

# Experience with Keras (Theano and TensorFlow)

---

Michela Paganini  
Yale University



**HEP Software Foundation Workshop**  
**May 3, 2016 - LAL Orsay**

# **PRESENTATION OUTLINE**

```
graph LR; A((WHAT IS KERAS)) --- B((WHY KERAS)); B --- C((WHAT DRIVES KERAS)); C --- D((WHAT'S IT LIKE TO USE KERAS));
```

**WHAT IS  
KERAS**

**WHY  
KERAS**

**WHAT  
DRIVES  
KERAS**

**WHAT'S IT  
LIKE TO USE  
KERAS**

# What is Keras

- Modular, powerful and intuitive Deep Learning python library built on **Theano** and **TensorFlow**
- Minimalist, user-friendly interface
- CPUs and GPUs support
- Open-source, developed and maintained by a community of contributors, and publicly hosted on GitHub
- Extremely well documented, lots of working examples
- Very shallow learning curve  
—> it is by far one of the best tools for both beginners and experts
- *"Developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research."*

# What Makes Keras Great

1. BEAUTIFUL MINIMALIST USER-FRIENDLY

2. POWERFUL ROBUST OPTIMIZED

# KERAS BACKENDS

- Keras doesn't itself handle any tensor ops

- Relies on tensor manipulation libraries (Theano & TensorFlow)

- Connects with both of them via the abstract Keras backend (<http://keras.io/backend/>)

## BACKEND ENGINES



## UNIFIED FRONTEND



# Theano & TF

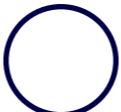




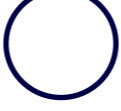


- High level wrappers around C++ —> let you call into the C++ efficiently while using syntax of higher level programming languages
- Theano and TF are in many ways very similar
  - Both let you define arbitrary functions on tensors and compute their derivatives easily
  - Not really Deep Learning packages
- Theano has been around for longer and it's a more stable package, but it's an academic project
- TF is backed by Google and have the advantage of natively built in support for distributed systems

# Theano

"Theano is a linear algebra compiler that optimizes a user's symbolically-specified mathematical computations to produce efficient low-level implementations."

"It can be used to symbolically define mathematical functions, automatically derive gradient expressions, and compile these expressions into executable functions"

# Properties of Theano

		Dynamic C code generation
Stability optimization		
		Execution speed optimization
Parallelism on CPU/GPU		
		Asynchronous calls on GPU
Lazy evaluation		
		Unit tests with Travis-CI
Symbolic differentiation		



# TensorFlow

- Evolution from 1st generation DistBelief (NIPS 2012)
- Meant for deployment on heterogeneous systems on a broad range of platforms including large-scale distributed systems
- Synchronous and asynchronous training on different devices
- Abstracts away underlying devices/computational hardware



# TensorFlow vs Numpy

Similar syntax  
+ some TF-specific entities (e.g. *session*)

Different functionality:

Numpy is doing  
operations in memory

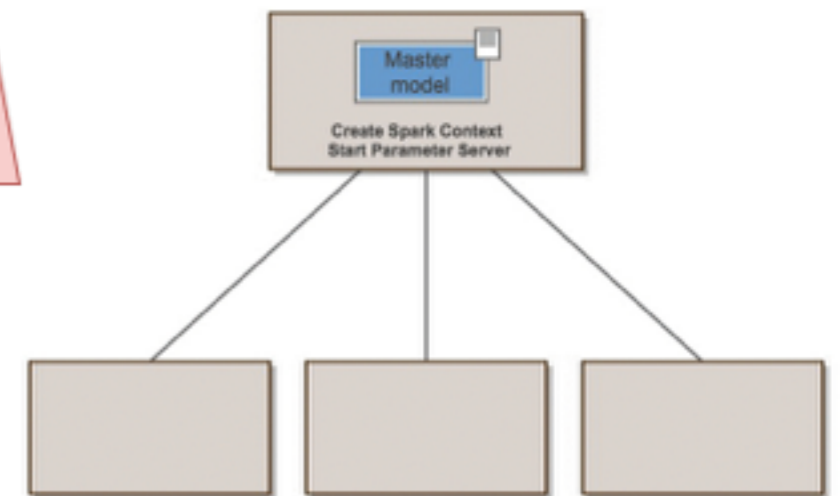
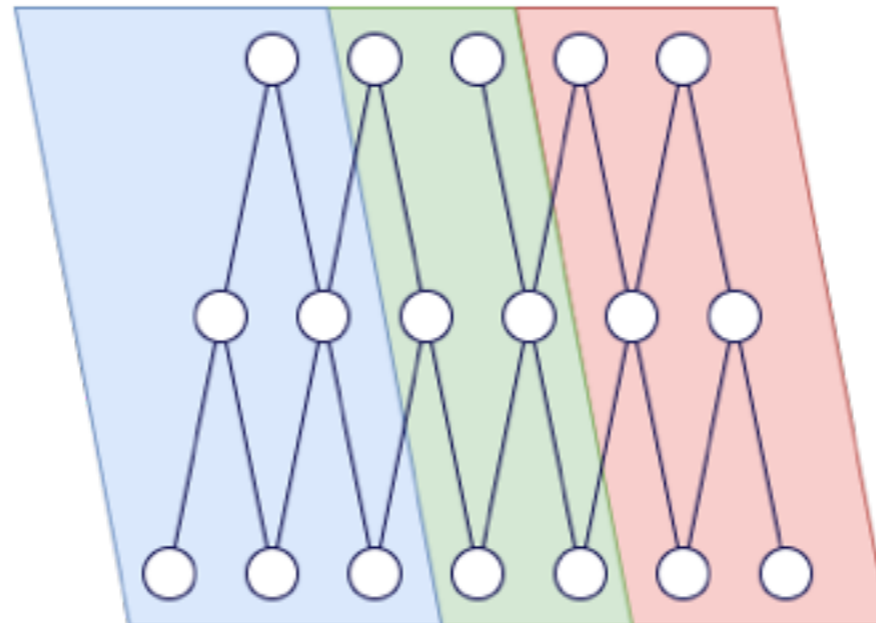
TF is symbolically  
describing operations,  
compiling and  
executing them in a  
session to get a  
numerical result  
(*feed & fetch*)

