DATASCIENCE.COM

Learn to be a painter using Neural Style Painting

July 7, 2017

ABOUT ME

Pramit Choudhary

@MaverickPramit

https://www.linkedin.com/in/pramitc/

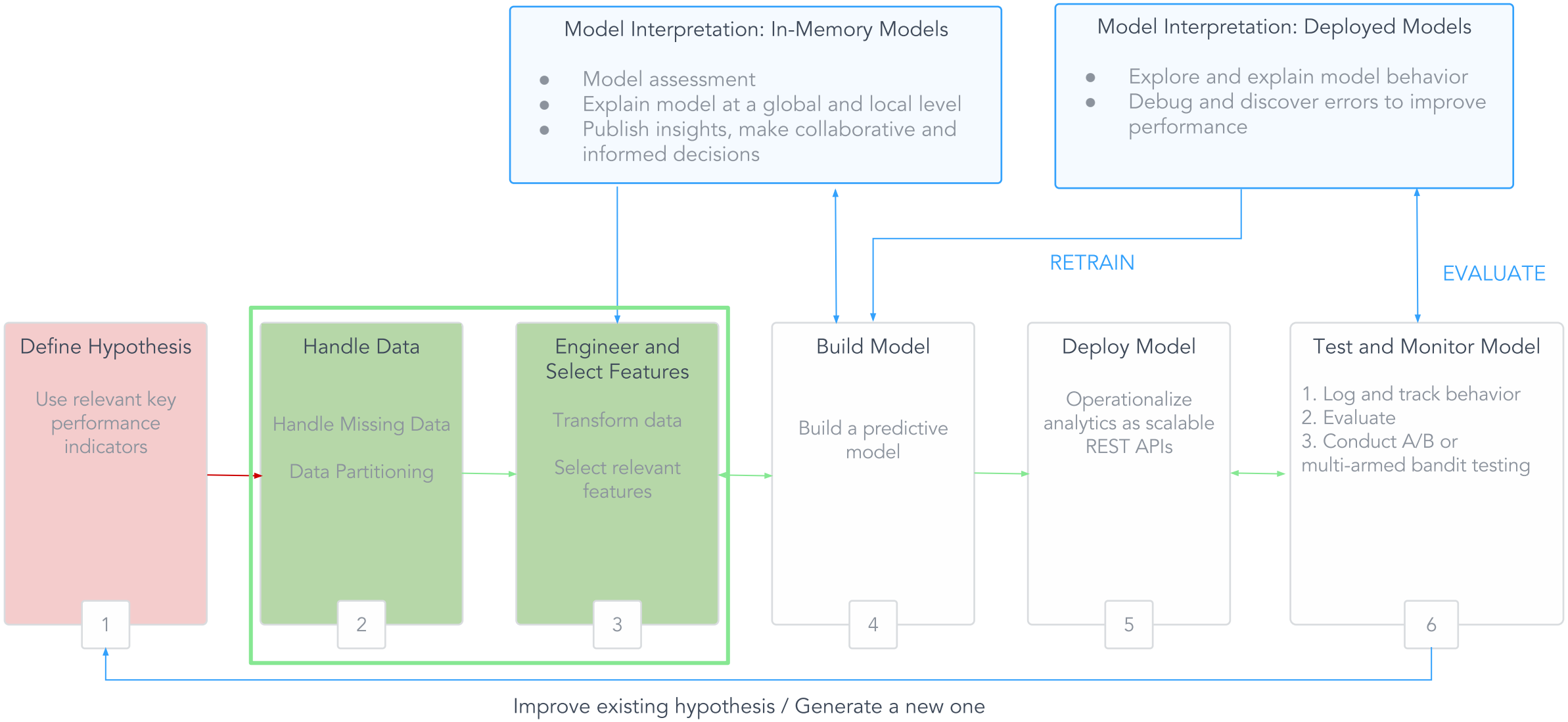
https://github.com/pramitchoudhary

I am a Lead data scientist at DataScience.com. My focus is on effective ways of optimizing and applying classical (Machine Learning) and Bayesian design strategy to solve real-world problems.

AGENDA

- Recap Analytical workflow and Deep Learning
- Explain Convolutional Neural Networks
- Introduce TensorFlow, Livy and sparkmagic kernel
- Style Transfer Algorithm
- Scalable parameter tuning for semantic style transfer
- QnA

MACHINE LEARNING WORKFLOW

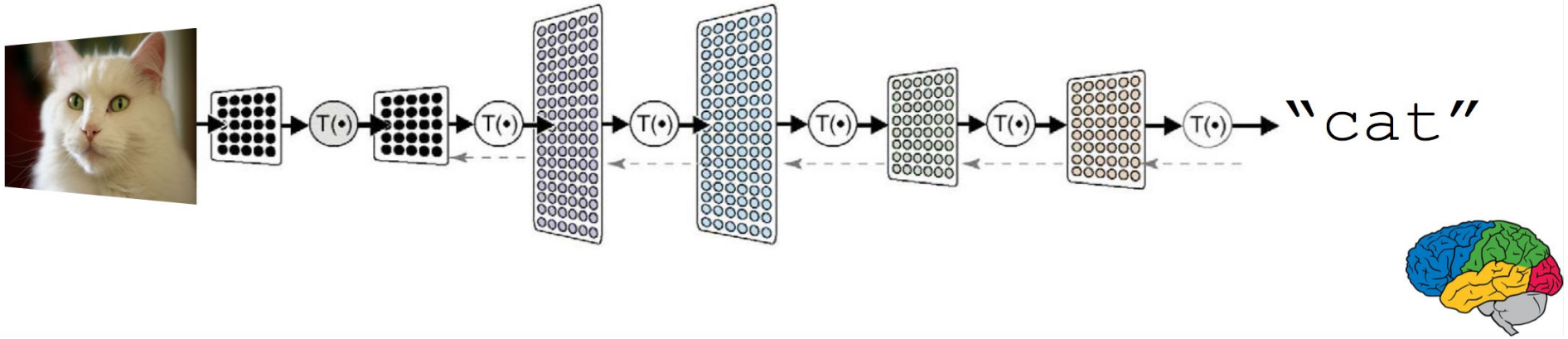


THE UNIVERSE IS CHANGING

- More connected devices
- More complicated and heterogeneous available data
- Need for better and automated ways to learn nonlinear relationships between features
- Computation Resources are getting cheaper and easily accessible

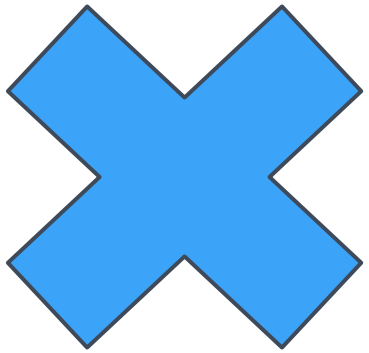
DEFINE DEEP LEARNING

- Is an ANN with multiple hidden layers between the input and output layers
- Uses a hierarchy of multiple layers of nonlinear processing units for feature extraction and transformation
- Is a collection of simple, trainable mathematical function $Y = g(W.X + b)$
- Usage continues to accelerate - image understanding, self driving vehicles, and more



