ML Ops and Kubeflow Pipelines

Solutions and Best Practices for DevOps of Production ML Services

Kaz Sato, Developer Advocate, Google Cloud
Kaz Sato

Developer Advocate
Data & Analytics
Google Cloud

@kazunori_279
What is "ML Ops"?

DevOps for ML
Launching is easy, Operating is hard.

"The real problems with a ML system will be found while you are continuously operating it for the long term"
What is DevOps?

“DevOps is a software engineering culture and practice that aims at **unifying** software development (Dev) and software operation (Ops).”

“(DevOps is to) strongly advocate **automation** and **monitoring** at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management.”

- Wikipedia
What is ML Ops?

ML Ops is a software engineering culture and practice that aims at unifying ML system development (Dev) and ML system operation (Ops).

(ML Ops is to) strongly advocate automation and monitoring at all steps of ML system construction, from integration, testing, releasing to deployment and infrastructure management.
Machine Learning: The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young
{dsculley, gholt, dgg, edavydov}@google.com
{toddphillips, ebner, vchaudhary, mwyoung}@google.com
Google, Inc
Rules of Machine Learning:

Best Practices for ML Engineering

Martin Zinkevich

This document is intended to help those with a basic knowledge of machine learning get the benefit of Google's best practices in machine learning. It presents a style for machine learning, similar to the Google C++ Style Guide and other popular guides to practical programming. If you have taken a class in machine learning, or built or worked on a machine-learned model, then you have the necessary background to read this document.
Agenda

Development anti-patterns
Deployment anti-patterns
Operation anti-patterns
"Depending on a ML superhero"

A ML superhero is:
- ML Researcher
- Data engineer
- Infra and Ops engineer
- Product Manager
- A partner to execs
- From PoC to production
Solution: split the roles, build a scalable team

Split the roles to:

ML Researcher
Data engineer
Infra and Ops engineer
Product Manager
Business decision maker
Example: Candy Sorter demo at I/O and Next

The team:
Researcher: ML models
Software engineer: software integration
Hardware engineer: robot SDK control
Project manager: leading team
Business decision maker: me
"A black box that nobody understands"

Scenario:
Researcher creates a prototype
Engineer refactors it, brings it to production
Researcher doesn't understand the code
Engineer doesn't understand the model
“At Google, a hybrid research approach where engineers and researchers are embedded together on the same teams has helped reduce this source of friction significantly.”
Example: Engineer intros scalable platform

Scenario:
Researcher creates scikit-learn model
Engineer ports it to Cloud AI Platform
Split data into training and testing

```python
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Setup the pipeline which will be used for both training and prediction

```python
pipeline = Pipeline(steps=[
    ('preprocessor', DictVectorizer(sparse=False)),
    ('estimator', RandomForestRegressor(max_depth=5))])
```

Train

```python
pipeline.fit(x_train, y_train)
```

Make predictions (on the local machine)

