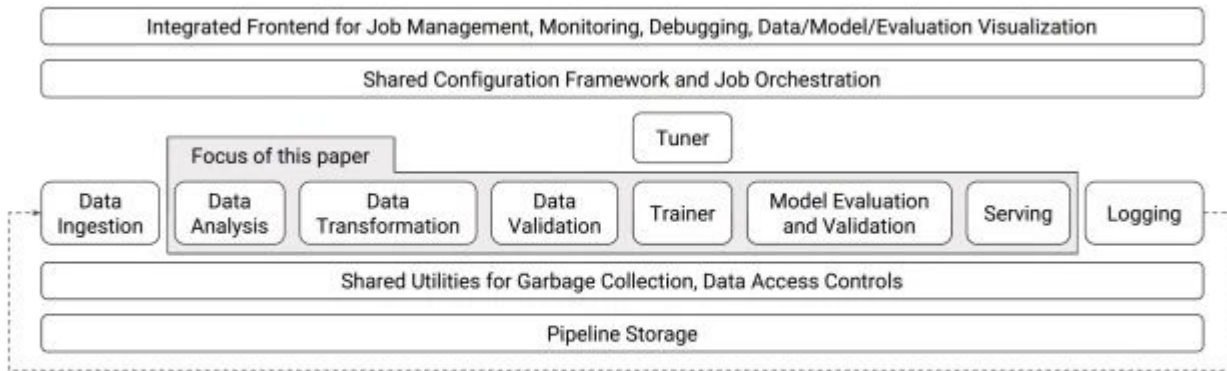


@joerg_schad

Kubeflow++

Building an Open Source Data Science Platform







Monitoring & Operations



DATADOG

TensorBoard

Data & Streaming

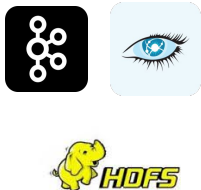
Model Engineering

Model Training

Model Management

Model Serving

Distributed Data Storage and Streaming



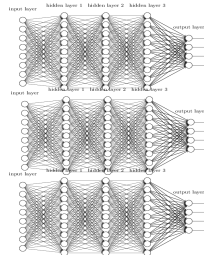
Data Preparation and Analysis



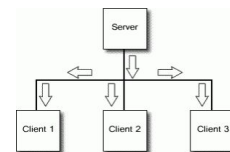
Distributed Training using Machine Learning Frameworks



Storage of trained Models and Metadata



Use trained Model for Inference



Feature Catalogue



Jenkins

Continuous Integration

TensorFlow Hub

Model Library

DC/OS

Resource and Service Management

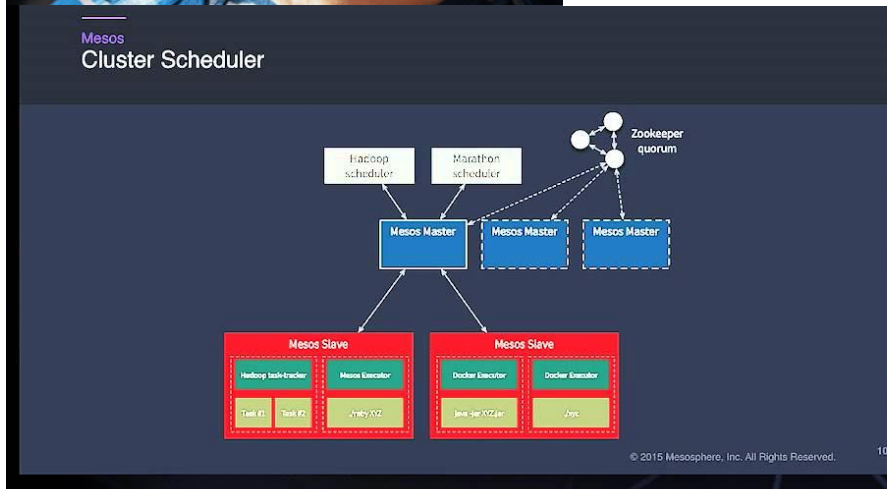
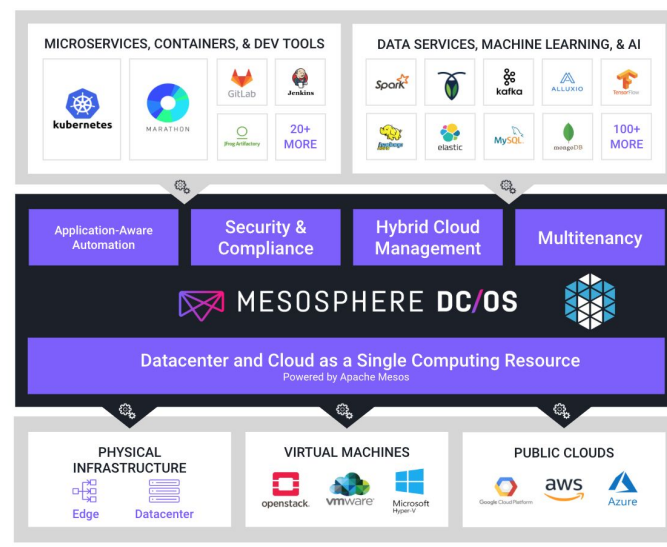
kubernetes

MESOS

Jörg Schad

Technical Lead/Engineer
Deep Learning

- Core Mesos developer at Mesosphere
- Twitter: @joerg_schad



Why is machine learning taking off?