PATTERN RECOGNITION IN VISUAL IMAGERY

Pattern to identify

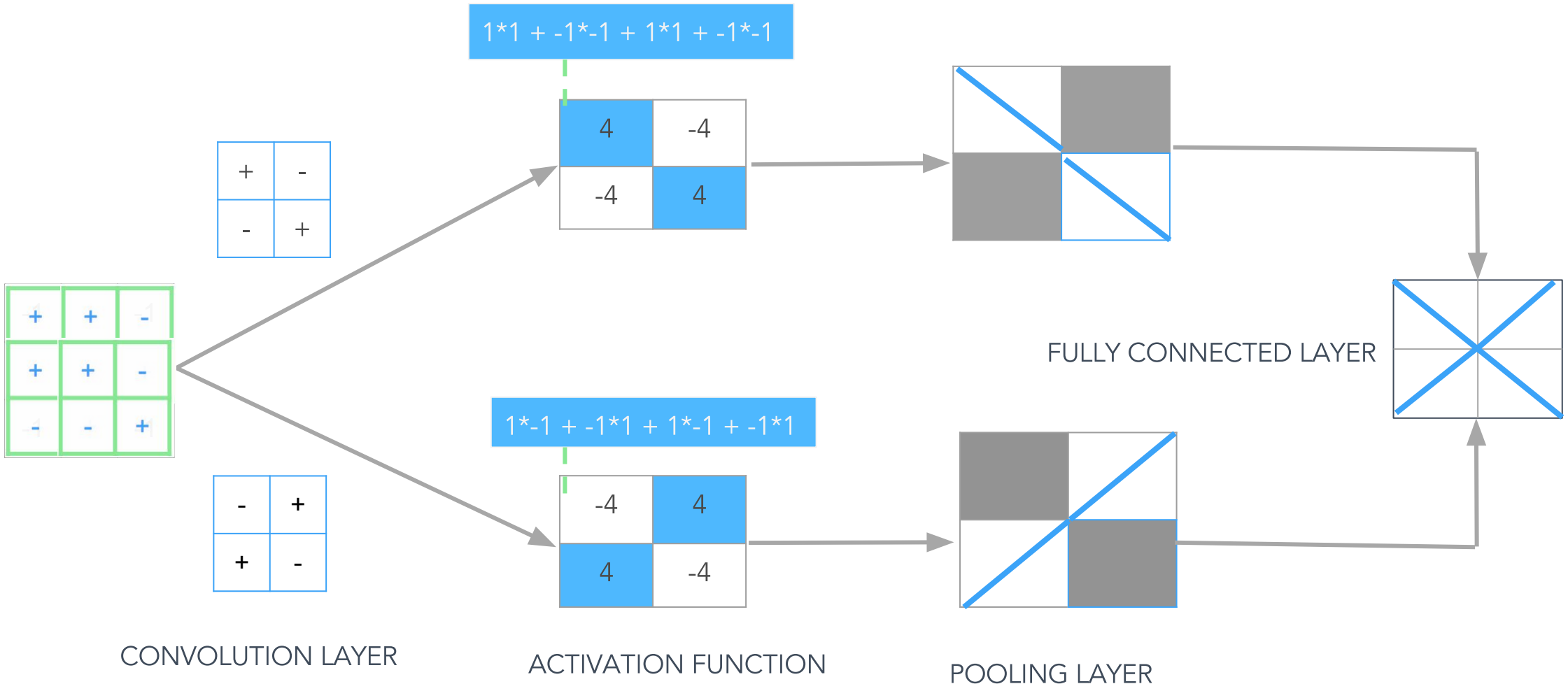


Represented as

-1	1	-1
1	-1	1
-1	1	-1

PATTERN RECOGNITION IN VISUAL IMAGERY

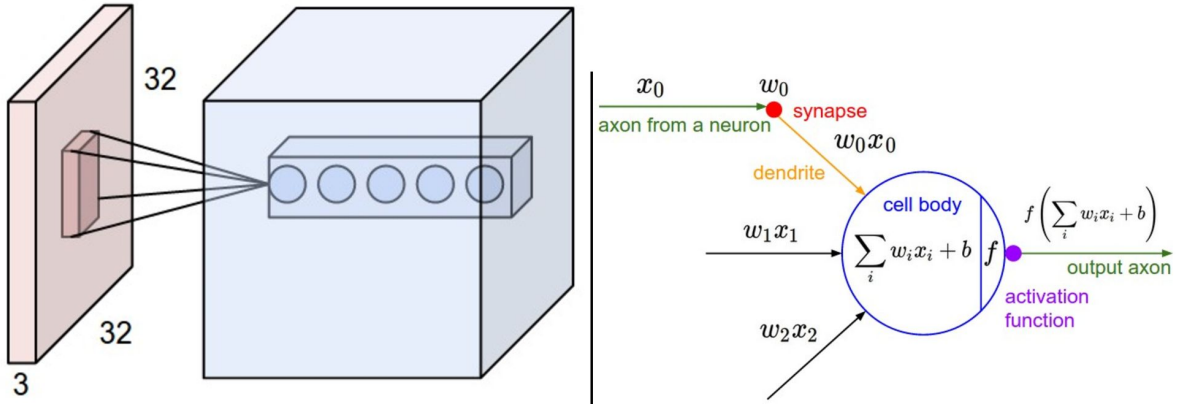
*Courtesy: Luis Serrano



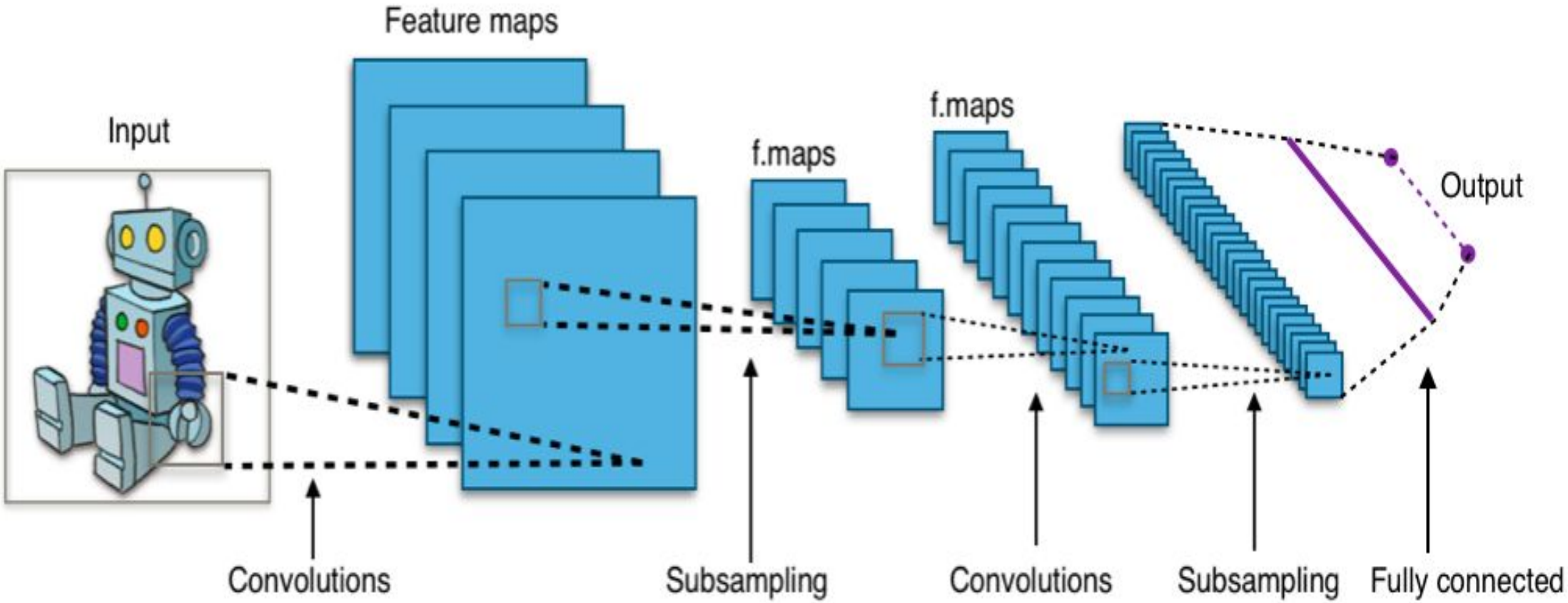
DEFINE CONVOLUTIONAL NETWORK

- Is a type of feed-forward artificial neural network which have proven successful in detecting patterns in the visual imagery

- Is based on the following principles
  - **Spatial Arrangement:**
    - Each layer has neurons arranged in 3 dimension
    - E.g. 5 pixels width and height, and depth 3(representing the color channels)
  - **Local Connectivity**
    - Neurons are connected to a small region
