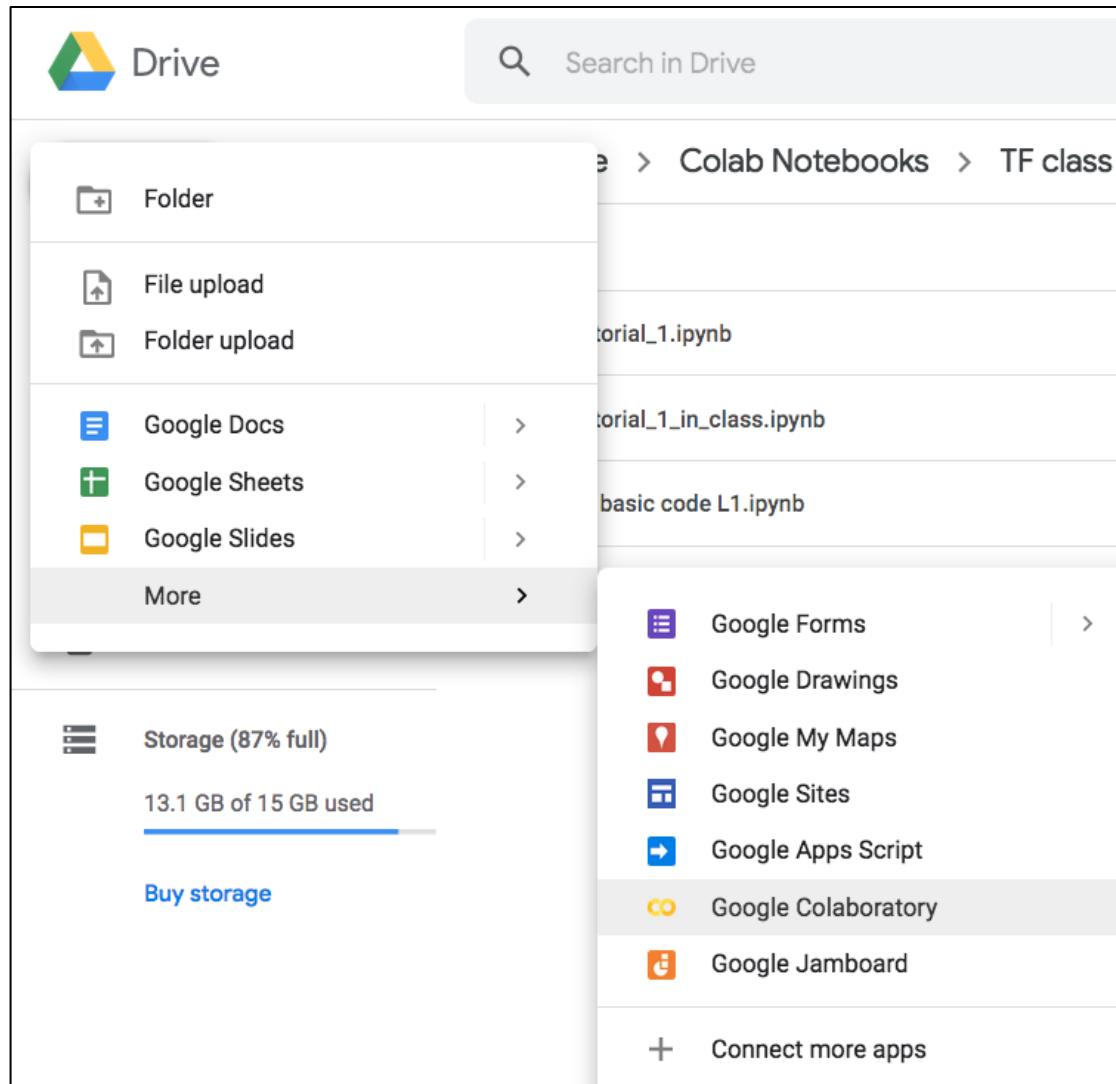
Comparison Chart 3/3

Shogun	Windows Linux UNIX Mac OS	Free	C++	Regression Classification Clustering Support vector machines. Dimensionality reduction Online learning etc.
Keras.io	Cross-platform	Free	Python	API for neural networks
Rapid Miner	Cross-platform	Free plan Small: \$2500 per year. Medium: \$5000 per year. Large: \$10000 per year.	Java	Data loading & Transformation Data preprocessing & visualization.

