Build an Event Driven Machine Learning Pipeline on Kubernetes

Hou Gang Liu
Advisory Software Developer IBM
kubeflow katib/manifest maintainer

Animesh Singh
STSM and Program Director IBM
kubeflow kfserving maintainer

hougangliu

animeshsingh
**Center for Open Source Data and AI Technologies (CODAIT)**

**Code** – Build and improve practical frameworks to enable more developers to realize immediate value.

**Content** – Showcase solutions for complex and real-world AI problems.

**Community** – Bring developers and data scientists to engage with IBM

- Team contributes to over 10 open source projects
- 17 committers and many contributors in Apache projects
- Over 1100 JIRAs and 66,000 lines of code committed to Apache Spark itself; over 65,000 LoC into SystemML
- Over 25 product lines within IBM leveraging Apache Spark
- Speakers at over 100 conferences, meetups, unconferences and more
1997

GARRY KASPAROV

1997
2011

IBM Watson

Jeopardy

2017

AlphaGo

Apple's releases Siri

1997

...

Facebook's face recognition

2015

2016

Siri gets deep learning

IBM Deep Blue

chess

AlexNet

Progress in Deep Learning

2012

Introduced deep learning with GPUs
2011 IBM Watson
2017 AlphaGo

1997 Apple's release Siri
2015 Facebook's face recognition
2016 Siri gets deep learning
2012 IBM Deep Blue chess
2012 Introduced deep learning with GPUs
2012 AlexNet
2017
Progress in Deep Learning

IBM Deep Blue chess

IBM Watson Jeopardy

1997

…

Apple’s releases Siri

2011

AlexNet

2012

Facebook’s face recognition

2015

Siri gets deep learning

2016

AlphaGo

2017

Introduced deep learning with GPUs

1997

IBM’s releases Siri

2011

AlexNet

2012

Facebook’s face recognition

2015

Siri gets deep learning

2016

AlphaGo

2017

Introduced deep learning with GPUs
A human brain has:
- 200 billion neurons
- 32 trillion connections between them

Deep Learning = Training Artificial Neural Networks
- 25 million “neurons”
- 100 million connections (parameters)
In reality ...

*Source: Hidden Technical Debt in Machine Learning Systems*
Neural Network Design Workflow

- design neural network
- data
- neural network
- hyperparameters
- HPO

Performance meets needs?
- optimal hyperparameters
- NO
- Start another experiment

Domain data
Neural Network Design Workflow

1. Design neural network
2. Perform Hyperparameter Optimization (HPO)
   - neural network structure
   - hyperparameters
3. Evaluate model performance:
   - Performance meets needs?
     - Yes: Deploy trained model
     - No: Start another experiment
4. Optimal hyperparameters:
   - Still good!
   - BAD

Domain data

Cloud
Let's understand it from the context of an AI Lifecycle
We need a Cloud native AI Platform to build, train, deploy and monitor Models
Many tools available to build initial models

Prepared and Analyzed Data

Create Model

Assign Hyperparameters and Train

Monitor

Validate and Deploy

Trained Model

Deployed Model

Initial Model
Neural Network Modeller within Watson Studio
An intuitive drag-and-drop, no-code interface for designing neural network structure

- Drag-and-drop network layers
- Real-time validation of network flow
- Customize layer by setting hyperparameters
- Generate CPU or GPU compatible code
- Save as popular framework code
- Export as a python notebook
- Execute as batch experiment
Many tools to train machine learning and deep learning models
Training is accomplished. Model is ready – Can we trust it?

Can the model be trusted?
What does it take to trust a decision made by a machine? (Other than that it is 99% accurate)?

Is it fair?  
Is it easy to understand?  
Did anyone tamper with it?  
Is it accountable?
Our vision for Trusted AI
Pillars of trust, woven into the lifecycle of an AI application

FAIRNESS  EXPLAINABILITY  ROBUSTNESS  ASSURANCE

supported by an instrumented platform
AIOpenScale
So let's start with vulnerability detection of Models?
Enter: Adversarial Robustness Toolbox

- Prepared and Analyzed Data
- Initial Model
- Trained Model
- Deployed Model
- Monitor
- Assign Hyperparameters and Train
- Validate and Deploy

ART
IBM Adversarial Robustness Toolbox

https://github.com/IBM/adversarial-robustness-toolbox

ART is a library dedicated to adversarial machine learning. Its purpose is to allow rapid crafting and analysis of attack and defense methods for machine learning models. The Adversarial Robustness Toolbox provides an implementation for many state-of-the-art methods for attacking and defending classifiers.

