Building ML Products with Kubeflow

Jeremy Lewi(jlewi@google.com)
Stephan Fabel(stephan.fabel@canonical.com)
Agenda

- Kubeflow background & rationale
- End to end example
  - Deployment Options
    - On Prem with Canonical Distribution of Kubernetes (CDK)
    - In the cloud using Google Kubernetes Engine (GKE)
  - Walk through building a product with Kubeflow
- Summary & Roadmap
Several Talks Related To Kubeflow

- **Tuesday, May 1:**
  - Red Hat OpenShift Commons Machine Learning Reception Panel

- **Wednesday, May 2:**
  - Kubeflow Intro - Michał Jastrzębski & Ala Raddaoui, Intel

- **Thursday, May 3:**
  - Kubeflow Deep Dive - Jeremy Lewi, Google
  - Building ML Products with Kubeflow - Jeremy Lewi, Google & Stephan Fabel, Canonical
  - Compliant Data Management and Machine Learning on Kubernetes - Daniel Whitenack, Pachyderm

- **Friday, May 4:**
  - Keynote: Kubeflow ML on Kubernetes - David Aronchick & Vishnu Kannan, Google
  - Conquering a Kubeflow Kubernetes Cluster with ksonnet, Ark, and Sonobuoy - Kris Nova, Heptio & David Aronchick, Google
  - Serving ML Models at Scale with Seldon and Kubeflow - Clive Cox, Seldon.io
ML is everywhere
Perception: ML Products are mostly about ML

- Data Collection
- Configuration
- Feature Extraction
- Data Verification
- Process Management Tools
- Analysis Tools
- Machine Resource Management
- Serving Infrastructure
- Monitoring
Reality: ML Requires DevOps; lots of it

Source: Sculley et al.: Hidden Technical Debt in Machine Learning Systems
You Know What’s Really Good at DevOps
Containers and Kubernetes
Kubeflow: Build Portable ML Products Using Kubernetes
What is Kubeflow?

● Community
  ○ Who: Data scientists, ml researchers, software engineers, product managers
  ○ What: Making Kubernetes the best platform for ML
  ○ Why: Because building a platform is too big a problem to tackle alone

● A K8s native platform for ML
  ○ Run wherever K8s runs
  ○ Use K8s for managing ML tasks
    ■ e.g. CRDs to manage distributed training and model deployment
  ○ Adopt K8s patterns
    ■ e.g. microservices and managing infrastructure declaratively
  ○ ksonnet packages to manage infrastructure declaratively
  ○ Support multiple ML frameworks (TensorFlow, PyTorch, scikits, xgboost etc...)
  ○ E2E solutions illustrating ML products built on Kubeflow
End to End Example
Deployment to on-premise and public cloud

- Rubber hits the road: with a great application, we now need to deploy it to our dev/test and production clusters
- Examples:
  - Canonical Distribution of Kubernetes (CDK), delivered on-prem
  - Google’s Kubernetes Engine (GKE) for production in the cloud

Deployment demo: [link](#)
Overview

Identify a problem ➔ Experiment ➔ Train ➔ Deploy ➔ Operate
Problem: GitHub Issues With Uninformative Titles

- User files bug; doesn't know what's wrong so generic title "X doesn't work"
  - Subsequent back and forth identifies the issue
  - Would like to update the title to better summarize the issue
  - Example: kubeflow/kubeflow#340
    - "Some setting problems--A new guy needs a little help" -> "[ksonnet] RUNTIME ERROR: Field does not exist: core"

- Start a feature request; narrow it down via subsequent discussion
  - kubeflow/kubeflow#265
  - "Connectors for popular DBs" -> "Connect to external MySQL/PostGres DB from Jupyter"
GitHub Issue

LabelEncoder transform fails for empty lists (for certain inputs)

#10458

Issue Title

Issue Body

Python 3.6.3, scikit_learn 0.19.1

Depending on which datatypes were used to fit the LabelEncoder, transforming empty lists works or not. Expected behavior would be that empty arrays are returned in both cases.

```python
>>> from sklearn.preprocessing import LabelEncoder
>>> le = LabelEncoder()
>>> le.fit([1.2])
```
Keeping GitHub Issue Titles Up To Date is Toil; Can We Automate This?
Demo
Enable exploration/experimentation

- Data scientists identify a dataset; GitHub Archive
- Download a slice of data
- Try different preprocessing
  - Tokenization, vocab generation etc...
  - Histograms of document length used to manually pick a padding length
- Try different models
  - Linear models, decision trees, deep learning
- Compute/plot various statistics to analyze the data
- Jupyter is one of the preferred tools of data scientists for exploration/analysis
- Jupyter Notebook
JupyterHub on K8s -> security & reproducibility

- Kubeflow runs JupyterHub on K8s
  - Uses [KubeSpawner](#); Project within JupyterHub
- Can provide stock or custom Jupyter kernels with packages that a team needs
  - Everyone gets the same images
  - Can be centrally managed
- More resources - leverage K8s scheduling to manage RAM/CPUs/GPUs
  - Scale beyond what a laptop can do
- Centralized storage - Use K8s volumes to manage data that can be shared by a team
- Security - Data never leaves the secure network; not on data scientist's laptop
  - SecOps policies can be managed centrally by IT experts.
Specifying the Environment

