YamFlow Documentation Documentation

Release 0.9

Dr. Thomas Lee

Apr 21, 2019

Contents

1 Pipelining **1**
- 1.1 Introduction . . . 1
- 1.2 Acquire Data . . . 2
- 1.3 Clean Data . . . 2
- 1.4 Label Data . . . 2
- 1.5 Preprocess Inference Inputs . . . 2
- 1.6 Preprocess Training Data . . . 2
- 1.7 Validate Inference Outputs . . . 2

2 Modeling **3**
- 2.1 Introduction . . . 3
- 2.2 Explore Data . . . 3
- 2.3 Experiment Models . . . 4
- 2.4 Optimize Hyperparameters . . . 4
- 2.5 Code Model . . . 4

3 Training **5**
- 3.1 Introduction . . . 5
- 3.2 Train Model . . . 5
- 3.3 Validate Model . . . 6
- 3.4 Register Model . . . 6

4 Inference **7**
- 4.1 Introduction . . . 7
- 4.2 Develop Inference Application . . . 7
- 4.3 Deploy Inference Application . . . 8
- 4.4 Update Model . . . 8
- 4.5 Serve Model . . . 8

5 Introduction **9**

6 YamFlow Overview **11**

CHAPTER 1

Pipelining

1.1 Introduction

The first swimlane in Fig. 6.1 illustrates the Pipelining work stream. In this stream, the data engineer prepares the data sets for ML modeling, training, and inference. The activities are summarized in the following table.

Activity	Description	Inputs	Outputs
Acquire Data	The necessary raw data from the data sources (e.g., databases, edge devices) are aggregated for ML modeling, training, and inference.	Various data sources	Raw data
Clean Data	The raw data are cleaned so that the data quality is sufficient for labeling or preprocessing.	Raw data	Cleaned data
Label Data	Data points are labeled with expected inference outputs. Manual labeling may be necessary.	Cleaned data	Labeled data
Pre-process Inference Inputs	The data are transformed to the specific format required for input to the inference application	Cleaned data	Inference inputs
Pre-process Training Data	The data sets are transformed to the specific file format required by the ML library for training.	Labeled data	Training data
Validate Inference Outputs	The actual inference outputs are compared against the expected outputs, which can be a manual process. Inaccurate outputs can be selected for relabeling to continously train and improve the ML model in the future.	Actual inference outputs	Selected outputs for relabeling

1.2 Acquire Data

In this activity, the necessary raw data sets are acquired for ML modeling, training, and inference. Data integration may be involved in generating ML data points that cover data fields from multiple data sources. Traditional Extract, Transform, Load (ETL) tools and Enterprise Service Bus (ESB) may be used for simplifying data acquisition and integration. Raw data often come in a variety of forms and thus are commonly stored in data files or a database with a schemaless or columnar structure.

1.3 Clean Data

In this activity, the quality of the data sets is improved up to the standard required for the downstream ML. Through data cleansing, the raw data are cleaned so that the resultant data for ML are as valid, accurate, complete, consistent, and uniform as expected. The data may go through a series of multiple data cleansing procedures, such as value validition, data reformatting, duplicate elimination, and statistical analysis. Since the inference data do not have labels, cleaned data are often directly be used for inference preprocessing.

1.4 Label Data

In this activity, the data are labeled to support the supervised ML. (For unsupervised ML, this step may be skipped.) The labeling process often involves manual labeling but can also be automated in some use cases. This aims to map each data point to the expected inference outputs. An output can be a set of labels (for classification) or values (for regression). A data scientist can use the labeled data set to experiment different ML models and tune the hyperparameters during ML design. After the ML model is designed and programmed, the labeld data set is used to train the model in the *Training* work stream.

1.5 Preprocess Inference Inputs

In this activity, the cleaned and / or labeled data are transformed to the input format required by the inference application to process in the *Inference* work stream, for example, in form of API requests, a message stream, or a data file. How the inference input data should be constructed and submitted depends on the specific input requirements of the inference application.

1.6 Preprocess Training Data

In this activity, the training data sets are produced for modeling *Modeling* work stream or training in the *Training* work stream. The file format can be specific to the ML library used. The training data can be split into two sets: training set and testing set. The training set is used to train the model while the testing set is used to validate the model for its accuracy.

1.7 Validate Inference Outputs

The outputs from the inference application in the *Inference* work stream are fed back for validation against the expected results. The outputs, especially those which are insufficiently accurate, can be relabeled (manually) can be merged into the new training and testing sets for retraining in the *Training* work stream so as to continously improve the model.

CHAPTER 2

Modeling

2.1 Introduction

The second swimlane in Fig. 6.1 illustrates the Modeling work stream. In this work stream, the data scientst explores the data sets prepared in the *Pipelining* work stream, experiments various ML algorithms and models with the data sets, optimizes the hyperparameters to get the desired results, and finally codes the selected and turned ML model.