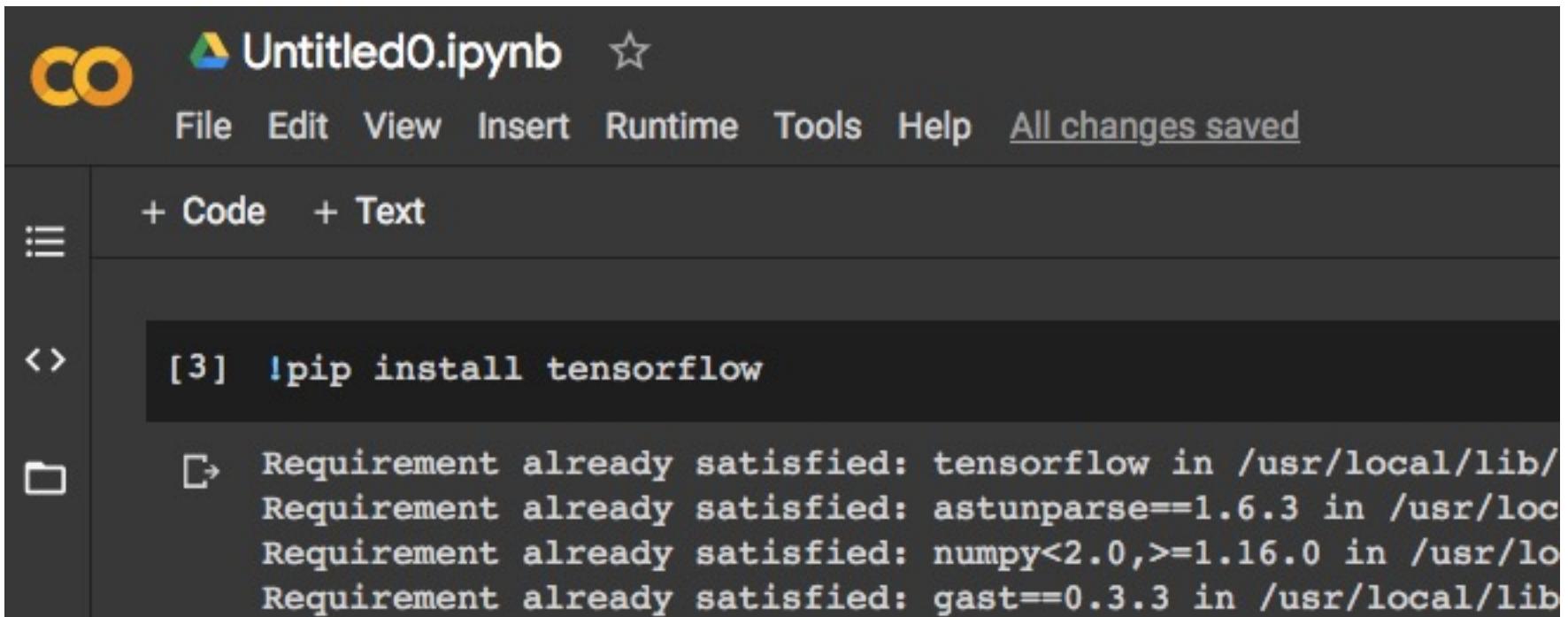
References

- <https://www.tensorflow.org/install>
- <https://www.udemy.com/tutorial/complete-guide-to-tensorflow-for-deep-learning-with-python/installing-tensorflow-environment/>
- <https://github.com/tensorflow/docs/blob/master/site/en/r1/guide/extend/architecture.md>
- https://www.tensorflow.org/guide/effective_tf2
- <https://www.tensorflow.org/guide/keras/overview>
- <https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/>
- <https://hub.packtpub.com/tensorflow-always-tops-machine-learning-artificial-intelligence-tool-surveys/>
- <https://medium.com/tensorflow/intelligent-scanning-using-deep-learning-for-mri-36dd620882c4>
- https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/1_Introduction/basic_operations.py
- <https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html>