**Image**
repo/image:tag

**CPU**
200m, 1.0, 2.5, etc

**Memory**
100Mi, 1.5Gi

**Extra Resource Limits**
{nvidia.com/gpu': '3'}

**Spawn**
What does Kubeflow Add

- ksonnet package to manage JupyterHub
- Curated Jupyter notebook images with ML packages (TF, TFX, Beam, etc...)
- Integration with other Kubeflow packages
  - e.g Pachyderm to manage datasets
Result: Recipe for building a model

- **Our model comes from Hamel Husain**
  - blog post "How to Create Data Products That Are Magical Using Sequence-ToSequence Models"

- **Built model in notebook**
  - Down sampled data: ~ 2 Million out of 5 million issues
  - Ran preprocessing
  - Trained the model
  - Generated predictions in the notebook
Scaling Preprocessing and Training

- GitHub Archive ~ 5 M issues
  - Blog post sampled ~ 2 M issues
    - Sampled preprocessing takes ~ 1 hour with 8 cores and 60 GB of RAM
- Preprocessing full dataset
  - 2-3 hours 20 CPUs 220 GB of Ram
- Run asynchronous batch jobs
  - K8s Job controller
  - Scale horizontally
  - Scale vertically by adding more CPU/GPUs or RAM
- Use TFJob to run distributed asynchronous training
What does Kubeflow Add

- **TFJob - K8s CRD for TensorFlow jobs**
  - especially valuable for distributed jobs
- **Model analysis**
  - Deployment/management of TensorBoard
  - TF model analysis packages in Jupyter
- **Coming: Integration with model DB and hyper parameter tuning**
Result: A working model ... in a notebook
Turning the model into a product
Deploying the model

- SeldonIO provides a model server for Python models and TF

```python
class IssueSummarization(object):
    def __init__(self):
        with open('body_pp.dpkl', 'rb') as body_file:
            body_pp = dpickle.load(body_file)
        with open('title_pp.dpkl', 'rb') as title_file:
            title_pp = dpickle.load(title_file)
        self.model = Seq2Seq_Inference(
            encoder_preprocessor=body_pp,
            decoder_preprocessor=title_pp,
            seq2seq_model=load_model('seq2seq_model_tutorial.h5'))

    def predict(self, input_text, feature_names):
        # pylint: disable=unused-argument
        return np.asarray([[self.model.generate_issue_title(body[0])[1] for body in input_text]])
```

- SeldonIO has a K8s CRD for deploying/managing models
- Kubeflow has ksonnet packages for deploying Seldon CRD and Seldon models
- Kubeflow also supports TFServing
So we have an API; now we want a web app
Deploying a **Web app** = K8s Bread Butter

**Github Issue Summarization**

Instructions: This is a demo of the github issue summarization model by Hamel Hussain. Enter the body of a github issue or the uri of a github issue and click on Submit. The model then tries to generate a title or summary of the issue.

Enter Github Issue Body

OR Enter Github Issue URL

https://github.com/kubeflow/kubeflow/issues/157

This demo is run using Kubeflow - a machine learning toolkit for Kubernetes. Kubeflow is dedicated to making deployment of machine learning on Kubernetes simple, portable and scalable.
Result
gh-demo.kubeflow.org
A distributed system with multiple microservices
Roadmap

- **Released 0.1 in April**
  - Core components: Argo, JupyterHub, TfJob - v1alpha1, Seldon, TFServing

- **0.2 ETA EOQ2**
  - New components: Katib for HP Tuning, PyTorch Operator, Batch Inference, Horvod Integration, Central UI, click to deploy
  - Improvements: TfJob - v1alpha2, ISTIO integration for serving

- **Kubeflow 1.0 targeting Kubecon USA 2018**
  - Demonstrate continuous integration
    - Continuously train the model as new data arrives
    - Evaluate the model
    - Rollout good models into production
Summary

- ML products are distributed systems with significant dev ops challenges
- Kubeflow is a K8s native platform to simplify building and deploying ML applications on-prem and in the cloud
Find out more

Code:

https://github.com/kubeflow/examples/tree/master/github_issue_summarization

Try it on [Katacoda](https://github.com/kubeflow/kubeflow)

[Hamel Husain's Blog Post](https://github.com/kubeflow/kubeflow-discuss@googlegroups.com)
Special Thanks

Hamel Husain - Datascientist at GitHub who built the model

Ankush Agarwal and Michelle Casbon - Googlers who productionized it
Appendix
Kubeflow Provides...

- Packages for each step in building ML products
  - Two types of packages
    - Packages developed outside Kubeflow but integrated into Kubeflow
      - e.g. Argo, JupyterHub, SeldonIO, Pachyderm, Ambassador, ....
    - Packages developed with Kubeflow
      - e.g. K8s custom resources for training models, Katib, KVC
  - Tooling to combine the packages into ML Applications
ML Applications with Kubeflow

- Use ksonnet to build ML applications
- Move those applications between environments
  - local -> cloud
  - dev -> test -> prod