Activity	Description	Inputs	Outputs
Explore Data	The available data sets are explored and the suitable ones are prepared (in the *Pipelining* work stream for modeling the ML problem.	Raw or cleaned data	Suitable data set
Experiment Models	Various ML algorithms and models (e.g., neural network architectures) are experimented using the data to select an effective model for the ML problem	Suitable data set	Selected ML model
Optimize Hyperparameters	Hyperparameters are turned to test the selected model so that ML training can be efficiently performed.	Selected ML model	Finalized model & hyperparameters
Code Model	The finalized ML model is programmed for training and inference. Application development may be involved.	Finalized model & hyperparameters	ML model coded for training & inference

2.2 Explore Data

In this activity, the data scientist explores the data sets prepared in the Pipelining work stream and selects the suitable data sets for training and inference. Usually, the data scientist uses interactive data science tools (e.g., R Studio, Jupyter Notebook, and Apache Zeppelin) to explore the data sets. If the suitable data sets are unavailable or incomplete or malformed or unclean, further data acquisition, cleansing, or preprocessing work in the Pipelining work stream will be required. The data scientist can also suggest how the suitable data sets should be prepared in the Pipelining work stream.

2.3 Experiment Models

In this activity, the data scientist designs the ML model using the selected data sets. For example, he or she experiments various ML models based on different algorithms (e.g., neural network architectures), trains the models with the training data set, validates the models with the testing data set, and then selects the suitable model based on the inference accuracy measures, such as precision and recall.

2.4 Optimize Hyperparameters

In this activity, the data scientist tunes the hyperparameters so that the model can be further optimized, for example, in terms of training speed and inference accurancy. Different ML algorithms may involve different sets of hyperparameters (e.g., learning rate, model size, number of passes, regularization). The data scientist may need to go back and forth between the *Experiment Models* activity and the *Optimize Hyperparameters* activity in order to optimize ML model.

2.5 Code Model

After the ML model with the hyperparameters are defined, the ML model is coded using ML libraries, e.g., TensorFlow, CNTK, and PyTorch. The coded model is to be incorporated in the inference application.

CHAPTER 3

Training

3.1 Introduction

The third swimlane in Fig. 6.1 illustrates the Training work stream. In this work stream, the ML model coded in the *Modeling* work stream is trained and validated; the validated trained model is registered for deployment in the *Inference* work stream. The activities in this work stream can be done mannually, or automated whereever possible especially for recurrent training.

Activity	Description	Inputs	Outputs
Train Model	The pre-trained ML model coded in the *Modeling* work stream is trainined with a updated training set using the selected hyperparameters.	Pre-trainined model & hyperparameters & training set	Trained model
Validate Model	The new trainied ML model is validated with a testing set on the inference accuracy. Only the model with an expected accuracy is used for deployment.	Trained model & testing set	Validated model
Register Model	The validated model with an expected accuracy is registered to the registry for deployment to the *Inference* work stream.	Validated model	Trained model published to registry

3.2 Train Model

The coded ML model and the selected hyperparameters from the *Modeling* work stream are passed to this activity. The pre-trainined ML model is recurrently trained when a updated training set is generated. This activity can be done mannually. If the trained model needs to be updated regularly, the training process should be programmatically automated. Note that the training set should be representative of the current data pattern for inference.

3.3 Validate Model

The newly trained model is validated against performing inference on a testing set. A validation rule can be set to define the criteria on the acceptance of an updated trained model. For example, the precision and recall of the updated trained model must be above predefined thresholds. Note that the testing set should be representative of the current data pattern for inference.

3.4 Register Model

The trained model accepted after validation can be registered to a registry for deployment to the *Inference* work stream.

CHAPTER 4

Inference

4.1 Introduction

The fourth swimlane in Fig. 6.1 illustrates the Inference work stream. In this work stream, the inference application is developed and deployed. When the ML model is updated, the inference application will load the model from the registry. Finally, the inference application serves the updated model.

Activity	Description	Inputs	Outputs
Develop Inference Application	The inference application is developed using the coded ML model from the *Modeling* work stream	Coded model	Inference app
Deploy Inference Application	The developed inference application is deployed to the server infrastructure.	Developed inference app	Deployed inference app
Update Model	The inference application loads the updated ML model from the registry.	Deployed inference app & updated model	Deployed inference app loaded with updated model
Serve Model	The inferene application serves the loaded ML model. The inference inputs are fed from the *Pipelining* work stream. The	Inference inputs & deployed inference app loaded with updated model	Inference outputs

4.2 Develop Inference Application

The software engineer designs and implements the inferene application that runs the coded ML model. The softwre engineer designs the application based on the business requirements. For example, the application may be designed to process the inference input file by batch or provide RESTful APIs to infer a message stream.

4.3 Deploy Inference Application

The inference application is deployed to the server infrastructure on-premises or in the cloud. The deployment can be done manually or automated with Infrastructure as Code.

4.4 Update Model

The inference application loads the ML model from the registry. When the ML model is updated in the *Training* work stream, the inference application will be triggered to load the updated model automatically.

4.5 Serve Model

After the inference applicaiton has loaded the ML model, the application processes the inference inputs from the *Pipelining* work stream and produces the inference outputs. The inference outputs will be regularly fed to the *Pipelining* work stream for validation. Some inference outputs can be selected to prepare the training data for re-training in the future.