The Adversarial Robustness Toolbox contains implementations of the following attacks:
- Deep Fool (Moosavi-Dezfooli et al., 2015)
- Fast Gradient Method (Goodfellow et al., 2014)
- Jacobian Saliency Map (Papernot et al., 2016)
- Universal Perturbation (Moosavi-Dezfooli et al., 2016)
- Virtual Adversarial Method (Moosavi-Dezfooli et al., 2015)
- C&W Attack (Carlini and Wagner, 2016)
- NewtonFool (Jang et al., 2017)

The following defense methods are also supported:
- Feature squeezing (Xu et al., 2017)
- Spatial smoothing (Xu et al., 2017)
- Label smoothing (Warde-Farley and Goodfellow, 2016)
- Adversarial training (Szegedy et al., 2013)
- Virtual adversarial training (Miyato et al., 2017)
Evasion attacks
- FGSM
- JSMA
- BIM
- PGD
- Carlini & Wagner
- DeepFool
- NewtonFool
- Universal perturbation

Evasion defenses
- Feature squeezing
- Spatial smoothing
- Label smoothing
- Adversarial training
- Virtual adversarial training
- Thermometer encoding
- Gaussian data augmentation

Poisoning detection
- Detection based on clustering activations
- Proof of attack strategy

Robustness metrics
- CLEVER
- Empirical robustness
- Loss sensitivity

Unified model API
- Training
- Prediction
- Access to loss and prediction gradients

Implementation for state-of-the-art methods for attacking and defending classifiers.
ART Demo: https://art-demo.mybluemix.net/

Try it out
1. Select an image to target

2. Simulate Attack
   Adversarial noise type
   C&W Attack

Determine strength
   None  low  med  high

3. Defend attack
   Gaussian Noise
   Spatial Smoothing
   Feature Squeezing

Original  Modified

94%
Siamese cat
Robustness check accomplished. How do we check for bias throughout lifecycle?

- Is the dataset biased?
- Are predictions biased?
- Are model weights biased?
Unwanted bias and algorithmic fairness

Machine learning, by its very nature, is always a form of statistical discrimination.

Discrimination becomes objectionable when it places certain privileged groups at systematic advantage and certain unprivileged groups at systematic disadvantage.

Illegal in certain contexts.
Unwanted bias and algorithmic fairness

Machine learning, by its very nature, is always a form of statistical discrimination

Unwanted bias in training data yields models with unwanted bias that scale out

Prejudice in labels

Undersampling or oversampling
AIF360 toolkit is an open-source library to help detect and remove bias in machine learning models.

The AI Fairness 360 Python package includes a comprehensive set of metrics for datasets and models to test for biases, explanations for these metrics, and algorithms to mitigate bias in datasets and models.

**Toolbox**
- Fairness metrics (30+)
- Fairness metric explanations
- Bias mitigation algorithms (9+)

**Supported bias mitigation algorithms**
- Optimized Preprocessing (Calmon et al., 2017)
- Disparate Impact Remover (Feldman et al., 2015)
- Equalized Odds Postprocessing (Hardt et al., 2016)
- Reweighing (Kamiran and Calders, 2012)
- Reject Option Classification (Kamiran et al., 2012)
- Prejudice Remover Regularizer (Kamishima et al., 2012)
- Calibrated Equalized Odds Postprocessing (Pleiss et al., 2017)
- Learning Fair Representations (Zemel et al., 2013)
- Adversarial Debiasing (Zhang et al., 2018)

**Supported fairness metrics**
- Comprehensive set of group fairness metrics derived from selection rates and error rates
- Comprehensive set of sample distortion metrics
- Generalized Entropy Index (Speicher et al., 2018)
Model is trained, tested and validated. Then we can deploy it. Do we need anything else?

How can I monitor and trace the model?

How can I manage multiple version of my model to enable dark launch, A/B test and traffic shift easily?

How can I add and enforce policies to the model easily?
Istio
An open service mesh platform to connect, observe, secure, and control microservices.