  - **Shared weights**
    - All neurons for a given convolutional layer respond to the same feature weights



CONVNET ARCHITECTURE



*Courtesy: wikipedia

ABOUT TENSORFLOW

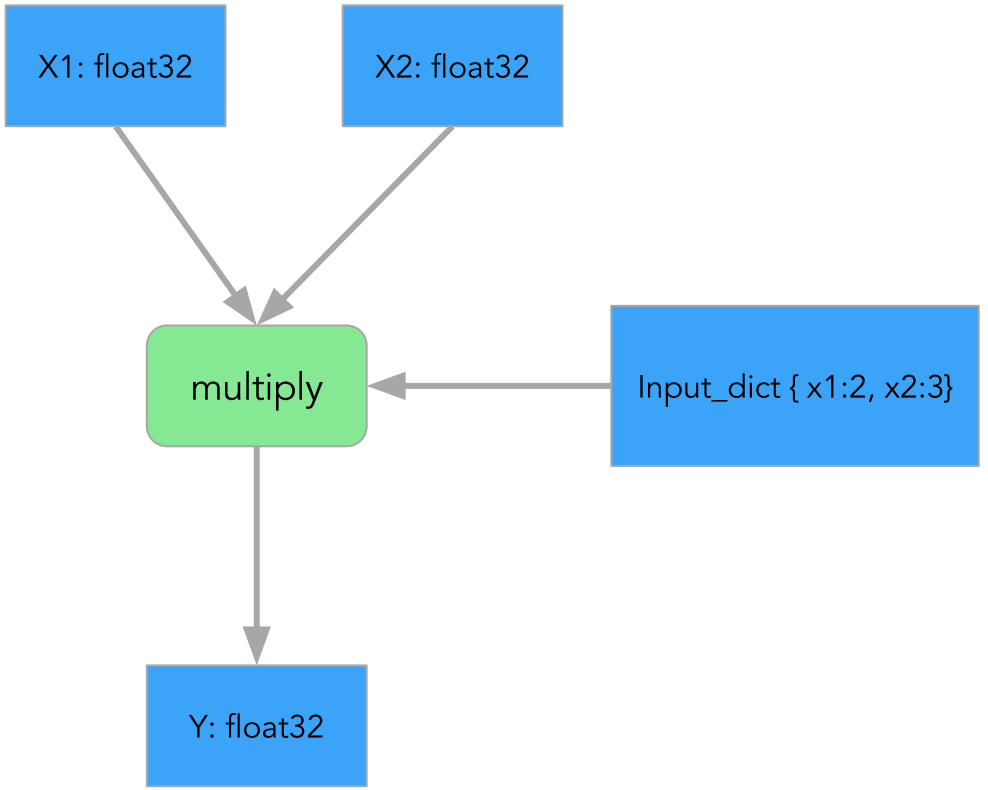
- Is great for Deep Learning
- Can also be used for general purpose numerical computation
- Core is C++, results in faster computation
- Python and c++ front enables low overhead

TENSORFLOW

- Computation is expressed as dataflow graph

```
# Define a placeholder for data
x1 = tf.placeholder(tf.float32)
x2 = tf.placeholder(tf.float32)

# specify an operation
y = tf.multiply(x1, x2)

input_dict = {x1:2, x2:3}

with tf.Session() as sess:
    print(sess.run(y, input_dict))
```

```
6.0
```