```python
print_predictions(pipeline.predict(x_test))
```

Export the model
Changing Anything Changes Everything

**Entanglement of ML system**

A change to one feature could affect all of the other features

A change of a hyper-param could affect the whole result (regularization, learning rates, sampling, thresholds, etc.)

"Launching is easy, operating is hard"

"I'm just changing one feature"
Rules of ML paper says:

“Rule #14: Starting with an interpretable model makes debugging easier”

“Rule #40: Keep ensembles simple”
Solution: use simple model and feature

Use complex model judiciously:
Linear v. Deep
Convex v. Non-convex
Interpretable v. black box
Solution: use ensembled model

Rules for using ensembled model:

Use either one of:
- A model taking input features: parts factory
- A model assembles those models: assembly plant

Use semantically interpretable model
for better robustness and easier troubleshooting
Ensemble model in Aucnet's used car classifier
"Lack of data validation"

In IT system:
The behavior of the system is defined by code
Validating functionality of your system with unit tests

In ML system:
The behavior of the system is defined by data
Validating functionality of your system with what?

"Data is the code"
"Training-serving skew"

**Cause:**
Any differences (data, preprocessing, window etc) between training and serving

**Result:**
Accuracy drops when serving
Solution: TensorFlow Extended (TFX)

An end-to-end tool for deploying production ML system

tensorflow.org/tfx
TensorFlow Data Validation (TFDV)

Helps developers understand, validate, and monitor their ML data at scale.

Used analyze and validate petabytes of data at Google every day.

Has a proven track record in maintaining the health of production ML pipelines.
“We want the user to treat data errors with the same rigor and care that they deal with bugs in code.”

Google Play app install rate improved 2% after introducing data validation, finding stale table
TensorFlow Data Validation Demo

An Example of a Key Component of TensorFlow Extended

Note: You can run this example right now in a Jupyter-style notebook, no setup required! Just click "Run in Google Colab"

View on TensorFlow.org Run in Google Colab View source on GitHub

This example colab notebook illustrates how TensorFlow Data Validation (TFDV) can be used to investigate and visualize your dataset. That includes looking at descriptive statistics, inferring a schema, checking for and fixing anomalies, and checking for drift and skew in our dataset. It’s important to understand your dataset’s characteristics, including how it might change over time in your production pipeline. It’s also important to look for anomalies in your data, and to compare your training, evaluation, and serving datasets to make sure that they’re consistent.

We’ll use data from the Taxi Trips dataset released by the City of Chicago.

Note: This site provides applications using data that has been modified for use from its original source, www.cityofchicago.org, the official website of the City of Chicago. The City of Chicago makes no claims as to the content, accuracy, timeliness, or completeness of any of the data provided at this site. The data provided at this site is subject to change at any time. It is understood that the data provided at this site is being used at one’s own risk.
Infer a schema

A schema defines constraints for the data such as:
"Lack of continuous monitoring"

**Scenario:**

Model accuracy drops over time

No practice for continuous monitoring

End users are frustrated with the experience

Business team notices it

Director asks the researcher to update the model ASAP

Don't you know what's happening now?!
“Not knowing the freshness requirements”

Different freshness for different applications:

- News aggregation: 5 min
- E-commerce item recommend: 1 day/week
- NLP for CSAT measurement: 1 month
- Voice recognition: years?
- Object detection for event: every setup
“Rule #8: Know the freshness requirements of your system”
TensorFlow Model Analysis (TFMA)

Compute and visualize **evaluation metrics** for ML models

Ensure to meet specific **quality thresholds** and **behaves as expected** for all relevant slices of data

Provide tools to create a **deep understanding** of model performance
Measure the delta between models

The delta defines the refresh requirement
Use "sliced" metrics for better model analysis.

Aggregate metric computed over the entire eval dataset

Metric "sliced" by different segments of the eval dataset

ROC Curve

Sensitivity (True Positive Rate) vs Specificity (False Positive Rate)

- All data
- Segment A
- Segment B
Visualizing TFMA Results

This notebook describes how to visualize the results generated by TFMA either locally or using DataFlow.

### Output Directory

Please set `PATH_TO_RESULT` to load the result:

- For local evaluation using `process_tfma_local.sh`, the evaluation result is defaulted to the folder under `os.path.join(os.
  'train', 'local_chicago_taxi_output', 'eval_result')`

Comment out / skip the cell below if you are trying to visualize results from DataFlow.

```python
[ ] # Assume the eval result is written to the first subfolder under the target directory.
  PATH_TO_RESULT = os.path.join(os.getcwd(), 'data', 'train', 'local_chicago_taxi_output', 'eval_result');
```

- For visualization of results from DataFlow using `process_tfma_dataflow.sh`, the evaluation result is outputted to a subfolder under `TFT_OUTPUT_PATH`.

Uncomment the cell below if you are trying to visualize results from DataFlow.

```python
[ ] # PATH_TO_RESULT = os.environ['TFT_OUTPUT_PATH'] + '/eval_result_dir'
```

### Loading the Result

Once the path is set, load the results into a `tfma.EvalResult` using `tfma.load_eval_result`.  

---

TensorFlow Model Analysis Demo
ML Fairness: Fairness Indicator
"Lack of ML lifecycle management"

**Scenario:**

Researcher creates a Notebook
He/she does everything on it from PoC to production

Data prep, transform, train, validation, serving, and deploy. Got high accuracy on prod service. Yay!

... and forget about the project
"Lack of ML lifecycle management"

One year later, somebody found the accuracy had been dropping slowly

The director asks the researcher to update the model ASAP

The researcher somehow finds the old Notebook on laptop. Tries to remember how to go through every process manually

And wonders, why am I doing the emergency plumbing?? Is this my job?
Solution: ML lifecycle management

- Integrated Frontend for Job Management, Monitoring, Debugging, Data/Model/Evaluation Visualization
- Shared Configuration Framework and Job Orchestration
  - Tuner
  - Data Ingestion
  - Data Analysis
  - Data Transformation
  - Data Validation
  - Trainer
  - Model Evaluation and Validation
  - Serving
  - Logging
- Shared Utilities for Garbage Collection, Data Access Controls
- Pipeline Storage
Kubeflow Pipelines

Enable developers to build custom ML workflows by easily “stitching” and connecting various components like building blocks.
What Constitutes a Kubeflow Pipeline

**Containerized implementations of ML Tasks**
- Containers provide portability, repeatability and encapsulation
- A containerized task can invoke other services like AI Platform Training and Prediction, Dataflow or Dataproc
- Customers can add custom tasks

**Specification of the sequence of steps**
- Specified via Python DSL

**Input Parameters**
- A “Job” = Pipeline invoked w/ specific parameters
Visual depiction of pipeline topology

1. **Data pre-processing & validation**
2. **Feature engineering**
3. **Training models in parallel**
4. **Score on test set & pick best model**
5. **Deploy for serving**
Easy comparison and analysis of runs
Intermediate

Kubeflow: ML App Development

Google Cloud
Summary
Development anti-patterns
Deployment anti-patterns
Operation anti-patterns
References:


Thank you!