Start with Colab



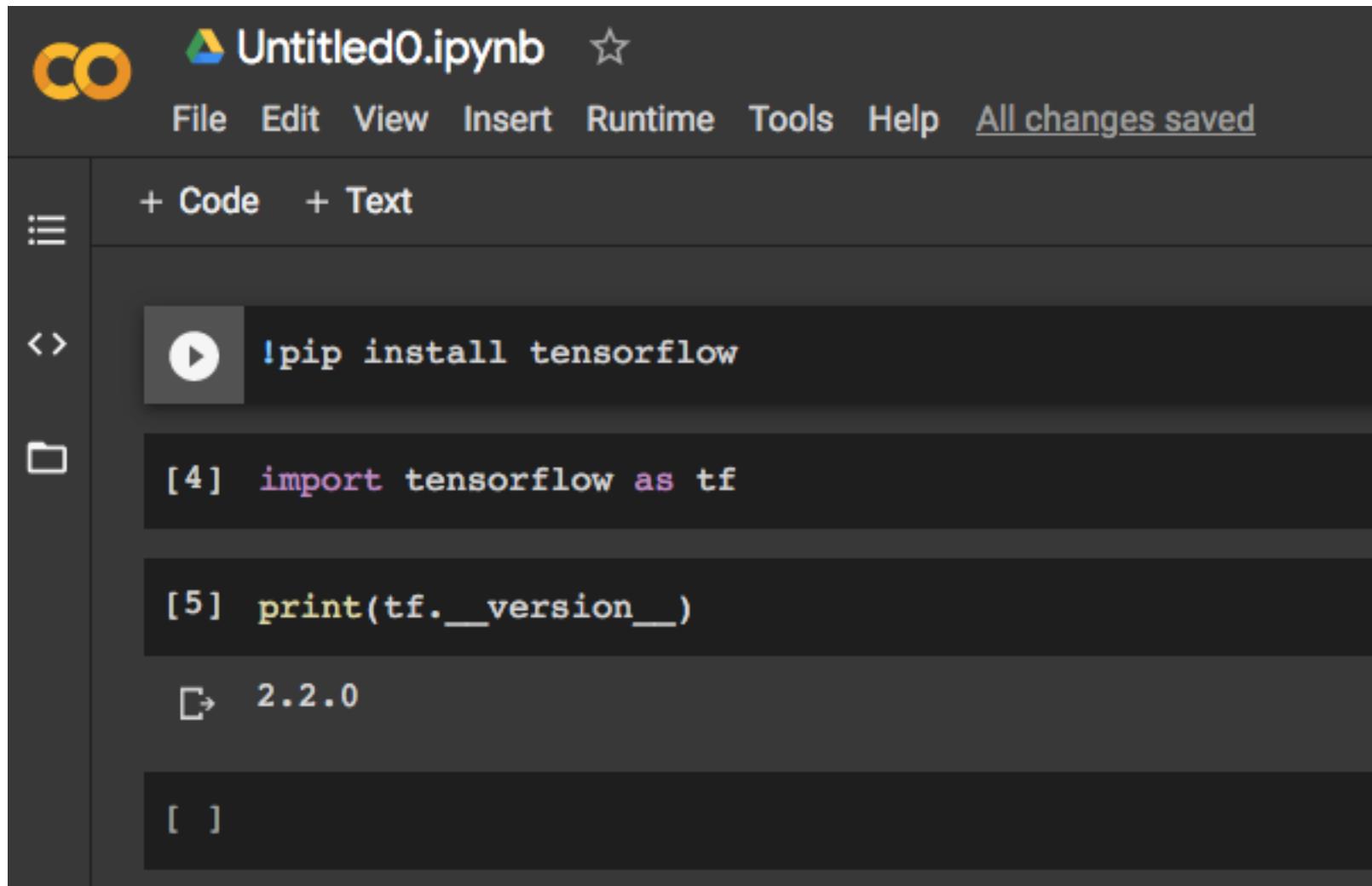
Start with Colab



The image shows a screenshot of the Google Colab interface. At the top, there's a logo consisting of two overlapping 'C's in orange and yellow, followed by the text "Untitled0.ipynb" with a star icon. Below the title is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", "Help", and "All changes saved". On the left side, there are three icons: a three-line menu icon, a code editor icon, and a file/folder icon. The main area contains a code cell with the command "[3] !pip install tensorflow". Below it, several lines of output are shown, all starting with "Requirement already satisfied": tensorflow in /usr/local/lib, astunparse==1.6.3 in /usr/loc, numpy<2.0,>=1.16.0 in /usr/lo, and gast==0.3.3 in /usr/local/lib.

```
[3] !pip install tensorflow
Requirement already satisfied: tensorflow in /usr/local/lib/
Requirement already satisfied: astunparse==1.6.3 in /usr/loc
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/lo
Requirement already satisfied: gast==0.3.3 in /usr/local/lib
```

Start with Colab



The screenshot shows the Google Colab interface. At the top, it displays the file name "Untitled0.ipynb" and a "All changes saved" status. Below the menu bar, there are buttons for "+ Code" and "+ Text". The main area contains four code cells:

- Cell 1: A command cell with a play button icon containing the command `!pip install tensorflow`.
- Cell 2: An output cell labeled [4] containing the Python code `import tensorflow as tf`.
- Cell 3: An output cell labeled [5] containing the command `print(tf.__version__)`.