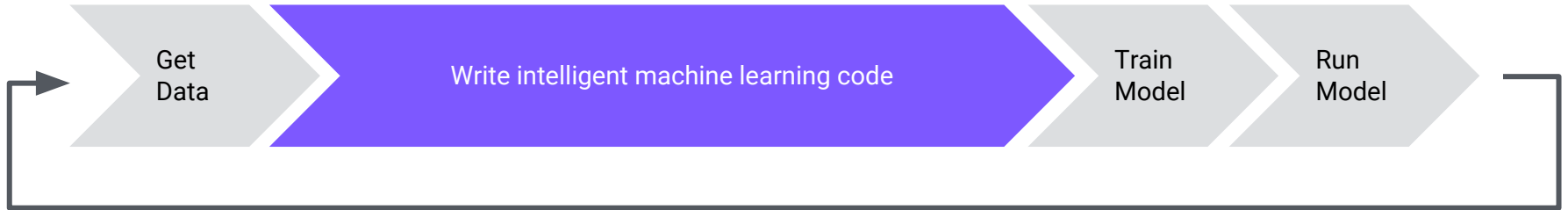






DEEPBACH: A STEERABLE MODEL FOR BACH CHORALES
GENERATION

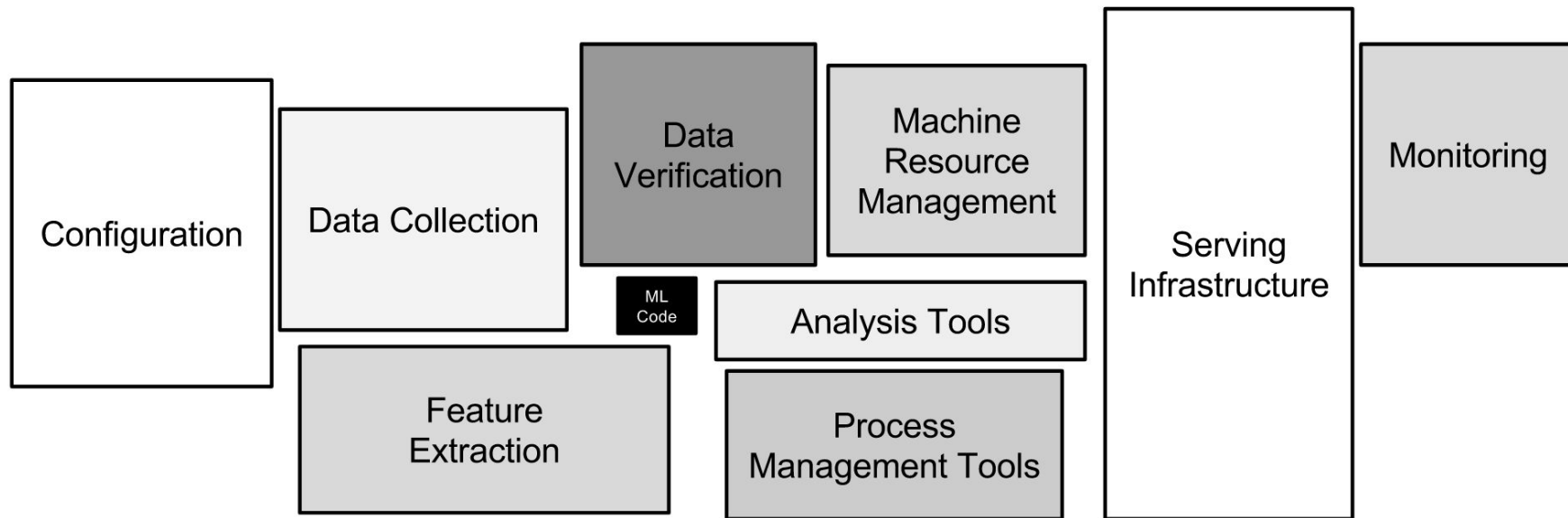
What you want to be doing



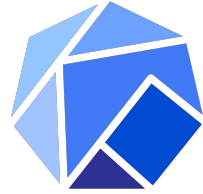
Repeat



What you're actually doing



Sculley, D., Holt, G., Golovin, D. et al. Hidden Technical Debt in Machine Learning Systems



Kubeflow



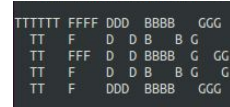
1. Data Preparation
& Model Engineering



2. Model Training

TensorBoard

3. Monitoring



4. Debugging



5. Model Serving

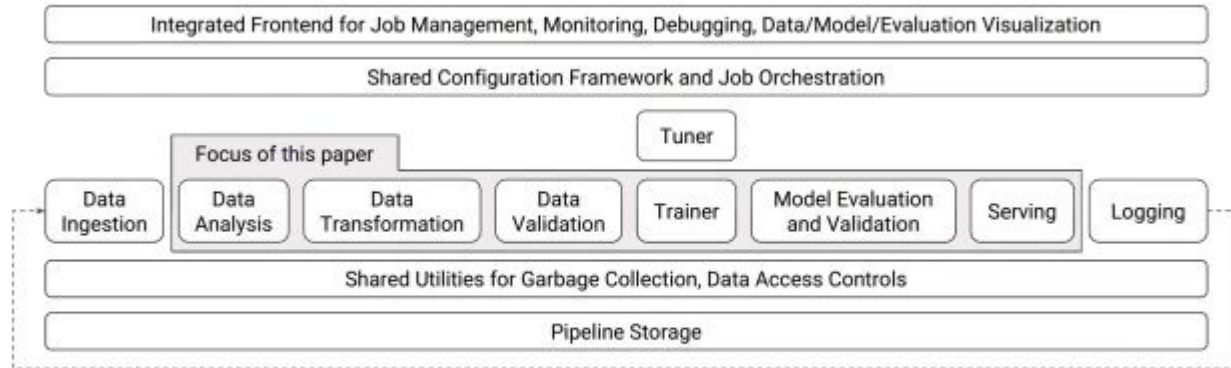
Kubeflow



The Kubeflow project is dedicated to making deployments of machine learning (ML) workflows on Kubernetes simple, portable and scalable.

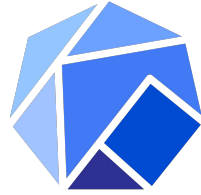
<https://www.kubeflow.org/docs/about/kubeflow/>

TFX: A TensorFlow-Based Production-Scale Machine Learning Platform



<http://www.kdd.org/kdd2017/papers/view/tfx-a-tensorflow-based-production-scale-machine-learning-platform>