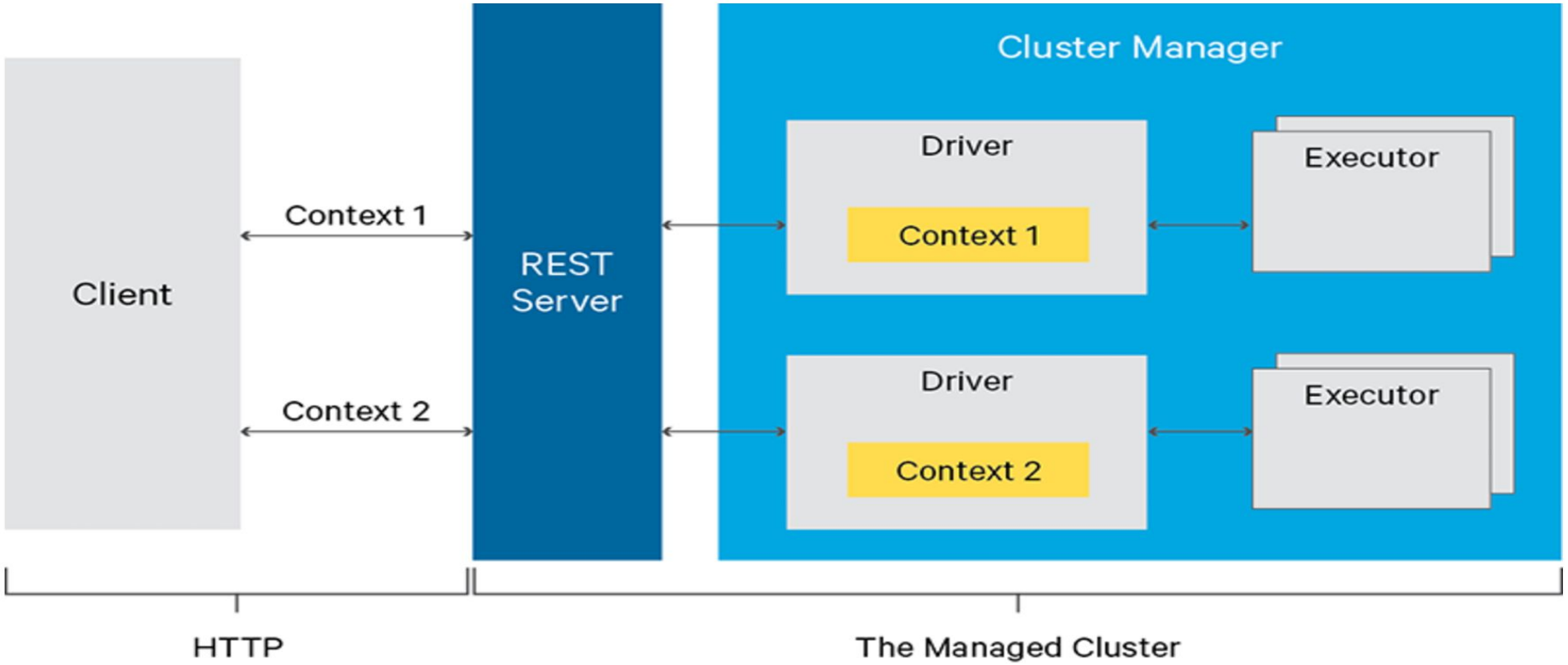


LIVY AND SPARKMAGIC

- Livy is a service that manages long running sparkcontexts in a cluster (Yarn/Mesos)
  - Livy enables easy interaction with Spark cluster over a REST Interface
  - Allows sharing cached RDDs or Dataframes across multiple jobs and clients
  - Allows managing multiple SparkContexts
  - Jobs can be submitted as precompiled jars, snippets of code, or via Java/Scala client API

- Sparkmagic provides a set of tools for interactively working with remote Spark clusters through Livy
  - Also includes a set of magics for interactively running Spark code in multiple languages

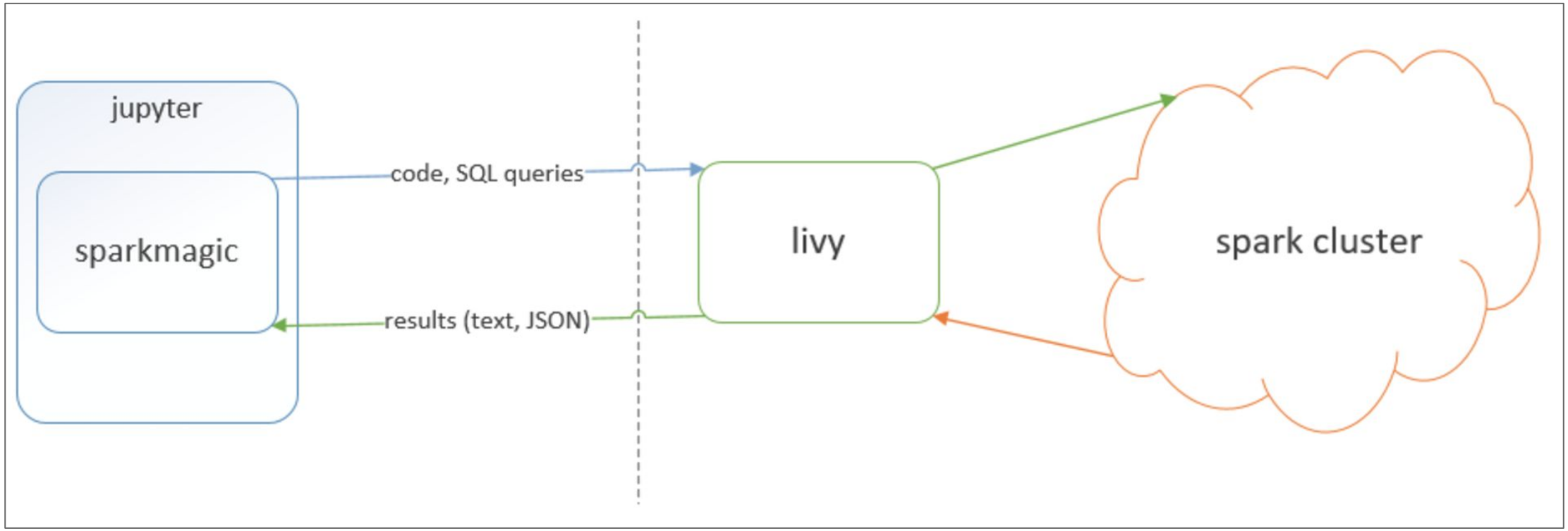
LIVY ARCHITECTURE



Livy Architecture

*Courtesy: Cloudera

SPARKMAGIC + LIVY = DATASCIENCE



PAINT LIKE PAINTERS

- Image Style Transfer : *Work by Gatys et al.*
  - One is provided with 2 images
  - Generate a third image with **semantic** content of the first image merged with the texture and style of the second image

Content extractor using higher layers of the network

Merger

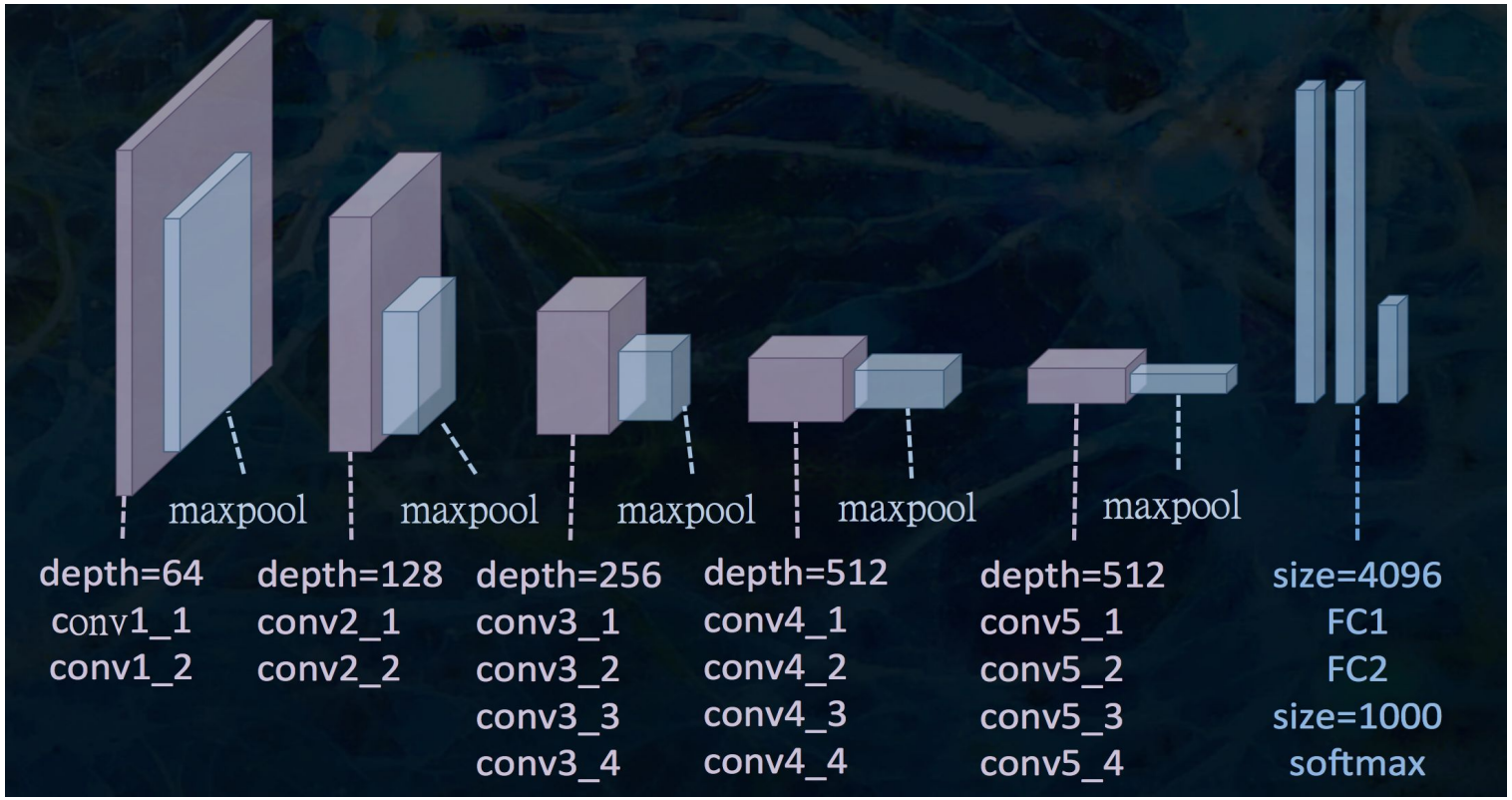
Style extractor using the lower layer of the network

