

# Alternate Universe Nio

Featuring:

Google Colaboratory

Spyder

Apache Zeppelin

By

Anusha Jangalapalli

Shenghao Huang

Sumalatha Konjeti



# Google Colaboratory

- Created by Google 2017
- GPU/TPI Support
- Google Drive
- Code anywhere



# Special Features!



AI Hub



## Tensorflow with GPU

This notebook provides an introduction to computing on a [GPU](#) in Colab. In this notebook you will connect to a GPU, and then run some basic TensorFlow operations on both the CPU and a GPU, observing the speedup provided by using the GPU.

## Enabling and testing the GPU

First, you'll need to enable GPUs for the notebook:

- Navigate to Edit→Notebook Settings
- select GPU from the Hardware Accelerator drop-down

Next, we'll confirm that we can connect to the GPU with tensorflow:

```
[ ] tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
TensorFlow 2.x selected.
Found GPU at: /device:GPU:0
```

DEVELOPMENT



DATA & MODEL  
STORAGE



REPOSITORY  
REVISION



# Jupyter Notebook!

- Kernel for multiple languages
- Data Visualization
- Markdown Cells
- Widgets





# Colab vs Jupyter

- EASY
  - No need to create virtualenv(Conda)
  - Kaggle Import
  - Organization
  - IDE Environment
- Supports many kernels
  - Easy Image sharing
  - Documentation

The image shows a sidebar titled "Table of contents" with a search icon and a close button. It lists the following sections:

- Copyright 2019 The TensorFlow Authors. Licensed under the Apache License, Version 2.0 (the "License");
- Text generation with an RNN
  - Setup
    - Import TensorFlow and other libraries**
    - Download the Shakespeare dataset
    - Read the data
  - Process the text
    - Vectorize the text
    - The prediction task
    - Create training examples and targets
    - Create training batches
  - Build The Model
  - Try the model
  - Train the model
    - Attach an optimizer, and a loss function
    - Configure checkpoints
    - Execute the training
  - Generate text
    - Restore the latest checkpoint
    - The prediction loop
  - Advanced: Customized Training

At the bottom, there is a "Section" header with a small icon.

# CONCLUSION



# SPYDER

Developed By Pierre Raybaut, Oct 2009

## Features:

- Code Auto Completion
- Syntax Highlighting
- Debugging
- Great visual representation (GUI)



# Special Features

- Its an IDE/notebook developed for programming written in Python for Python.
- It mainly comes with Editor
- IPython console
- Variable explorer
- Debugger
- Features like Syntax highlighting and better Graphical User Interface (GUI) are especially useful to work on projects with large code.
- Easy to debug and trace the values in real time using variable explorer



# Spyder

- Variable Explorer( giving suggestions about name, size , type and value of the objects)
- Debugging the code
- Notebook have collapse and expand code feature
- Less computational time
- Great for big projects

# Jupyter

- Data Analysis
- Good to show visualization
- Great for small projects
- Slower runtime
- Good to analyze data
- No IDE integration
- Not good for large projects because of no built in debugger

# Contrast from jupyter

- Run code line by line
- It has PyQt5 library support
- Excellent variable explorer



# Places to use instead of Jupyter:

- When working with large projects
- Unittesting
- GUI
- Debugging

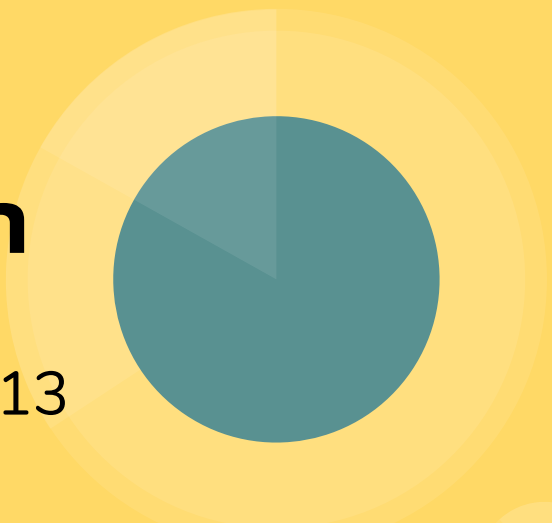


Do You  
Recommend?



# Zeppelin

- Who Built it --> Apache in 2013
- What is Apache Zeppelin?
  - A web-based notebook that enables interactive data analytics.
  - Multiple language backend embedded.
  - Supports Single user and multi-user deployment.



# Special features



- Supports multiple languages like r, Scala, sql and many more in one dashboard.
- Uses interpreter for each paragraph.
- Visualize same data in multiple formats. For example bar chart to pie chart with no additional development and with just one click.
- Runs in parallel processing mode to make everything faster.
- Self-describing reports

# Comparison between Jupiter / Zeppelin

## Jupiter



- In-line code execution using blocks
- Multiple kernel support in different notebooks
- In-line graphing is supported

## Zeppelin



- In-line code execution using paragraphs
- Multiple interpreter in same dashboard.
- In-line graphing is supported

# Jupyter Vs Zeppelin

## Jupyter



- Good for Data scientists
- Strong community
- Lots of examples
- One kernel for interpretation in one notebook
- Must define type of chart in the block / cell
- More extensions

## Zeppelin



- Good for deep data analytics
- Small community
- Few but very well written examples
- Supports multiple interpreters in one dashboard.
- Standard zeppelin provides changing to different types of charts



# Why Zeppelin over Jupyter

- Security --> Allows flexible security configurations for the end users.
- When multiple plots for charting is required.
- Multiple languages are supported in one dashboard.
- Much simpler data visualization.

# Final showdown – Who wins???

- It depends
- If security is of very critical --> You will go for Zeppelin.
- Need to use wide range of languages --> you will go for Jupyter. Jupyter supports more than 85 languages whereas Zeppelin supports 20.
- Need different interpreters / different types of charts in one dashboard --> you will go for Zeppelin.
- Flexibility is of importance --> Jupyter.