<https://www.youtube.com/watch?v=fPTwLVCq00U>



Kubeflow



Katib

Hyperparameter
Optimization



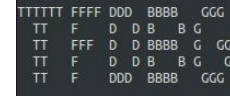
1. Data Preparation
& Model Engineering



2. Model Training

TensorBoard

3. Monitoring



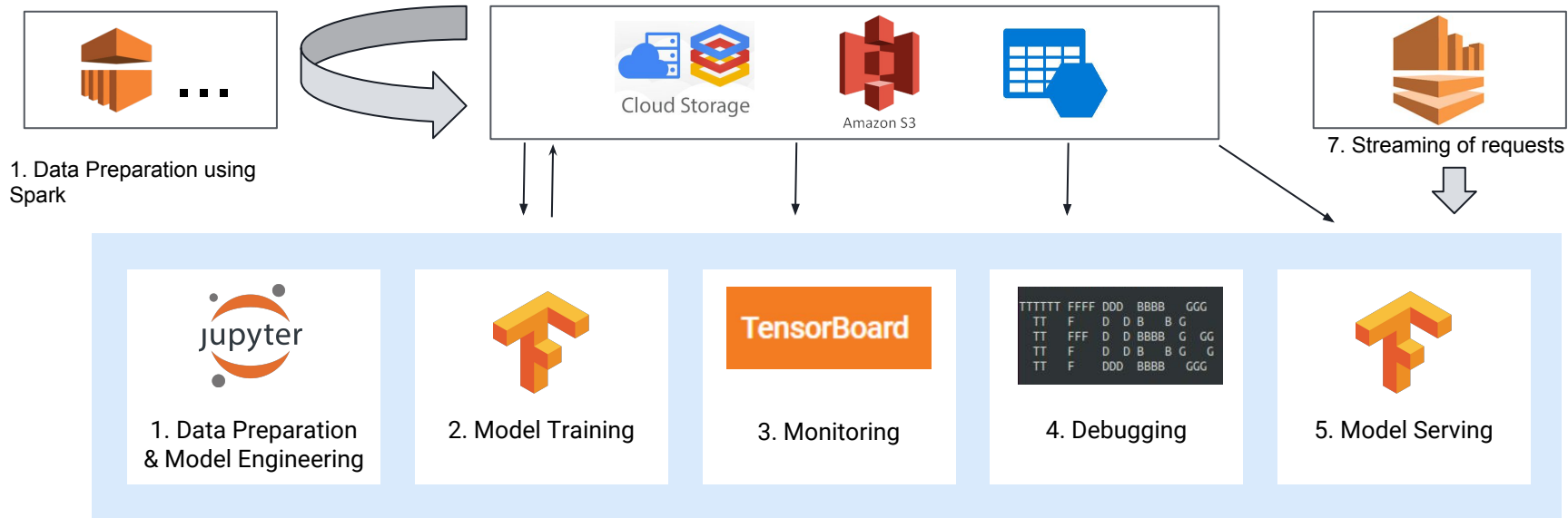
4. Debugging



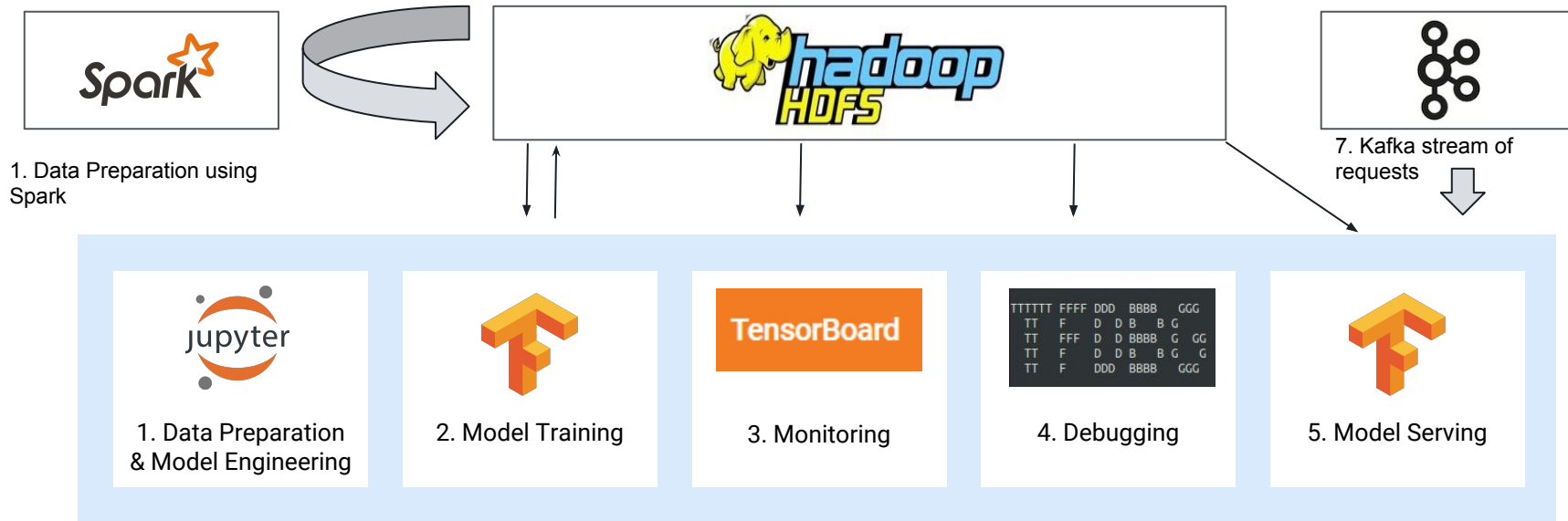
SELDON

5. Model Serving

Public Cloud Pipeline



DIY Open Source Pipeline



Monitoring & Operations



DATADOG

TensorBoard

Data & Streaming

Model Engineering

Model Training

Model Management

Model Serving

Distributed Data Storage and Streaming



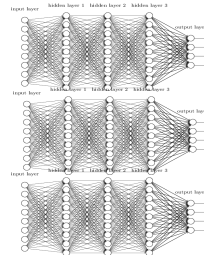
Data Preparation and Analysis



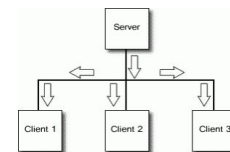
Distributed Training using Machine Learning Frameworks



Storage of trained Models and Metadata



Use trained Model for Inference



Feature Catalogue



Jenkins

Continuous Integration

TensorFlow Hub

Model Library

DC/OS

Resource and Service Management

kubernetes

MESOS

Challenge: Persona(s)

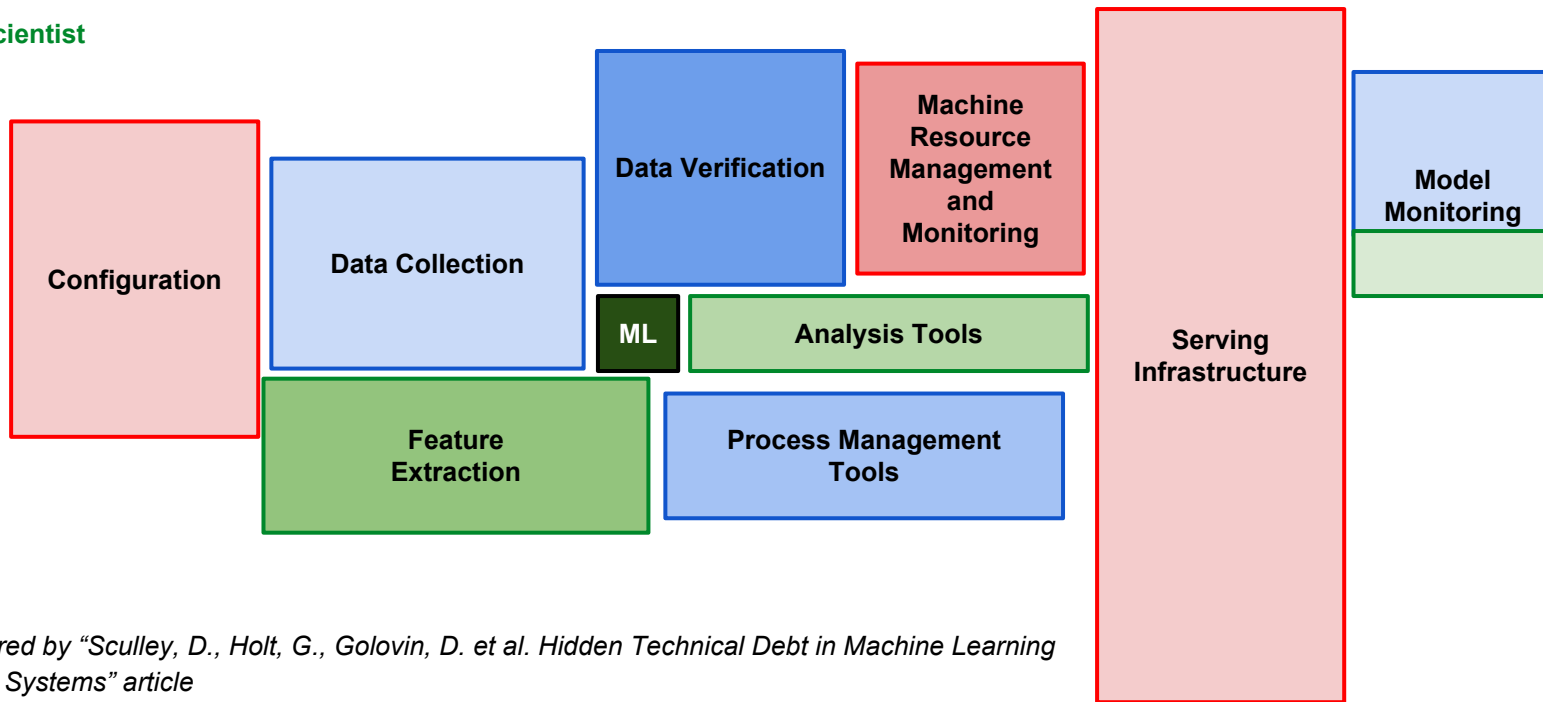


Division of Labor

System Admin/ DevOps

Data Engineer/DataOps

Data Scientist



Inspired by "Sculley, D., Holt, G., Golovin, D. et al. Hidden Technical Debt in Machine Learning Systems" article

The Rise of the *DataOps Engineer*

Combines two key skills:

- Data science
- Distributed systems engineering

The equivalent of *DevOps* for *Data Science*



SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

Software Engineering

**The application of a systematic, disciplined,
quantifiable approach to the development,
operation, and maintenance of software**

IEEE Standard Glossary of Software Engineering
Terminology

**Do we need Data Science Engineering
Principles?**

Do

A screenshot of a tweet from Ian Goodfellow (@goodfellow_ian) replying to @rctatman. The tweet text reads: "I have a different controversial opinion: ML development is a different kind of software development and has a different set of best practices." The tweet is dated 7:59 PM - 18 Sep 2018. The interface includes a profile picture, name, handle, a "Follow" button, and a dropdown arrow.

 **Ian Goodfellow**
@goodfellow_ian

Following

Replying to @rctatman

I have a different controversial opinion: ML development is a different kind of software development and has a different set of best practices.