VGG 19

- Result of the work by Karen Simonyan & Andrew Zisserman, 2014
- Is a pretrained model trained on ImageNet database
- ImageNet database used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)
- VGG-19 is trained over a million images to classify 1000 object categories
- Is capable of recognizing feature representations of wide range of images

- VGG 19 is composed of
  - 16 Convolutional Layers (width=3, height=3)
  - 5 max pooling layers(width=2, height=2)
  - 3 fully connected layers

VGG 19 REPRESENTATION



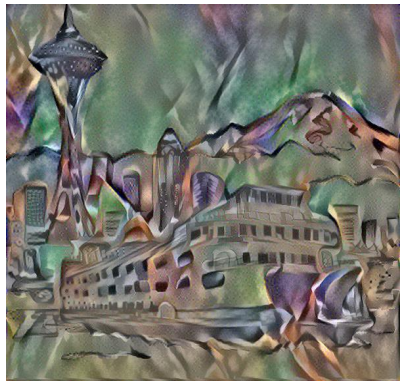
wget http://www.vlfeat.org/matconvnet/models/imagenet-vgg-verydeep-19.mat

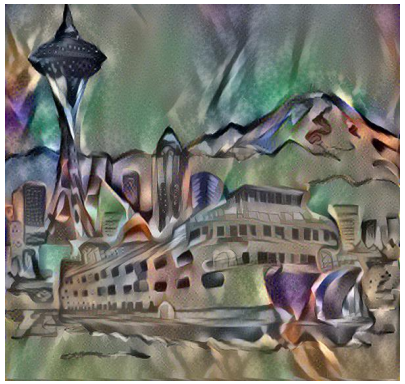
```
# One can peek into the weights using loadmat
from scipy.io import loadmat
import pandas as pd

file_handle = loadmat('../imagenet-vgg-verydeep-19.mat')
print(pd.DataFrame(file_handle['layers'])[1])
```

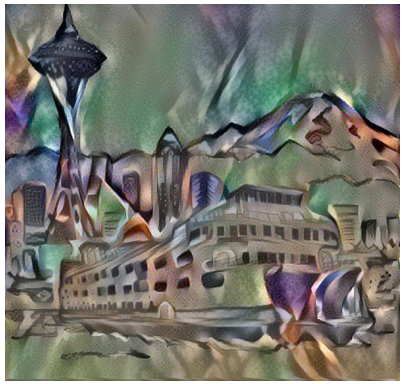
LETS PAINT

Content Input

Style Input







=

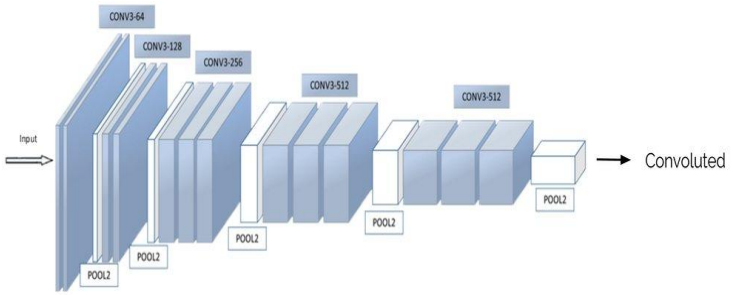
OPTIMIZING ON LOSS FUNCTION

Space Needle

Picabia's Udnie

Conv4_2
Conv5_2

VGG 19



Conv1_1
Conv2_1
Conv3_1
Conv4_1
Conv5_1

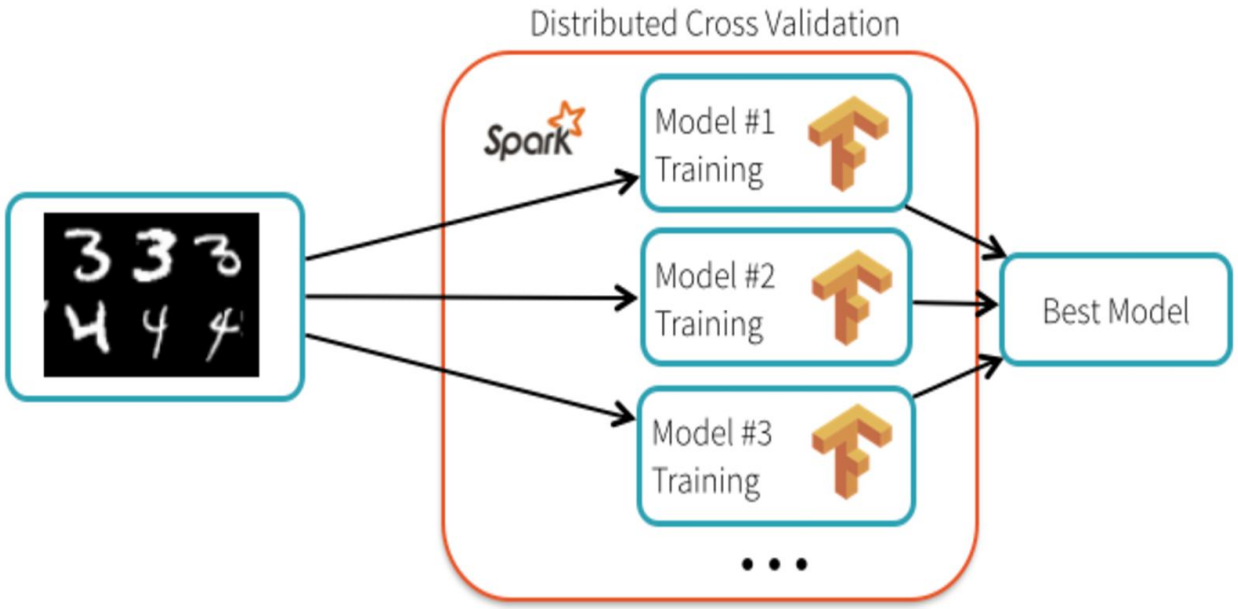
$$L_{total}(p, a, x) = \alpha L_{content}(p,x) + \beta L_{style}(a,x) + total\_variational\_denoising$$