- Cell 4: An output cell labeled [] indicating no output has been generated yet.

The output for Cell 3 is shown as 2.2.0.

Machine Learning With TensorFlow

- **tf.Session:**

- [Type](#): type
- A class for running TensorFlow operations.
- A `Session` object encapsulates the environment in which `Operation` objects are executed, and `Tensor` objects are evaluated.

For example:

```
# Build a graph.  
a = tf.constant(5.0)  
b = tf.constant(6.0)  
c = a * b  
sess = tf.Session() # Launch the graph in a session.  
print(sess.run(c)) # Evaluate the tensor `c`.
```

Basic Code Examples

- **EXAMPLE 1: CREATING A ONE DIMENSIONAL ARRAY**
- One dimensional tensor - normal array structure that has a set of values of the same type

```
>>> import numpy as np  
>>> tensor_1D = np.array([2.4, 5, 1.0, 18.49])  
>>> print(tensor_1D)
```



The screenshot shows a Jupyter Notebook interface. At the top, there is a toolbar with a yellow 'CO' icon, a file named '1D ARRAY.ipynb', and a star icon. Below the toolbar are menu options: File, Edit, View, Insert, Runtime, Tools, Help. Underneath the menu is a row with '+ Code' and '+ Text'. The main area contains a code cell with a play button icon, containing the Python code from above. Below the code cell is an output cell with a copy icon, displaying the resulting list: [2.4 5. 1. 18.49]. Further down, two more code cells are shown, each with a play button icon and their respective outputs: '2.4' and '1'.

```
import numpy as np  
tensor_1D = np.array([2.4, 5, 1.0, 18.49])  
print(tensor_1D)
```

```
[ 2.4 5. 1. 18.49]
```

```
>>>print tensor_1d[0]  
2.4
```

```
>>>print tensor_1d[2]  
1
```