7:59 PM - 18 Sep 2018

ering

Challenge: Requirements Engineering

- Do I need Machine Learning? *
- Do I need {Neural Networks, Regression,...}*

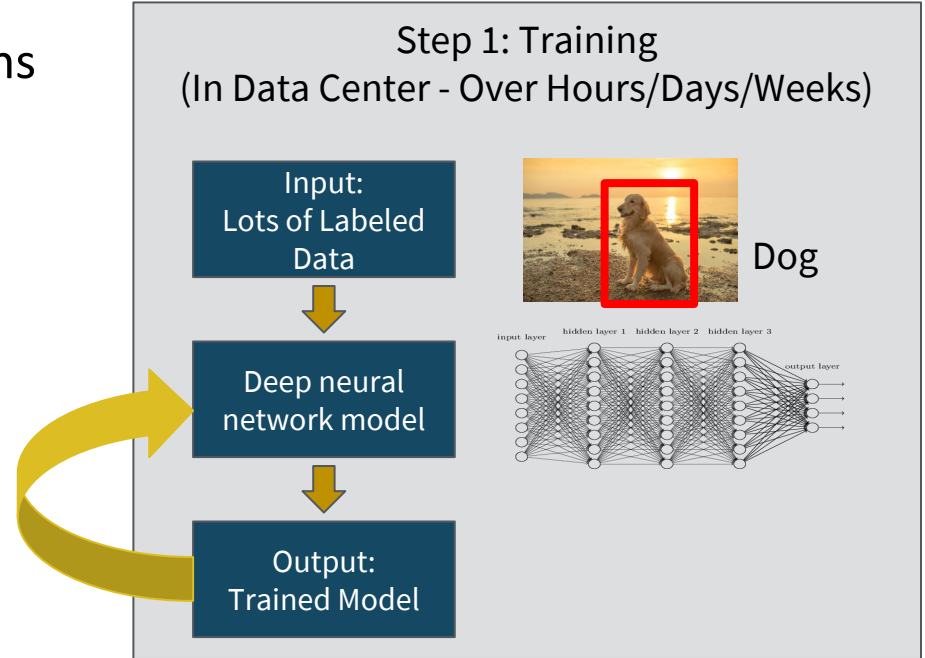
- What dataset(s)?
 - Quality?
- What target/serving environment?
- What model architecture?
- Pre-trained model available?
- How many training resources?

* Can I actually use ...

Challenge: Reproducible Builds

- Many adhoc model/training runs
- Regulatory Requirements
- Dependencies
- CI/CD
- Git

mlflow



mlflow



Tracking

Record and query experiments: code, data, config, results

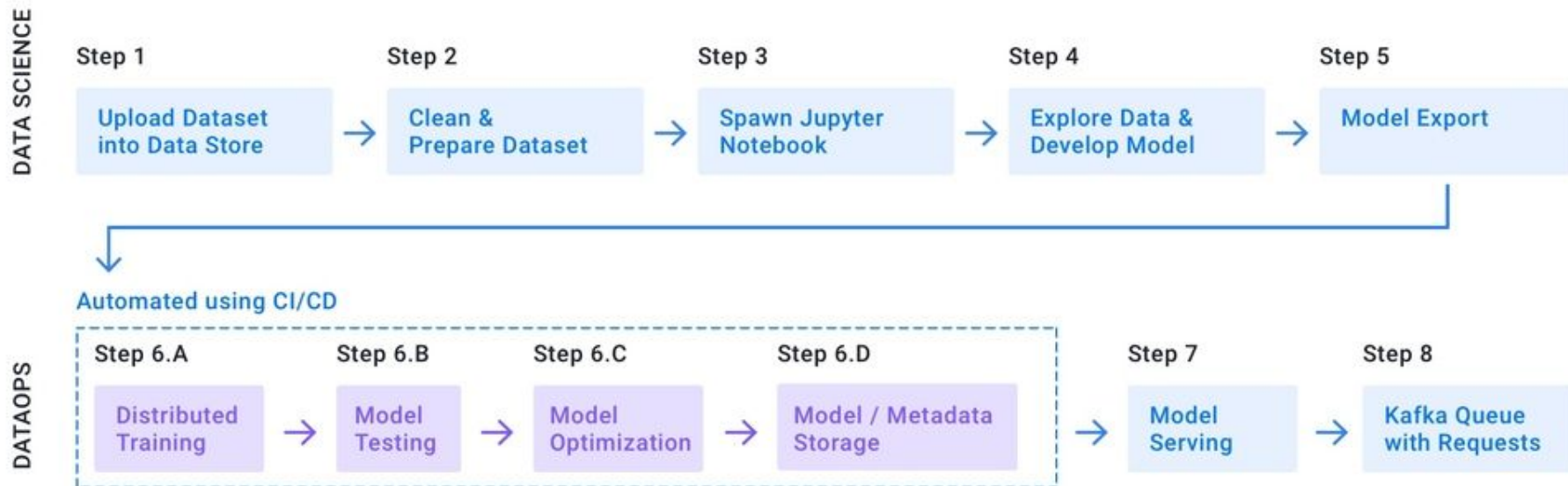
Projects

Packaging format for reproducible runs on any platform

Models

General format for sending models to diverse deploy tools

Challenge: Automation & CI/CD



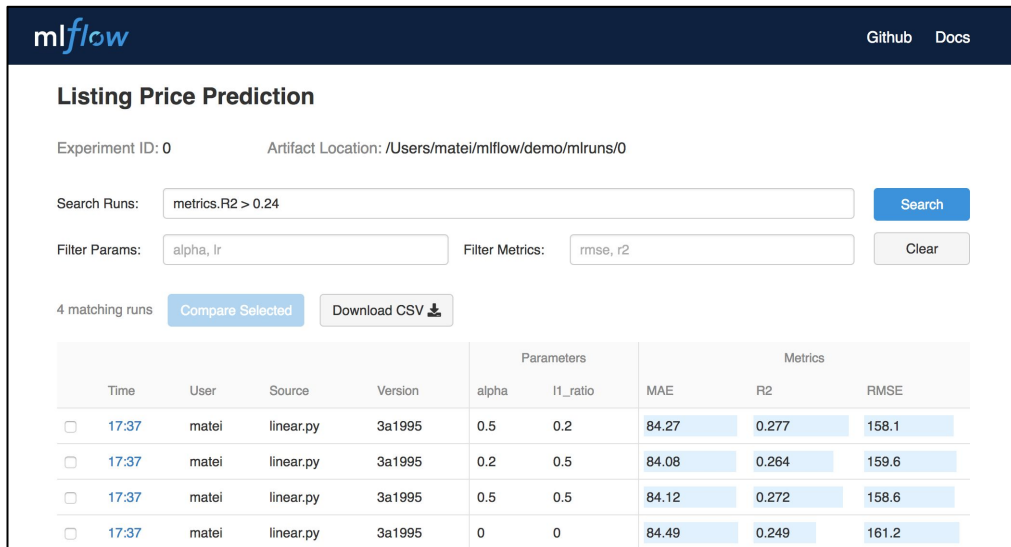
MFlow Tracking

```
import mlflow

# Log parameters (key-value pairs)
mlflow.log_param("num_dimensions", 8)
mlflow.log_param("regularization", 0.1)

# Log a metric;
mlflow.log_metric("accuracy", 0.1)
...
mlflow.log_metric("accuracy", 0.45)

# Log artifacts (output files)
mlflow.log_artifact("roc.png")
mlflow.log_artifact("model.pkl")
```



The screenshot displays the MFlow web interface for an experiment titled "Listing Price Prediction". The interface includes a search bar with the query "metrics.R2 > 0.24" and a "Search" button. Below the search bar, there are filter options for "Filter Params" (set to "alpha, l1") and "Filter Metrics" (set to "rmse, r2"), along with a "Clear" button. The results section shows "4 matching runs" with buttons for "Compare Selected" and "Download CSV". A table lists the runs with columns for Time, User, Source, Version, Parameters (alpha, l1_ratio), and Metrics (MAE, R2, RMSE).