This specification is maintained as a GitHub project. Please leave your feedback on GitHub Issues.

CHAPTER 5

Introduction

YamFlow proposes a reference workflow for the machine learning (ML) development lifecyle. This reference workflow is aimed to provide a canonical taxonomy for practitioners to understand and communicate the activity sequences and the data flows typically involved in a ML process. In addition, YamFlow also serves as the baseline for YAM AI Machinery to design ML programming frameworks for developing interoperable and composable ML tasks.

CHAPTER 6

YamFlow Overview

Fig. 6.1 shows the flowchart of YamFlow, which specifies the overall process of a typical ML implementation in the design time and run time. Although the actual processes of different ML implementations may vary, the activity sequences and data flows should largely resemble YamFlow.

YamFlow consists of the following work streams:

- *Pipelining* specifies the work stream for building the data pipelines for ML modeling, training, and inference in the development time.
- *Modeling* specifies the work stream for exploring the data, and design and code the ML model in the design time.
- *Training* specifies the work stream for training the coded model in the recurrent training time.
- *Inference* specifies the work stream for deploying the inference application and serve the trained model in the runtime.

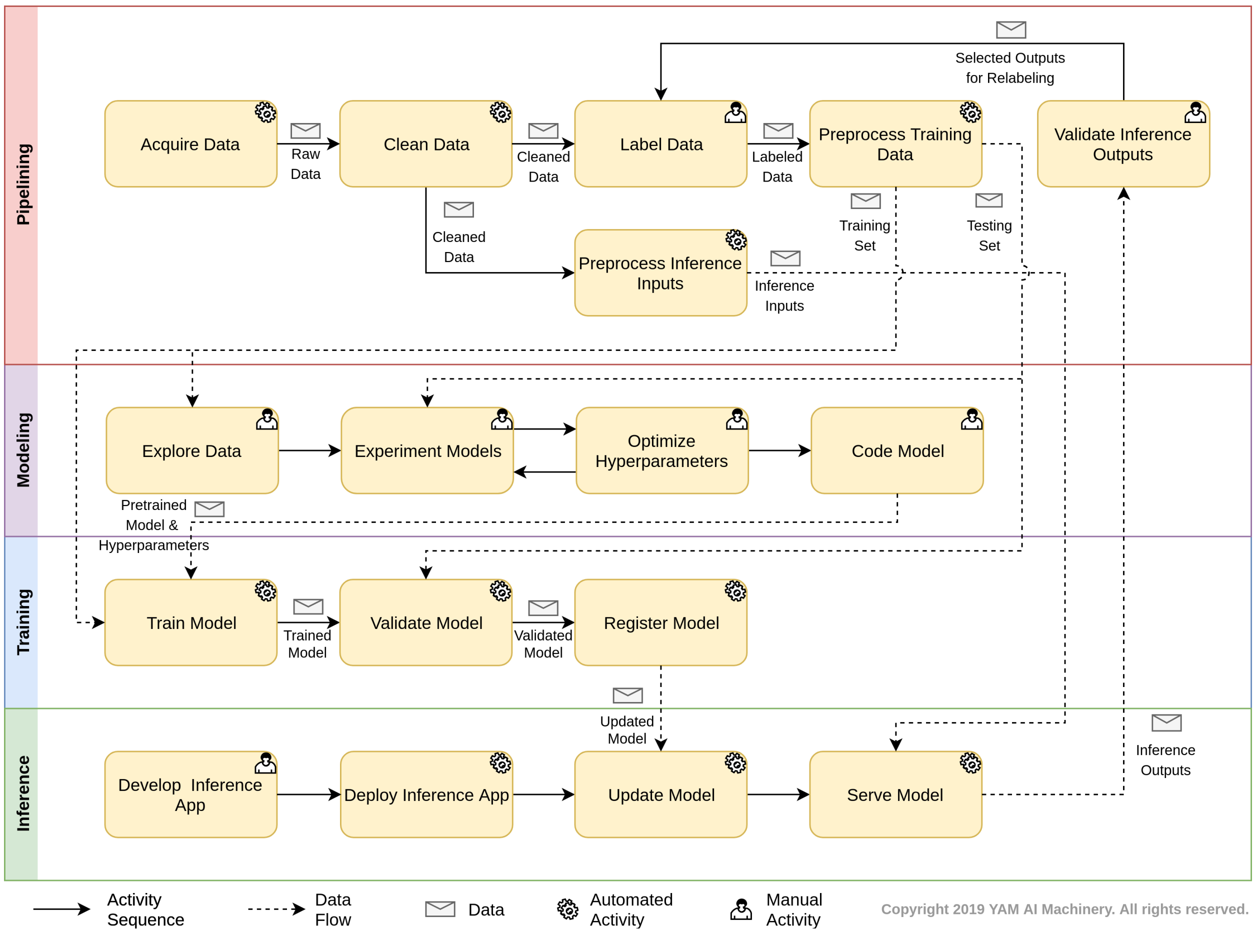

Fig. 6.1: YamFlow Chart.