Bayesian Dropout Approximation (BDA) Engine design

Overview

The BDA Engine acts as a back end for modeling and analysis of data stored in the CESMII Smart Manufacturing Platform. It will exist as a cloud service, served from a Kubernetes cluster instantiated on Microsoft Azure and made available via the MS Azure Marketplace. An instance of the Engine and the K8 cluster on which it runs (together called the BDA Service) is expected to be single-tenant and created by an individual instance of a third-party app (3PA) that wishes to make use of the Service's modeling capabilities and which already interacts with the CESMII SM Platform. This instance of the Service will receive and fulfill requests through a RESTful interface.

In response to requests, the Engine will operate in two modes: training mode and inference mode.

In training mode, the Engine will receive and authenticate requests from the 3PA to train a model based on a specified set of data from the SM Platform. The Engine will then fetch the data and use it to train a model, and return an encapsulated data package called a Model Kernel Object (MKO), which represents all the information necessary to compute predictions with the model and is expected to be retained by the 3PA. Local caching or remote storage of the MKO is a possible additional feature.

In inference mode, the Engine will receive and authenticate requests from the 3PA to use a provided (or possibly cached) MKO and provided model inputs to compute samples of possible model outputs. These samples can then be returned to the 3PA or used in a variety of ways by the Engine's "post-processing" toolset, with the resulting analysis being returned to the 3PA. The tools in this toolset will include capabilities to create histogram images, form probability distribution functions (PDFs), create statistical summaries, integrate certain quantities over PDFs, and visualize model predictions.

Design Considerations

- 3PAs may have high latency or intermittent connections to their instance of the BDA Service.
- The SM Platform service is highly available, low latency, and GraphQL based.
- Instances of the BDA Service are likely not persistent for the useful lifetime of a trained model.
- A typical training dataset from the SM Platform service may be large, 10^5 - 10^6.
- The typical number of degrees of freedom in an MKO is order of magnitude 10^3-10^4.

- Post training, internal data structures in the Engine will be a few hundred MB at the outside, typically much smaller.
- Training task times will be from seconds to multiple hours.
- Inference task times will be from seconds to multiple days of processor time, with typical values being in tens of minutes.
- MKOs should embed provenance and metadata describing what system it represents and which data it is derived from.

Approach

We propose a hybrid microservices architecture within a Kubernetes cloud service, employing shared memory libraries for highly replicated functionality that involves sharing large or complex data structures.

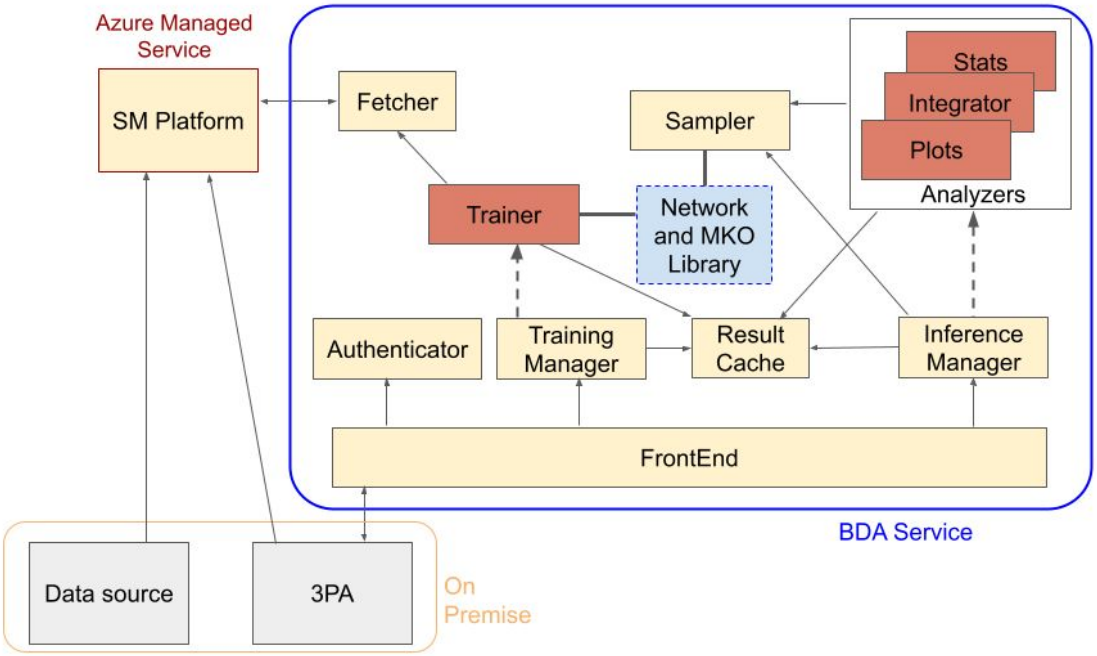

Figure 1: Architecture of BDA Service and external points of contact. Microservices within the BDA Service may be scaled out to meet demand.

External communications

Requests to the BDA Service will be through a FrontEnd via REST API, with java web tokens (JWTs) used to manage ongoing authentication and service continuity. Many types of requests will require long-running operations to return results. In these cases, a token, an expected wait time, and an expiration date will be returned, which can be used to retrieve the forthcoming result.

Communication with the SM Platform will be via its GraphQL interface. Authentication will be via a JWT provided in the 3PA's request.

Internal communications

Internal communications will be via REST APIs, with two major exceptions. First, Trainers and Generators will be executed via POSIX system call and given control parameters via standard input. Second, as the neural network functionality will be provided by the TensorFlow 2.x framework, TensorFlow functions will be called directly by the Trainer service and by the Sampler service. Subsequently, the TensorFlow data structures will be encoded into and decoded from MKOs directly in memory.

Individual Services

FrontEnd - REST API. The FrontEnd service will listen on an exposed port and take HTTPS requests. It will call the Authenticator service for users' authentication. FrontEnd will call either the Training Manager or the Inference Manager to handle a request.

Training Manager - REST API. The Training Manager service sanity checks requests for training a model, determines model hyperparameters, makes estimates for time to complete, and reports errors. It creates instances of Trainers and handles returning MKOs for both short and long operation results. It is responsible for retrieving long-operation MKOs from the Results Cache and returning them to the 3PA.

Trainer - POSIX executable. Instances are created by Training Manager to instantiate a TensorFlow network and train the network parameters. Employs Fetcher to get training data. Posts MKOs to the Results Cache service and exits.

Fetcher - REST API. Fills requests to retrieve data from SM Platform.

Result Cache - REST API. Accepts various results from the Trainer and the Generators and holds them until retrieved by Training Manager or Inference Manager or until the expiration date.

Inference Manager - Rest API. The Inference Manager service sanity checks requests for inference with a model, makes estimates for time to complete, and reports errors. It creates an instance of the appropriate Generator via system call, passing it an array of MKOs and control parameters, and handles returning inference results for both short and long operation results. It is responsible for retrieving long-operation inferences from the Result Cache and returning them to the 3PA.

Generators - Posix executables. A set of analysis tools that all operate on model samples. Samples are obtained by passing an MKO and control parameters to Sampler. There may be

multiple calls to multiple Samplers. Responsibly written Generators release their Samplers when they are done.

Sampler - REST API. Fills requests for obtaining samples, by creating and initializing a network to the state represented by an MKO. Session based, so that a network can be built and persist across multiple requests for samples.