	Time	User	Source	Version	Parameters		Metrics		
					alpha	l1_ratio	MAE	R2	RMSE
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.2	84.27	0.277	158.1
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.2	0.5	84.08	0.264	159.6
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.5	84.12	0.272	158.6
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0	0	84.49	0.249	161.2

MFlow Project

```
name: My Project
conda_env: conda.yaml
entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default:
0.1}
    command: "python train.py -r
{regularization} {data_file}"
  validate:
    parameters:
      data_file: path
    command: "python validate.py {data_file}"
```

```
$mflow run example/project -P alpha=0.5
$mflow run git@github.com:databricks/mlflow-example.git
```

MFlow Model

```
time_created: 2018-02-21T13:21:34.12
```

```
flavors:
```

```
  sklearn:
```

```
    sklearn_version: 0.19.1
```

```
    pickled_model: model.pkl
```

```
python_function:
```

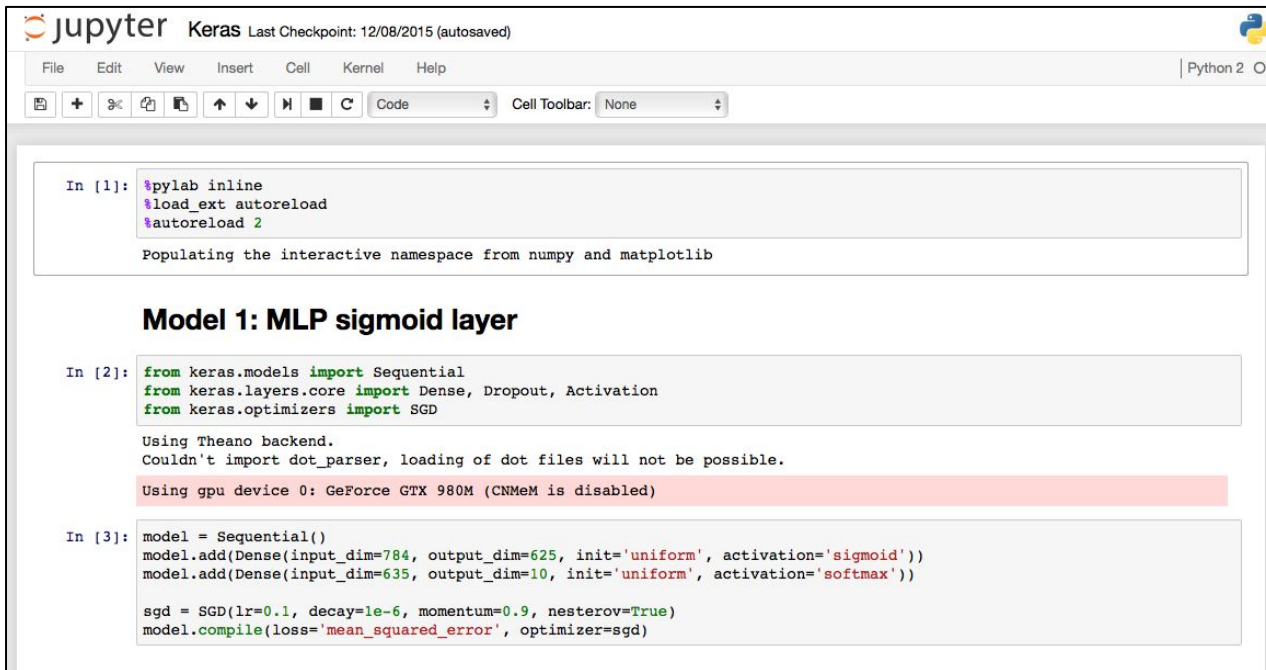
```
  loader_module: mlflow.sklearn
```

```
  pickled_model: model.pkl
```

```
$mlflow run example/project -P alpha=0.5
```

```
$mlflow run git@github.com:databricks/mlflow-example.git
```

Challenge: Data Science IDE



The screenshot displays the JupyterLab interface with the following components:

- Header:** "jupyter Keras Last Checkpoint: 12/08/2015 (autosaved)" and a Python 2 logo.
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Help.
- Toolbar:** Includes icons for file operations, a "Code" dropdown menu, and a "Cell Toolbar: None" dropdown.
- Code Cell 1:**

```
In [1]: %pylab inline
        %load_ext autoreload
        %autoreload 2
```

Populating the interactive namespace from numpy and matplotlib
- Section Header:** **Model 1: MLP sigmoid layer**
- Code Cell 2:**

```
In [2]: from keras.models import Sequential
        from keras.layers.core import Dense, Dropout, Activation
        from keras.optimizers import SGD
```

Using Theano backend.
Couldn't import dot_parser, loading of dot files will not be possible.
Using gpu device 0: GeForce GTX 980M (CNMem is disabled)
- Code Cell 3:**

```
In [3]: model = Sequential()
        model.add(Dense(input_dim=784, output_dim=625, init='uniform', activation='sigmoid'))
        model.add(Dense(input_dim=635, output_dim=10, init='uniform', activation='softmax'))

        sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
        model.compile(loss='mean_squared_error', optimizer=sgd)
```


Challenge: Data Quality

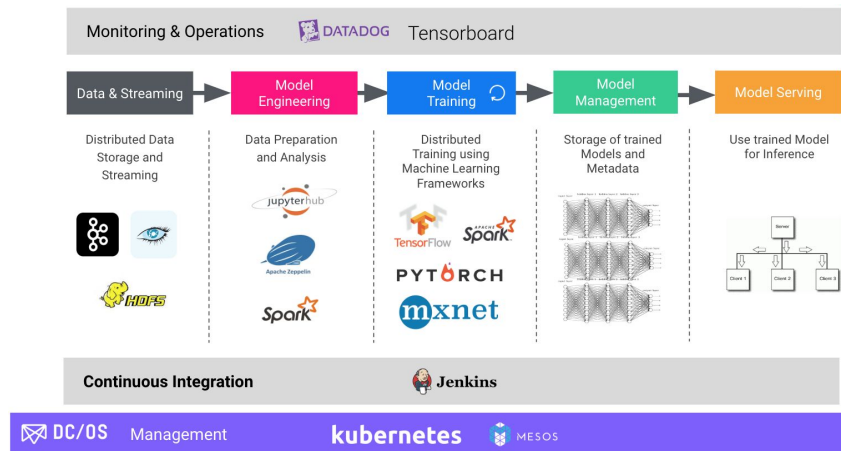
- Data is typically not ready to be consumed by ML job*
 - Data Cleaning
 - Missing/incorrect labels
 - Data Preparation
 - Same Format
 - Same Distribution

* Demo datasets are a fortunate exception :)

Challenge: Data Quality

- Data is typically not ready to be consumed by ML job*
 - Data Cleaning
 - Missing/incorrect labels
 - Data Preparation
 - Same Format
 - Same Distribution

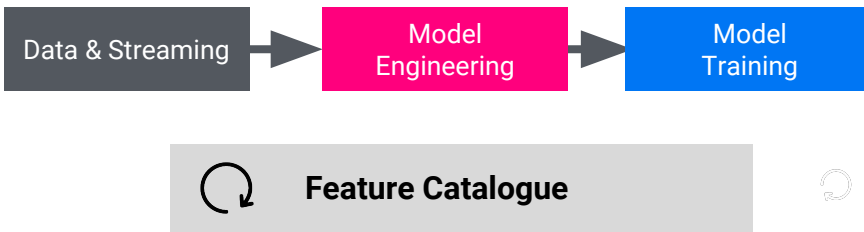
Don't forget about the serving environment!!



* Demo datasets are a fortunate exception :)

Challenge: Data (Preprocessing) Sharing

- Preprocessed Data Sets valuable
 - Sharing
 - Automatic Updating
- Feature Catalogue \approx Preprocessing Cache + Discovery