**Connect:** Traffic Control, Discovery, Load Balancing, Resiliency

**Observe:** Metrics, Logging, Tracing

**Secure:** Encryption (TLS), Authentication, and Authorization of service-to-service communication

**Control:** Policy Enforcement
So AI in general and Deep Learning in particular are very iterative and repetitive. And they need Cloud. Why?
AI requires the strength of HPC & GPUs
Ability to scale AI workloads on demand
Ability to utilize various technologies and achieve high performance computing.

1. Model/Data Parallelism

2. MPI/NCCL

NCCL (pronounced "Nickel") is a stand-alone library of standard collective communication routines for GPUs, implementing all-reduce, all-gather, reduce, broadcast, and reduce-scatter. It has been optimized to achieve high bandwidth on platforms using PCIe, NVLink, NVswitch, as well as networking using InfiniBand Verbs or TCP/IP sockets.

NCCL supports an arbitrary number of GPUs installed in a single node or across multiple nodes, and can be used in either single- or multi-process (e.g., MPI) applications.
To scale we need to go Cloud native for AI
Access to elastic compute leveraging Kubernetes

Auto-allocation means infrastructure is used only when needed.
Model training distributed across containers

- NVIDIA GPUs
- Kubernetes
- container orchestration
- training runs
- containers
- server cluster

Cloud Object Storage

dataset
Kubernetes is not the end game

Kubernetes is a platform for building platforms. It's a better place to start; not the endgame.

556  1:04 PM - Nov 27, 2017
Numpy
Jupyter
TF.Transform
TF.Estimator
Docker
Seldon

Experiment Tracking
Declarative
HP Tuning
Profiling
Validation

Resource
Scheduling
Access
Drivers
Orchestration
Lifecycle
Networking

Source: kubeCon Barcelona 2019
Oh, you want to use ML on K8s?

First, can you become an expert in ...

- Containers
- Packaging
- Kubernetes service endpoints
- Persistent volumes
- Scaling
- Immutable deployments
- GPUs, Drivers & the GPL
- Cloud APIs
- DevOps
- ...

Source: kubeCon Barcelona 2019
Kubeflow architecture

- Make it super easy to deploy and administer a platform
  - Leverage KF & non KF components
- Tie it together using
  - Orchestration
    - Combine components into complex workflows
  - Metadata
    - Collect data from multiple components
Introduce Kubeflow
- JupyterHub
- TFJob
- TFServing

May 2018
- Kubeflow 0.1
  - Argo
  - Ambassador
  - Selldon

Aug 2018
- Kubeflow 0.2
  - Katib
  - HP Tuning
  - Kubebench
  - PyTorchs

Sep 2018
- Contributor Summit

Oct 2018
- Kubeflow 0.3
  - kfctl.sh
  - TFJob v1alpha2

2019 Jan
- Kubeflow 0.4
  - Pipelines
  - JupyterHub UI refresh
  - TFJob, PyTorch beta

2019 April
- Kubeflow 0.5
  - Fairing
  - Jupyter WebApp + CR
Getting Started

- Getting started with Kubeflow
  Quickly get running with your ML workflow on an existing Kubernetes installation

- Microk8s for Kubeflow
  Quickly get Kubeflow running locally on native hypervisors

- Minikube for Kubeflow
  Quickly get Kubeflow running locally

- Kubernetes Engine for Kubeflow
  Get Kubeflow running on Google Cloud Platform. This guide is a quickstart to deploying Kubeflow on Google Kubernetes Engine

- Requirements for Kubeflow
  Get more detailed information about using Kubeflow and its components
@dsl.pipeline(
    name='Object detection',
    description='Object detection'
)
def object_detection(worker=3):
    getData = get_data()
    pre_process = pre_process(getData.output)
    hpo = hyperparameter_tune(pre_process.output)
    train = start_train(hpo.output, worker)
    r_check = robustness_check(train.output)
    f_check = fairness_check(train.output)
    deploy = deploy_model(r_check.output, f_check.output)

# dsl-compile --py object_detection.py --output object_detection.tgz
Kubeflow pipeline

Pipelines

object detection

Pipe: object detection

- get-data
- pre-process
- hyperparameter-tune
- train
- fairness-check
- robustness-check
- deploy-model
Infact, we need a transparent, trusted and **automated** AI Pipeline

- Prepared and Analyzed Data
- Assign Hyperparameters and Train
- Monitor
- Validate and Deploy
- Deployed Model
- Trained Model
- Initial Model