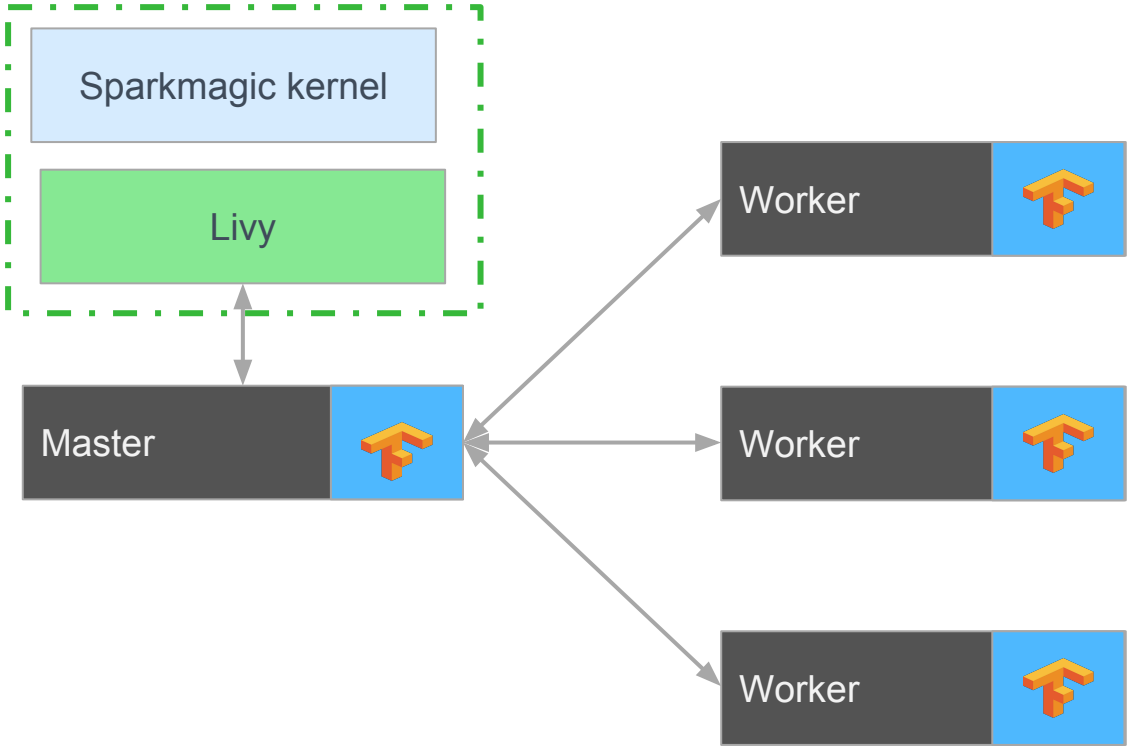
SPARK + TENSORFLOW

- Is there a need to combine Spark and TensorFlow ?

- Spark could be used for
  - **Hyperparameter tuning**
    - Find the best set of parameters for the deep learned model to lower the error rate
    - E.g Number of neurons in each layer, learning rate for optimization, decay rate
  - Helping **reduce the train time**
  - **Deploying models at scale**
    - Use the trained model to deploy and score large amount of data

SPARK + TENSORFLOW ARCHITECTURE





PARAMETER TUNING

- STEP 1: Get the client ready to interact with the Spark Cluster (Yarn/Mesos)

```
%%configure -f
{"name":"", "executorMemory": "4G", "executorCores":4}
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
16	application_1497992685935_0129	pyspark3	idle	Link	Link	✔

```
SparkSession available as 'spark'.
```

Current session configs: `{'kind': 'pyspark3', 'name': '', 'executorMemory': '4G', 'executorCores': 4}`

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
16	application_1497992685935_0129	pyspark3	idle	Link	Link	✔

PARAMETER TUNING

- STEP 2: Specify the parameters

In [7]:
```
import itertools

# A simple example
epochs = [20]
learning_rate = [0.1, 0.0001]
# Build a list of list of hyperparameter values
sample_hyperparams = [learning_rate, epochs]
sample_combinations = [item for item in itertools.product(*sample_hyperparams)]
print('Simple combination for a standard CNN skeleton for Image Classification ...')
print(sample_combinations)


# some parameters for playing with neural style painting
iterations = [500]
learning_rate = [1e1, 0.1]
style_wt = [5e1, 5e2]
content_wt_blend = [1, 0.5]
style_blend_weights = [0.4, 0.5, 0.8]
preserve_colors = [True, False]
# Build a list of list of hyperparameter values
hyperparams = [iterations, learning_rate, style_wt]
combinations = [item for item in itertools.product(*hyperparams)]
print('Some combinations for Neural Style Painting ...')
print(combinations)
```

```
Simple combination for a standard CNN skeleton for Image Classification ...
[(0.1, 20), (0.0001, 20)]
Some combinations for Neural Style Painting ...
[(500, 10.0, 50.0), (500, 10.0, 500.0), (500, 0.1, 50.0), (500, 0.1, 500.0)]
```

PARAMETER TUNING

- STEP 3: Define the command or function to be executed

```
In [ ]: cmd = 'python "../neural_style.py"
"--content" "../neural-style/examples/<input-content>.jpg"
"--styles" "../neural-style/examples/<input-style>.jpg"
"--output" "/tmp/results/output_{}.jpg" "--iterations" {} "--network"
"../vgg_network/imagenet-vgg-verydeep-19.mat" "--learning-rate" {}
"--style-weight" {} "--content-weight" {} "--content-weight-blend" {}'
```

```
In [ ]: rdd_list = sc.parallelize(combinations)
cmd_rdd = rdd_list.map(lambda x:
                       cmd.format((x[1],x[0],x[2], x[3], x[4]),
                                  x[0], x[1], x[2], x[3], x[4]))
cmd_rdd.collect()
```

QUIZ

A = [x1, x2]
B = [x3, x4, x5]
C = [x6, x7]

Combinations = ???

Answer = 2 * 3 * 2 (12)

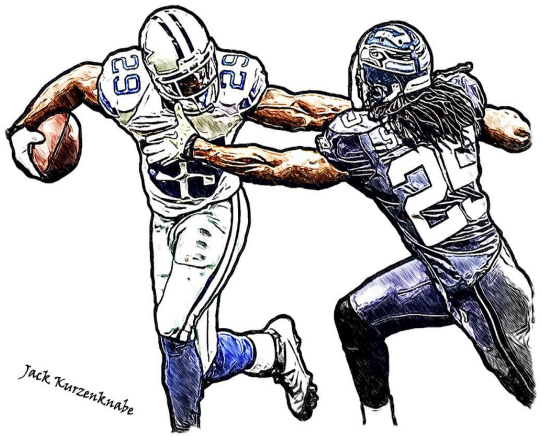
PARALLELIZE TUNING

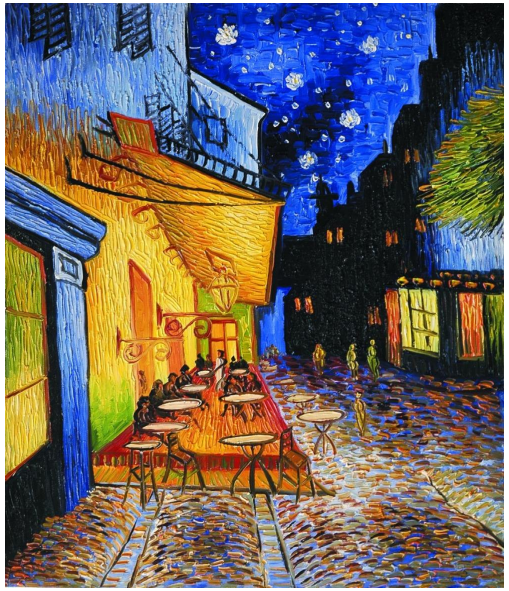
In [ ]:

```
import subprocess
import shlex
from __future__ import print_function

# A simple function to extract model metric for evaluation
def get_metric(input_str):
    pattern = re.compile(r"Accuracy\:.*")
    return pattern.search(input_str).group(0)

# A simple function to do parallel computation
def launch_job(cmd_str):
    call_params = shlex.split(cmd_str)
    proc = subprocess.Popen(call_params, stdout=subprocess.PIPE,
            stderr=subprocess.PIPE)
    out, err = proc.communicate()
    return out

results = cmd_rdd.map(lambda k: get_metric(launch_job(k)))
results.collect()
```

GRAPHIC CONTENT GENERATION

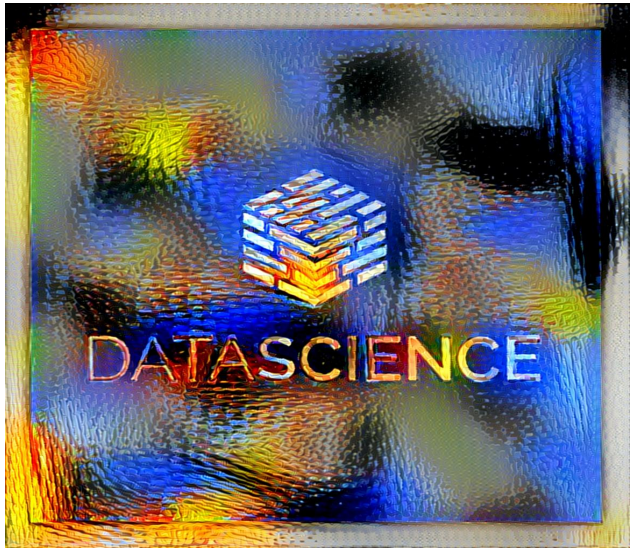
SPORTS CONTENT GENERATION



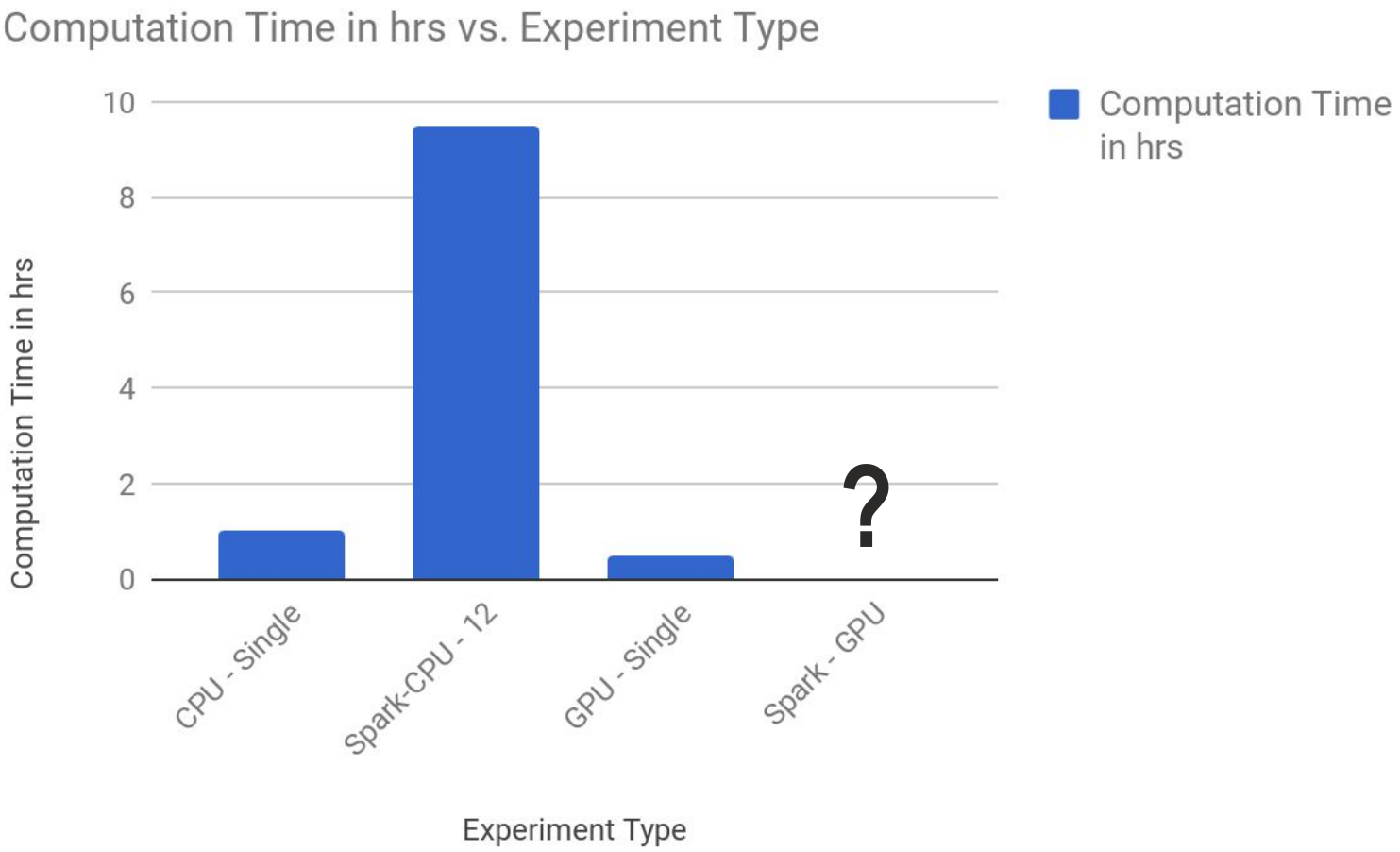


BRANDING CONTENT GENERATION

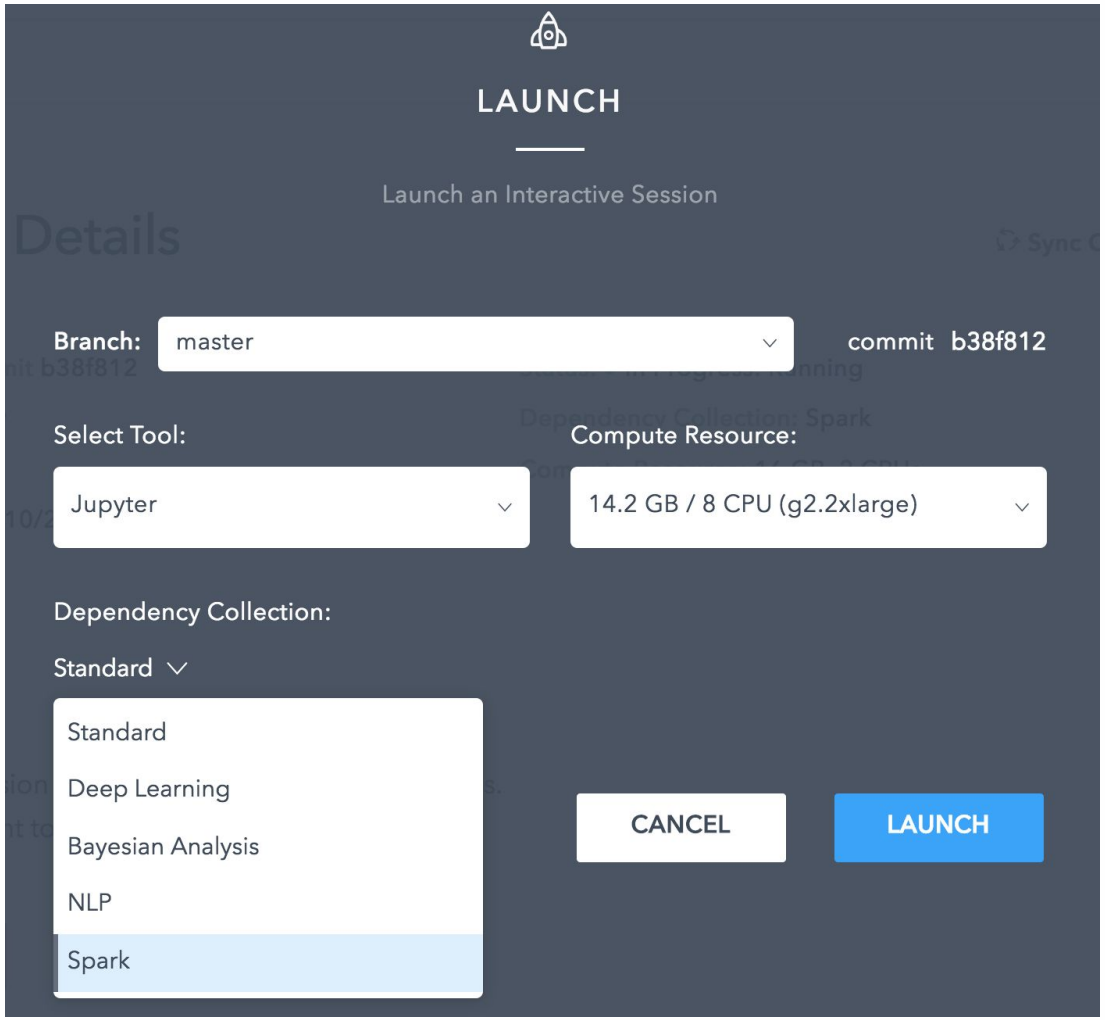
DATASCIENCE.COM

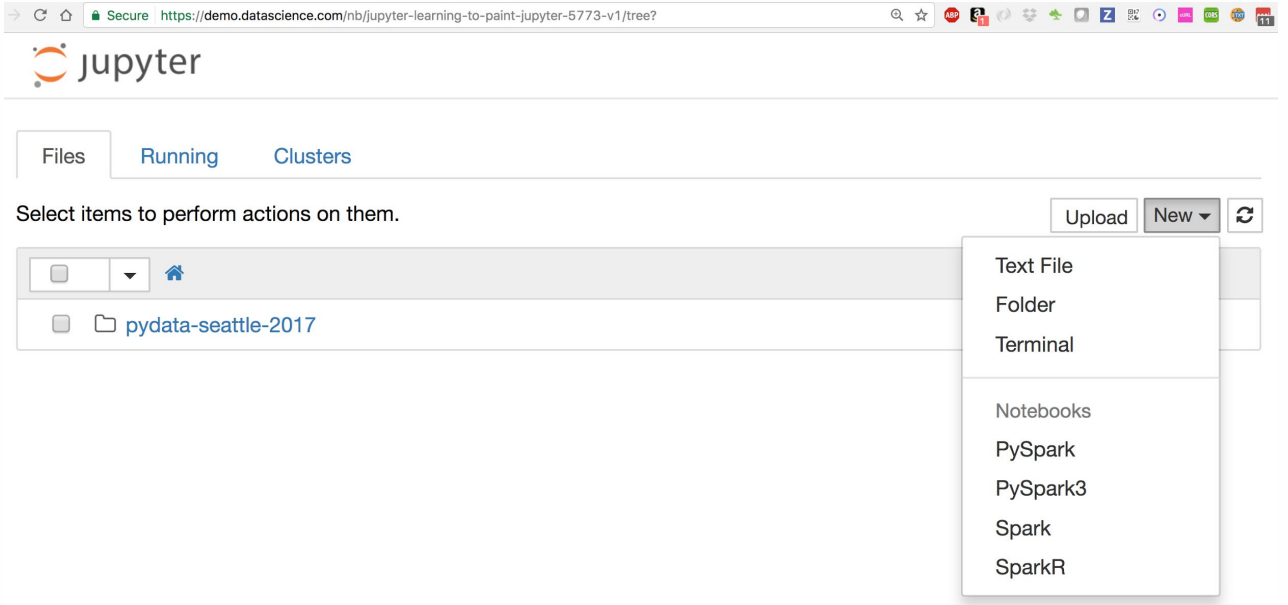


CPU Vs GPU


Computation Time in hrs vs. Experiment Type
Computation Time in hrs
10
8
6
4
2
0
?
CPU - Single
Spark-CPU - 12
GPU - Single
Spark - GPU
Experiment Type
Computation Time in hrs

HOW DO WE DO IT - *datascience.com* platform





OTHER PROJECTS

DATASCIENCE.COM

SKATER

- Open source python library for Model Interpretation
- Algorithms:
  - **Global Interpretation**
    - Model Agnostic Partial Dependence Plot
    - Model Agnostic Variable Importance