Challenge: Model Libraries

- Existing architectures
- Pretrained models

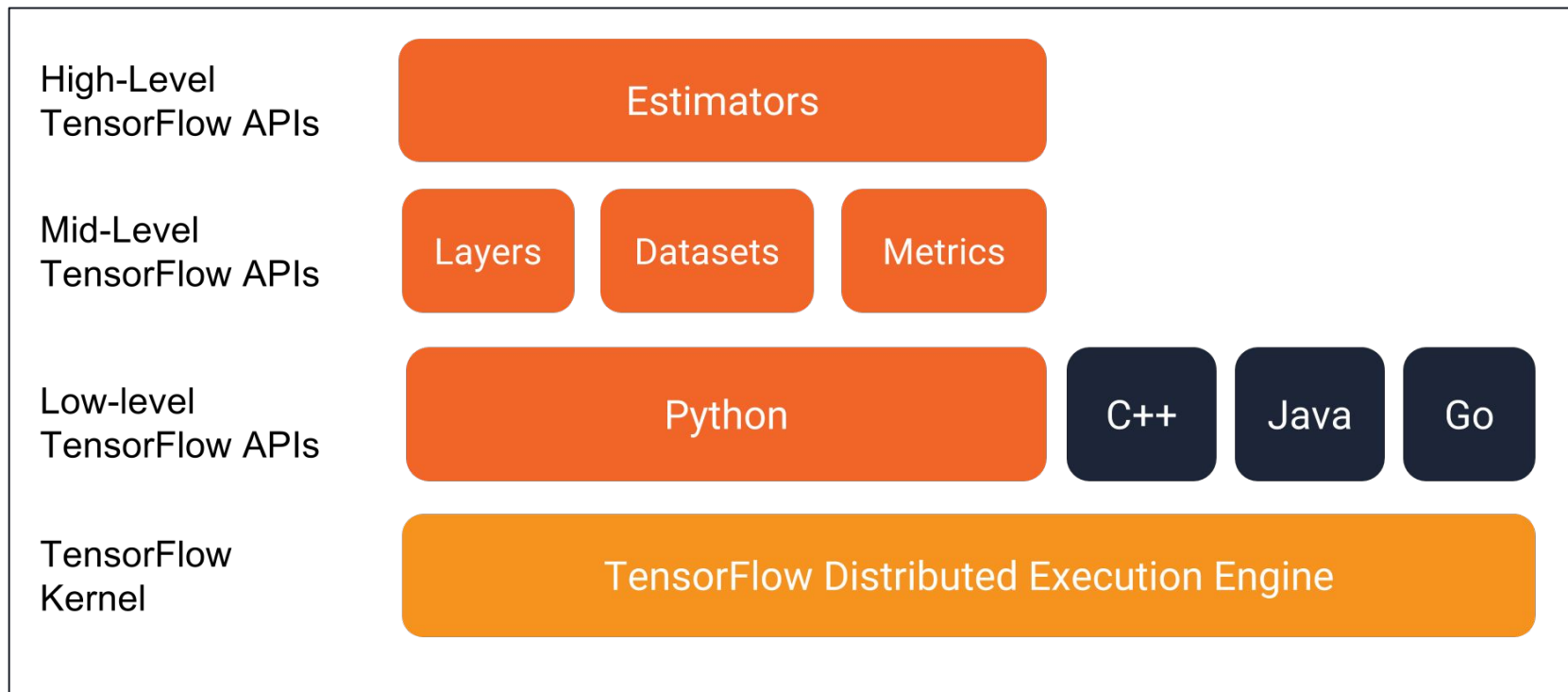
```
import tensorflow as tf
import tensorflow_hub as hub

with tf.Graph().as_default():
    embed = hub.Module("https://tfhub.dev/google/nlm-en-dim128-with-normalization/1")
    embeddings = embed(["A long sentence.", "single-word", "http://example.com"])

    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        sess.run(tf.tables_initializer())

    print(sess.run(embeddings))
```

Challenge: Writing Distributed Model Functions



Challenge: Debugging

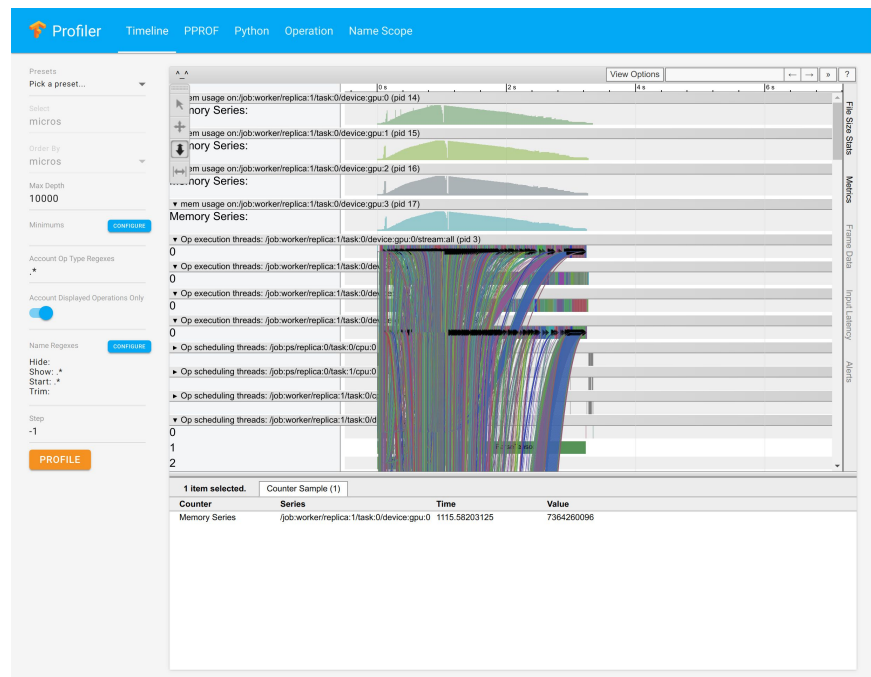
The screenshot displays the TensorBoard Debugger interface, which is currently inactive. The interface is divided into several sections:

- Node List:** Shows a list of nodes for the current session. The selected node is `/job:localhost/replica:0/task:0/device:CPU:0`. It includes checkboxes for `Add`, `Const`, `Mul`, and `Variable`.
- Source Code:** Displays the source code for the file `tdp_demo.py`. The code includes imports for TensorFlow and TensorFlow Debug, and defines variables `a`, `b`, `c`, `x`, and `y`. It also shows the session configuration and the execution of `tf.global_variables_initializer()`.
- Runtime Graphs:** Shows a graph of the runtime operations. The selected device is `/job:localhost/replica:0/task:0/device:CPU:0`. The graph includes nodes for `Variable_1`, `Add`, and `Variable`.
- Tensor Value Overview:** A table showing the values of tensors in the current session. The table has columns for Tensor, Count, DType, Shape, and Value. The tensors listed are `Variable:0`, `Variable/read:0`, `Const:0`, and `Variable_1:0`.
- Session Runs:** A table showing the session runs. The table has columns for Feeds, Fetches, Targets, #(Devices), and Count. The session runs are `init` and `Add:0`.
- Health Pill:** A visual indicator of the session's health, showing a blue pill and a legend for `NaN`, `0`, and `+∞`.

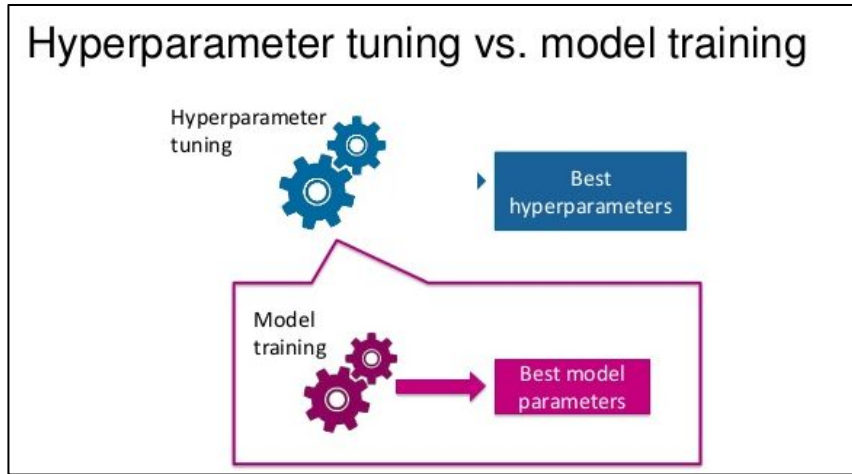
https://www.tensorflow.org/programmers_guide/debugger

Profiling

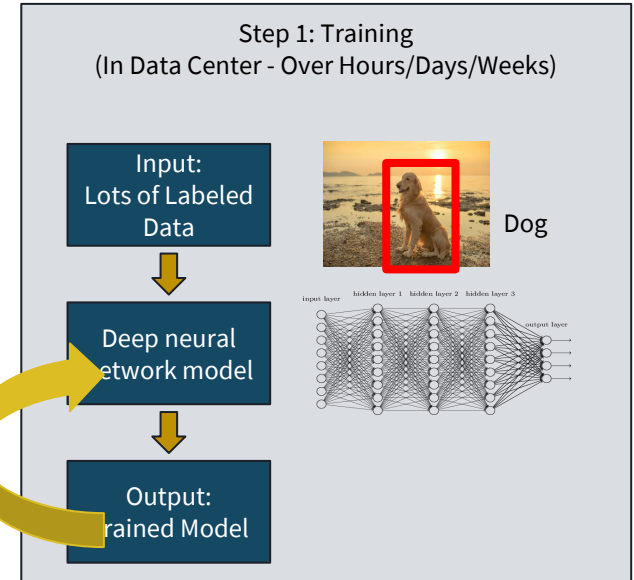
- Crucial when using “expensive” devices
- Memory Access Pattern
- “Secret knowledge”
- More is not necessarily better....