Questions:
- Has the training data changed?
- Is the dataset biased?
- Are predictions biased?
- Are model predictions less accurate?
- Is model training showing increasing loss?
- Are Hyperparameters suboptimal?
- Is the model vulnerable to adversarial attacks?
- Are model weights biased?
Transparent, trusted, automated, event driven and auditable AI Pipeline

- **Trigger**: Trained Model is showing increasing loss
  - Prepare Data
  - Train
  - Deploy

- **Prepare and Analyzed Data**
  - Assign Hyperparameters and Train

- **Initial Model**
  - Create Model
  - Monitor
  - Validate and Deploy

- **Trained Model**
  - Deployed Model

- **Trigger**: Data changed
  - Implement Defense
  - Train
  - Deploy

- **Trigger**: Bias Detected
  - Optimize Hyperparameters
  - Train
  - Deploy

- **Trigger**: Model is vulnerable to attack
  - Harden
  - Train
  - Deploy

- **Trigger**: Model performance is suboptimal
  - Optimize Hyperparameters
  - Train
  - Deploy
Trigger: Trained Model is showing increasing loss

Prepare Data
Train
Deploy

Trigger: Data changed

Prepare and Analyzed Data

Trigger: Bias Detected

Actions

Optimize Hyperparameters
Train

Trigger: Model is vulnerable to attack

Implement Defense
Train
Deploy

Trigger: Model performance is biased

Debias
Train
Deploy

Initial Model
Create Model
Assign Hyperparameters and Train

Monitor

Prepare and Analyzed Data

Validate and Deploy

Deployment Model

Trained Model

Optimize Hyperparameters
Train
Knative
Kubernetes-based platform to build, deploy, and manage modern serverless workloads.

**Build**
Provides easy-to-use, simple source-to-container builds, so you can focus on writing code and know how to build it. Knative solves for the common challenges of building containers and runs it on cluster.

**Serving**
Run serverless containers on Kubernetes with ease, Knative takes care of the details of networking, autoscaling (even to zero), and revision tracking. You just have to focus on your core logic.

**Eventing**
Universal subscription, delivery, and management of events. Build modern apps by attaching compute to a data stream with declarative event connectivity and developer-friendly object model.
Build — Source-to-container build orchestration

Knative Build Components

- Build
- Builder
- BuildTemplate

For example, you can write a build that uses Kubernetes-native resources to obtain your source code from a repository, build a container image, then run that image.

- A Build can include multiple steps where each step specifies a Builder.
- A Builder is a type of container image that you create to accomplish any task, whether that's a single step in a process, or the whole process itself.
- The steps in a Build can push to a repository.
- A BuildTemplate can be used to defined reusable parameterized templates.
Serving — Request-driven compute model, scale to zero, autoscaling, routing and managing traffic

Knative Serving components

- Configuration
  - Desired current state of deployment (#HEAD)
  - Records both code and configuration (separated, ala 12 factor)
  - Stamps out builds/revisions as it is updated

- Revision
  - Code and configuration snapshot
  - k8s infra: Deployment, ReplicaSet, Pods, etc

- Route
  - Traffic assignment to Revisions (fractional scaling or by name)
  - Built using Istio

- Service
  - Provides a simple entry point for UI and CLI tooling to achieve common behavior
  - Acts as a top-level controller to orchestrate Route and Configuration.
Broker and Trigger are CRDs providing an event delivery mechanism that hides the details of event routing from the event producer and event consumer.

The Event Registry maintains a catalog of the event types that can be consumed from the different Brokers.

Event Sources are Kubernetes Custom Resources which provide a mechanism for registering interest in a class of events from a particular software system.