# TENSORFLOW GRAPH

- Execution core directly in C++ with very low overhead
- Just like Theano, it allows us to define NNs in a programmatic way
- A TensorFlow computation is described by a directed **graph**, which is composed of a set of **nodes**
- Clients typically construct a computational graph using one of the supported front-end languages (C++ or Python)
- Each node represents the instantiation of an **operation**
- An operation represents an abstract computation (e.g., “matrix multiply”, or “add”)
- A **kernel** is a particular implementation of an operation that can be run on a particular type of device (e.g., CPU or GPU)
- Client programs interact with the TensorFlow system by creating a **session**
- Computations represented as a dataflow graph where **tensors** flow along the graph edges

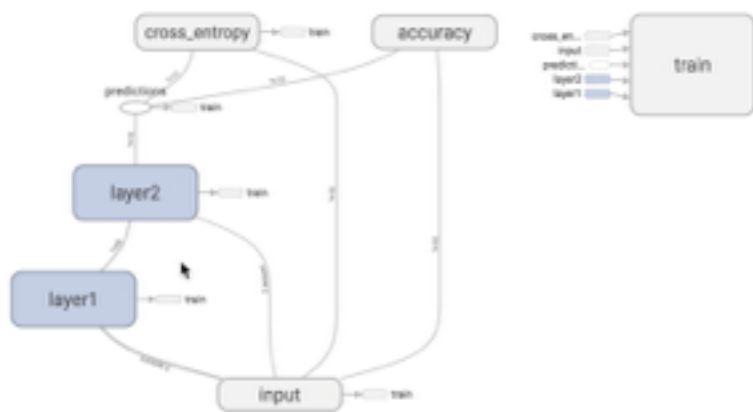
# TensorFlow Pro

1st device 2nd device 3rd device



Main Graph

Auxiliary nodes



**Visualization Module:  
TensorBoard**

**Model Parallelism:  
Subgraph Computation**

**Data  
Parallelism**

**What's it like to use  
Keras?**

# iPython Notebook



<http://nbviewer.jupyter.org/github/mickypaganini/IPNN/blob/master/TF/DummyNet.ipynb>

# Keras Support

- Google Group: <https://groups.google.com/forum/#!forum/keras-users>
- Github issues: <https://github.com/fchollet/keras/issues>
- Slack: <https://kerasteam.slack.com>
- StackOverflow: <http://stackoverflow.com/questions/tagged/keras>

# CERN INTEGRATION

The screenshot shows the GitHub repository page for 'dguest / lwtnn'. At the top, there are navigation links for 'Code', 'Issues', 'Pull requests', 'Wiki', 'Pulse', and 'Graphs'. Below this, the repository name 'light NN client' is displayed. A progress bar shows 121 commits, 3 branches, 0 releases, and 2 contributors. The current branch is 'rnn', and there are buttons for 'New pull request', 'New file', 'Upload files', 'Find file', 'HTTPS', and 'Download ZIP'. The repository is 44 commits ahead of master. A list of files and their commit messages is shown below:

File	Commit Message	Time
dguest	Break embedding config into its own object	Latest commit 8369f66 an hour ago
converters	Add embedding info to NN layer config	2 hours ago
data	Add some handling for NaN values	6 months ago
include/lwtnn	Break embedding config into its own object	an hour ago
scripts	Add more documentation	3 months ago
src	Break embedding config into its own object	an hour ago
.gitignore	Refactor	a month ago
README.md	Major cleanup for Julian's format	3 months ago
makefile	Turn on pedantic warnings	2 months ago
test-rn.sh	Rename all executables to share the 'lwtnn' prefix	3 months ago

NN CLASS IN ATHENA

TF C++

TENSORFLOW C++ WITH ATHENA



# Questions