Hyperparameter Optimization



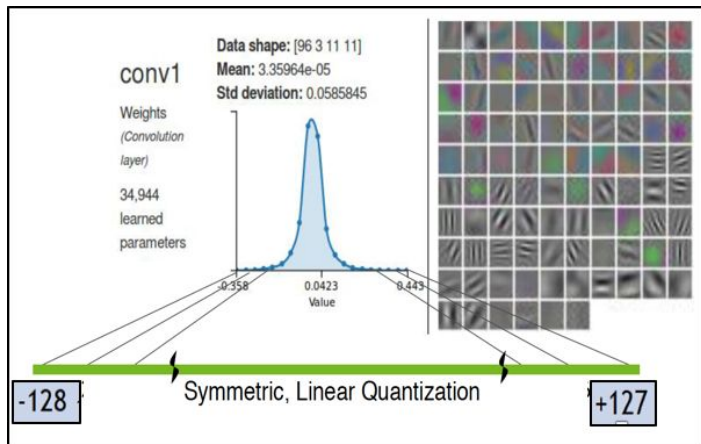
- Networks Shape
- Learning Rate
- ...



Model Optimization

```
transform_graph \  
  --in_graph=unoptimized_cpu_graph.pb \  
  --out_graph=optimized_cpu_graph.pb \  
  --inputs='x_observed:0' \  
  --outputs='Add:0' \  
  --transforms='                               \  
    strip_unused_nodes  
    remove_nodes(op=Identity, op=CheckNumerics)  
    fold_constants(ignore_errors=true)  
    fold_batch_norms  
    fold_old_batch_norms  
    quantize_weights  
    quantize_nodes '  
  < Original Graph  
  < Transformed Graph  
  < Feed (Input)  
  < Fetch (Output)  
  < List of Transforms
```

Model Optimization

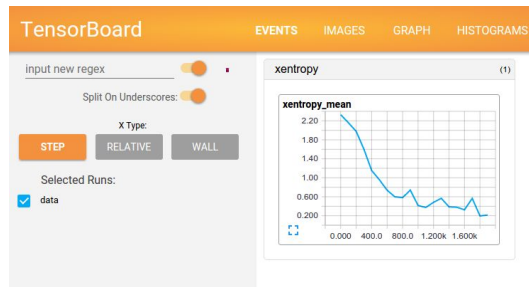


	Dynamic Range	Min Pos Value
FP32	$-3.4 \times 10^{38} \sim +3.4 \times 10^{38}$	1.4×10^{-45}
FP16	$-65504 \sim +65504$	5.96×10^{-8}
INT8	$-128 \sim +127$	1

Challenge: Monitoring

- Understand {...}
- Debug
- Model Quality
 - Accuracy
 - Training Time
 - ...
- Overall Architecture
 - Availability
 - Latencies
 - ...

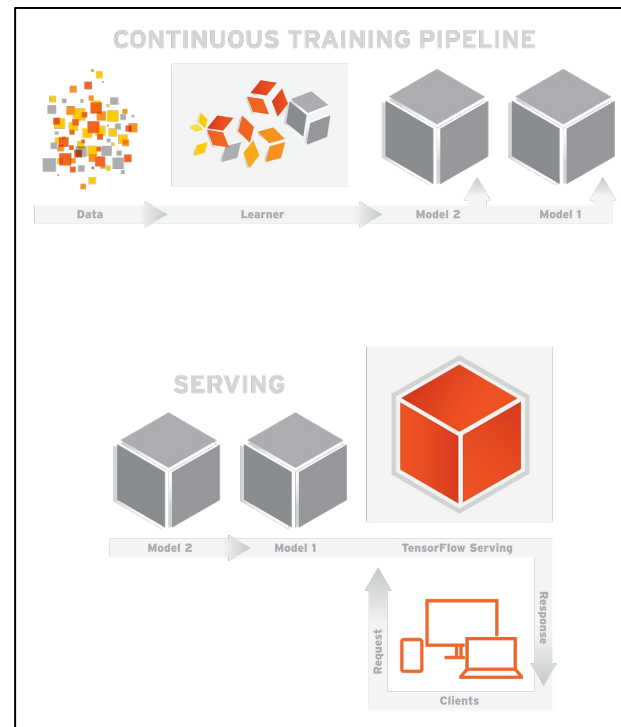
- TensorBoard



- Traditional Cluster Monitoring Tool

Challenge: Serving Environment

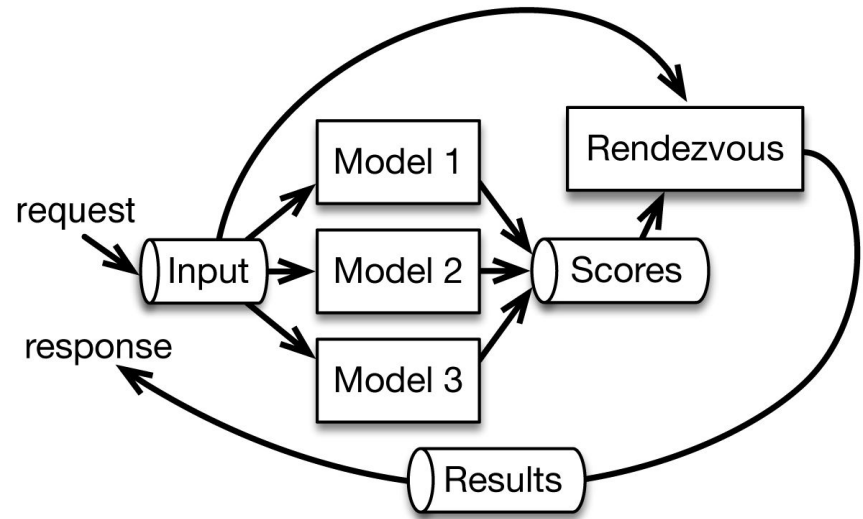
- How to Deploy Models?
 - Zero Downtime
 - Canary
- Multiple Models?
 - Testing



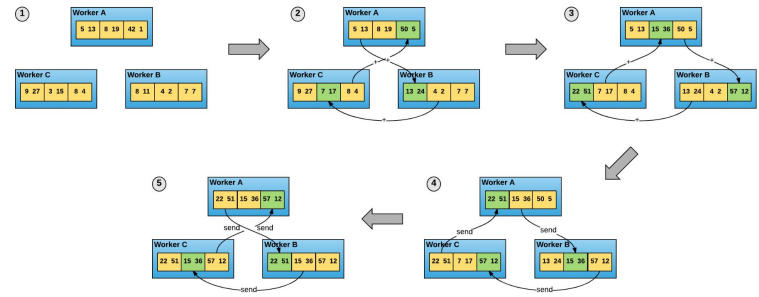
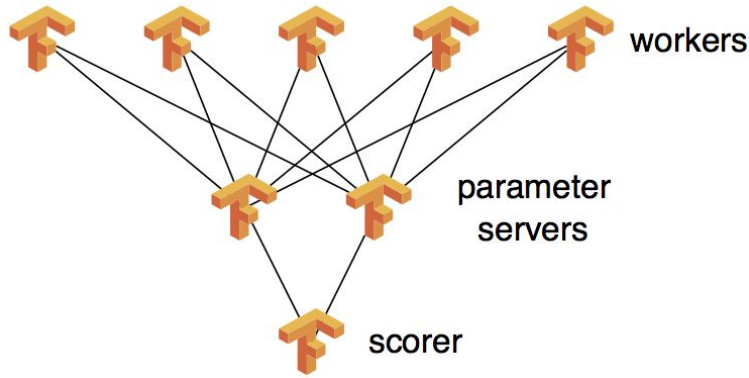
<https://ai.googleblog.com/2016/02/running-your-models-in-production-with.html>