Basic Code Examples

- **EXAMPLE 2 : CREATING A TWO DIMENSIONAL ARRAY :**
- Two dimensional Tensors Sequence of arrays are used for creating “two dimensional tensors”

```
>>>import numpy as np  
  
>>> tensor_2d=np.array([(1,2,3,4),(4,5,6,7),(8,9,10,11),(12,13,14,15)])  
  
>>> print(tensor_2d)  
[[ 1 2 3 4]  
 [ 4 5 6 7]  
 [ 8 9 10 11]  
 [12 13 14 15]]
```

Basic Code Examples

- **EXAMPLE 3 : CREATING MULTIDIMENSIONAL ARRAY**

```
import tensorflow as tf

import numpy as np

m1 = np.array([(3,3,3),(3,3,3),(3,3,3)],dtype='int32')

m2 = np.array([(2,2,2),(2,2,2),(2,2,2)],dtype='int32')

print (m1)

print (m2)

m1 = tf.constant(m1)

m2 = tf.constant(m2)

m_product = tf.matmul(m1, m2)

m_sum = tf.add(m1,m2)
```

```
[[3 3 3]
 [3 3 3]
 [3 3 3]]
 [[2 2 2]
 [2 2 2]
 [2 2 2]]
```

try it in class ...



Basic Code Examples

```
m_3 = np.array([(4,5,6),(3,4,1),(2,0,1)],dtype='float32')

print(m_3)

m_det = tf.matrix_determinant(m_3)

with tf.Session() as sess:

    result1 = sess.run(m_product)

    result2 = sess.run(m_sum)

    result3 = sess.run(m_det)

print(result1)

print(result2)

print(result3)
```

```
In [7]: print (result1)
...
...
...
...
[[18 18 18]
 [18 18 18]
 [18 18 18]]
[[5 5 5]
 [5 5 5]
 [5 5 5]]
-37.00004
```

try it in class ...

Basic Code Examples

CO matrixmultiplication.ipynb ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

```
▶ import tensorflow as tf
import numpy as np
m1 = np.array([(3,3,3),(3,3,3),(3,3,3)],dtype='int32')
m2 = np.array([(2,2,2),(2,2,2),(2,2,2)],dtype='int32')
print (m1)
print (m2)
m1 = tf.constant(m1)
m2 = tf.constant(m2)
m_product = tf.matmul(m1, m2)
m_sum = tf.add(m1,m2)
m_3 = np.array([(4,5,6),(3,4,1),(2,0,1)],dtype='float32')
print (m_3)
m_det = tf.matrix_determinant(m_3)
with tf.Session() as sess:
    result1 = sess.run(m_product)
    result2 = sess.run(m_sum)
    result3 = sess.run(m_det)
print (result1)
print (result2)
print (result3)
```

▶ [[3 3 3]
[3 3 3]
[3 3 3]]
[[2 2 2]
[2 2 2]
[2 2 2]]
[[4. 5. 6.]
[3. 4. 1.]
[2. 0. 1.]]
[[18 18 18]
[18 18 18]
[18 18 18]]
[[5 5 5]
[5 5 5]
[5 5 5]]
-37.000004

Basic Code Examples

- EXAMPLE 4 : MATRIX ADDITION :

```
import tensorflow as tf

c = tf.Variable([[4,5], [7,1]], name="matrix_c")

d = tf.Variable([[6,8], [5,2]], name="matrix_d")

init = tf.variables_initializer([c, d], name="init")

p = tf.add(c, d)

with tf.Session() as s:

    writer = tf.summary.FileWriter('graphs', s.graph)

    s.run(init)

    print(s.run(p))

writer.close()
```

Basic Code Examples

CO mymatrixaddition.ipynb ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

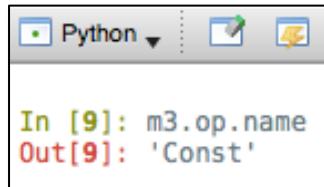
```
▶ import tensorflow as tf
c = tf.Variable([[4,5], [7,1]], name="matrix_c")
d = tf.Variable([[6,8], [5,2]], name="matrix_d")
init = tf.variables_initializer([c, d], name="init")
p = tf.add(c, d)
with tf.Session() as s:
    writer = tf.summary.FileWriter('graphs', s.graph)
    s.run(init)
    print(s.run(p))
writer.close()
```