Channels are Kubernetes Custom Resources which define a single event forwarding and persistence layer.
<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Support Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS SQS</strong></td>
<td>Proof of Concept</td>
<td>Brings AWS Simple Queue Service messages into Knative.</td>
</tr>
<tr>
<td><strong>Apache Camel</strong></td>
<td>Proof of Concept</td>
<td>Allows to use Apache Camel components for pushing events into Knative.</td>
</tr>
<tr>
<td><strong>Apache Kafka</strong></td>
<td>Proof of Concept</td>
<td>Brings Apache Kafka messages into Knative.</td>
</tr>
<tr>
<td><strong>BitBucket</strong></td>
<td>Proof of Concept</td>
<td>Registers for events of the specified types on the specified BitBucket organization/repository. Brings those events into Knative.</td>
</tr>
<tr>
<td><strong>Cron Job</strong></td>
<td>Proof of Concept</td>
<td>Uses an in-memory timer to produce events on the specified Cron schedule.</td>
</tr>
<tr>
<td><strong>GCP PubSub</strong></td>
<td>Proof of Concept</td>
<td>Brings GCP PubSub messages into Knative.</td>
</tr>
<tr>
<td><strong>GitHub</strong></td>
<td>Proof of Concept</td>
<td>Registers for events of the specified types on the specified GitHub organization/repository. Brings those events into Knative.</td>
</tr>
<tr>
<td><strong>GitLab</strong></td>
<td>Proof of Concept</td>
<td>Registers for events of the specified types on the specified GitLab repository. Brings those events into Knative.</td>
</tr>
<tr>
<td><strong>Google Cloud Scheduler</strong></td>
<td>Active Development</td>
<td>Create, update, and delete Google Cloud Scheduler Jobs. When those jobs are triggered, receive the event inside Knative.</td>
</tr>
<tr>
<td><strong>Google Cloud Storage</strong></td>
<td>Active Development</td>
<td>Registers for events of the specified types on the specified Google Cloud Storage bucket and optional object prefix. Brings those events into Knative.</td>
</tr>
<tr>
<td><strong>Kubernetes Api Server</strong></td>
<td>Active Development</td>
<td>Brings Kubernetes resource changes into Knative as references or as full resources.</td>
</tr>
</tbody>
</table>
Event Driven ML pipeline

```yaml
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: GitHubSource
metadata:
  name: my-github-source
spec:
  eventTypes:
    - pull_request
  ownerAndRepository: hougangliu/test
  accessToken:
    secretKeyRef:
      name: my-githubsecret
      key: accessToken
  secretToken:
    secretKeyRef:
      name: my-githubsecret
      key: secretToken
sink:
  apiVersion: v1
  kind: Service
  name: pipeline-launcher
  namespace: kube-system
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: pipeline-launcher
  namespace: kube-system
data:
  configFile: |
    version: v1alpha1
    eventMap:
      github.com/hougangliu/object_detection:
        pull:
          - "object_detection"
        push:
          - "object_detection"
      github.ibm.com/hougangliu/test:
        pull:
          - "test1"
          - "test2"
        push:
          - "test"
```
Event Driven ML pipeline

```python
@dsl.pipeline(
    name='Object detection',
    description='Object detection'
)
def object_detection(
    worker=3,
    new_image_name="hougangliu/object_detection:latest"):
    getData = get_data()
    pre_process = pre_process(getData.output)
    new_image = build_image(new_image_name)
    hpo = hyperparameter_tune(pre_process.output, new_image.output)
    train = start_train(hpo.output, new_image.output, worker)
    r_check = robustness_check(train.output)
    f_check = fairness_check(train.output)
    deploy = deploy_model(r_check.output, f_check.output)
```

```
apiVersion: build.knative.dev/v1alpha1
kind: Build
metadata:
  name: build-objective-detection
spec:
  serviceAccountName: build-auth
  source:
    git:
      url: https://github.com/hougangliu/object_detection.git
      revision: master
  steps:
  - name: hougangliu/image-build:latest
    image: hougangliu/image-build:latest
    args: ["make", "build"]
  - name: build-image
    image: hougangliu/image-push:latest
    args: ["make", "push"]
  - name: push-image
    image: hougangliu/image-push:latest
    args: ["make", "push"]
    volumeMounts:
      - name: docker-socket-example
        mountPath: /var/run/docker.sock
```

---

Graph:
- **get-data**
- **build-new-image**
- **pre-process**
- **hyperparameter-tune**
- **train**
- **fairness-check**
- **robustness-check**
- **deploy-model**
Together: A Transparent, and trusted event driven Open Source AI Pipeline

- Prepared and Analyzed Data
- Create Model
- Assign Hyperparameters and Train
- Initial Model
- Trained Model
- Deployed Model
- Validate and Deploy
- Jupyter Enterprise Gateway
- kubectl
- AIF360
- MAX
- istio
- Kubeflow
- kubernetes
- TensorFlow
- Keras
- PyTorch
- PaddlePaddle
- mxnet
- tensorrt-inference-server
- seldon
- kfservice
- Anaconda
- R Studio
- SPSS Modeler
- kube-batch
Build an Event Driven Machine Learning Pipeline on Kubernetes

THANKS