Challenge: Serving Environment

- How to Deploy Models?
 - Zero Downtime
 - Canary
- Multiple Models?
 - Testing



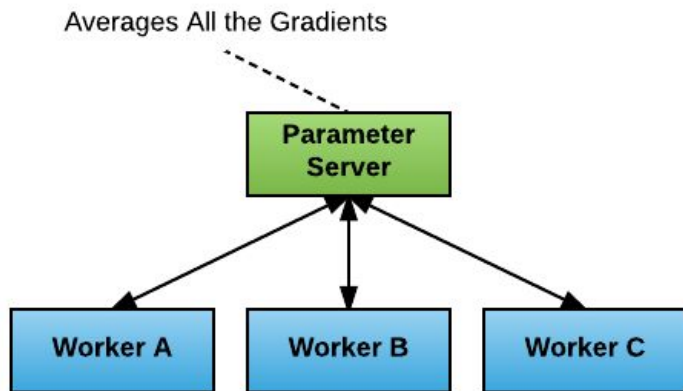
Challenge: Distributed TensorFlow



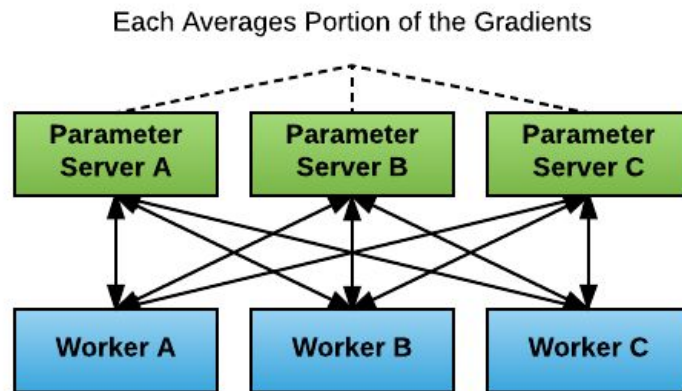
<https://eng.uber.com/horovod/>

<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/distribute>

Challenge: Distributed TensorFlow

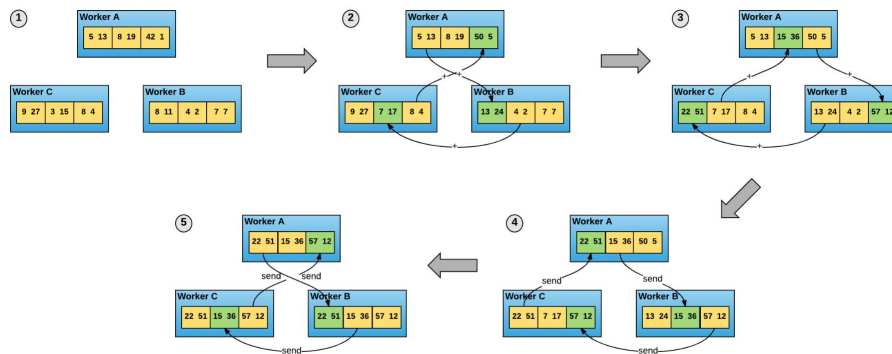


or



Horovod

- All-Reduce to update Parameter
 - Bandwidth Optimal
- Uber Horovod is MPI based
 - Difficult to set up
 - Other Spark based implementations
- Wait for TensorFlow 2.0 ;)

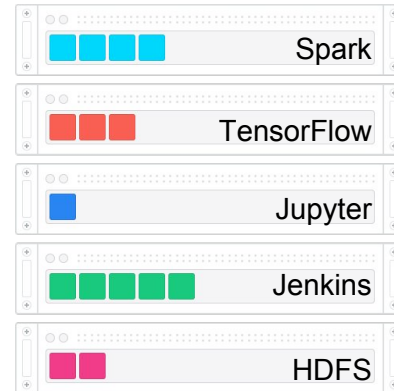


TF Distribution Strategy

- [MirroredStrategy](#): This does in-graph replication with synchronous training on many GPUs on one machine. Essentially, we create copies of all variables in the model's layers on each device. We then use all-reduce to combine gradients across the devices before applying them to the variables to keep them in sync.
- [CollectiveAllReduceStrategy](#): This is a version of `MirroredStrategy` for multi-working training. It uses a collective op to do all-reduce. This supports between-graph communication and synchronization, and delegates the specifics of the all-reduce implementation to the runtime (as opposed to encoding it in the graph). This allows it to perform optimizations like batching and switch between plugins that support different hardware or algorithms. In the future, this strategy will implement fault-tolerance to allow training to continue when there is worker failure.
- [ParameterServerStrategy](#): This strategy supports using parameter servers either for multi-GPU local training or asynchronous multi-machine training. When used to train locally, variables are not mirrored, instead they placed on the CPU and operations are replicated across all local GPUs. In a multi-machine setting, some are designated as workers and some as parameter servers. Each variable is placed on one parameter server. Computation operations are replicated across all GPUs of the workers.

Challenge: Resource and Service Management

- Different Distributed Systems
 - Deployment
 - Updates
 - Failure Recovery
 - Scaling
- Resource Efficiency
 - Multiple VM per Service?



Typical Datacenter
siloed, over-provisioned servers,
low utilization

Monitoring & Operations



DATADOG

TensorBoard

Data & Streaming

Model Engineering

Model Training

Model Management

Model Serving

Distributed Data Storage and Streaming

Data Preparation and Analysis

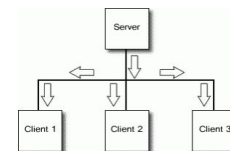
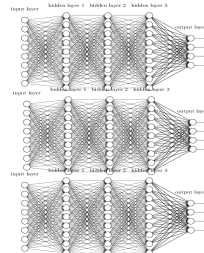
Distributed Training using Machine Learning Frameworks

Storage of trained Models and Metadata

Use trained Model for Inference



PYTORCH



Feature Catalogue



Jenkins

Continuous Integration

TensorFlow Hub

Model Library

DC/OS

Resource and Service Management

kubernetes

MESOS



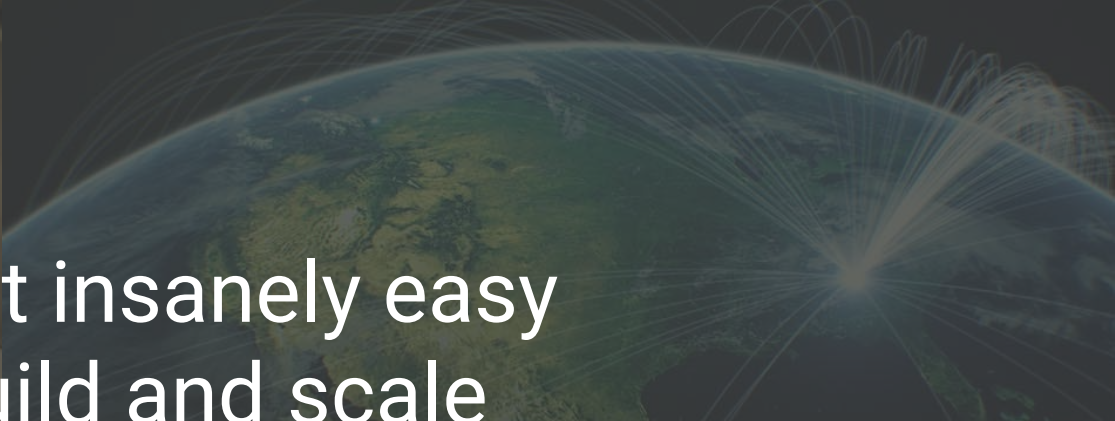

@mesosphere

THANK YOU!

ANY
QUESTIONS?



<https://mesosphere.com/resources/building-data-science-platform/>



Make it insanely easy
to build and scale
world-changing technology



MESOSPHERE