⇨ [[10 13]
 [12 3]]

Basic Code Examples

- More examples:

```
1 ## Tensor Types
2 import tensorflow as tf
3 import numpy as np
4
5 # Define a 2x2 matrix in 3 different ways
6 m1 = [[1.0, 2.0], [3.0, 4.0]]                                     # <class 'list'>
7 m2 = np.array([[1.0, 2.0], [3.0, 4.0]], dtype=np.float32)          # <class 'numpy.ndarray'>
8 m3 = tf.constant([[1.0, 2.0], [3.0, 4.0]])                         # <class 'tensorflow.python.framework.ops.Tensor'>
9 print(type(m1))
10 print(type(m2))
11 print(type(m3))
12
13 # Create tensor objects out of various types
14 t1 = tf.convert_to_tensor(m1, dtype=tf.float32)                      # <class 'tensorflow.python.framework.ops.Tensor'>
15 t2 = tf.convert_to_tensor(m2, dtype=tf.float32)                      # <class 'tensorflow.python.framework.ops.Tensor'>
16 t3 = tf.convert_to_tensor(m3, dtype=tf.float32)                      # <class 'tensorflow.python.framework.ops.Tensor'>
17 print(type(t1))
18 print(type(t2))
19 print(type(t3))
```



... try it in class

A screenshot of a Jupyter Notebook cell showing the output of the previous code. It displays the type of each variable: 'list', 'numpy.ndarray', and three instances of 'tensorflow.python.framework.ops.Tensor'.

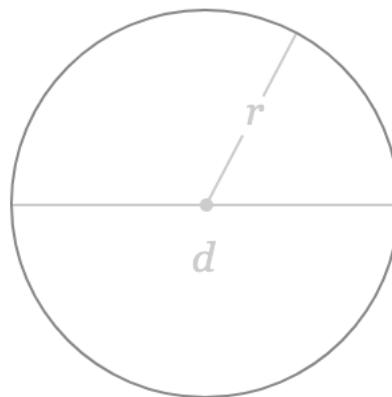
```
In [1]: (executing cell "Tensor Types" (line 2 of "types.py"))
<class 'list'>
<class 'numpy.ndarray'>
<class 'tensorflow.python.framework.ops.Tensor'>
<class 'tensorflow.python.framework.ops.Tensor'>
<class 'tensorflow.python.framework.ops.Tensor'>
<class 'tensorflow.python.framework.ops.Tensor'>
```

Class exercise:

Using TF, find the area of a circle, where:

- You use a **constant** for π (3.14 or 22/7)
- You ask the user for an input of a radius r = using **input**
- Use the formula for an area that is: $A = \pi * r^2$
- Print your result like this: “**Area of circle 78.5**”

$$A = \pi r^2$$



Solution: find the area of a circle

```
import tensorflow as tf

g = tf.Graph()

with g.as_default():
    pi = tf.constant(3.14) #define constant
    radius = tf.constant(5, tf.float32)
    area = tf.multiply((radius ** 2 ), pi)

# launch the graph in a session
with tf.compat.v1.Session(graph = g) as sess:
    print("Area of circle ",sess.run(area))

#OUTPUT ----- Area of circle 78.5
```

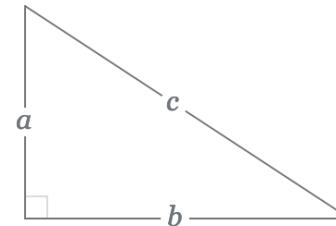
Basic operations

HW1:

Find the area of a right triangle with the following parameters:

a Leg

b Leg



Solution

$$A = \frac{ab}{2} = \frac{3 \cdot 5}{2} = 7.5$$