  - **Local Interpretation**
    - Local Interpretable Model Explanation (LIME)
- Github: https://github.com/datascienceinc/Skater

Q&A

dstm@datascience.com

pramit@datascience.com

REFERENCE

- https://arxiv.org/pdf/1603.01768.pdf
- http://www.asimovinstitute.org/neural-network-zoo/
- Russakovsky, O., Deng, J., Su, H., et al. "ImageNet Large Scale Visual Recognition Challenge." *International Journal of Computer Vision (IJCV)*. Vol 115, Issue 3, 2015, pp. 211–252
- Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014)
- TensorFlow: A system for large-scale machine learning: https://arxiv.org/abs/1605.08695
- Neural Style TensorFlow implementation: https://github.com/anishathalye/neural-style
- Neural Style Caffe implementation: https://github.com/fzliu/style-transfer
- Neural Style Keras implementation: https://github.com/fchollet/keras/blob/master/examples/neural_style_transfer.py
- Other Frameworks
  - Keras : https://github.com/fchollet/keras
  - TFLearn: https://github.com/tflearn/tflearn

UPCOMING TALKS AND EVENTS

DATASCIENCE.COM
ELEVATE

Andrea Trevino: will be a panelist discussing data science best practices with industry leaders from Google, Netflix, eHarmony, Live Nation and others

Aaron Kramer: "Interactive natural language processing with SpaCy and Jupyter"