AI Pro: Data Processing Framework for AI Models

Richie Frost[†], Debjyoti Paul[†], Feifei Li
School of Computing, University of Utah
Salt Lake City, Utah, USA
{rfrost,deb,lifeifei}@cs.utah.edu

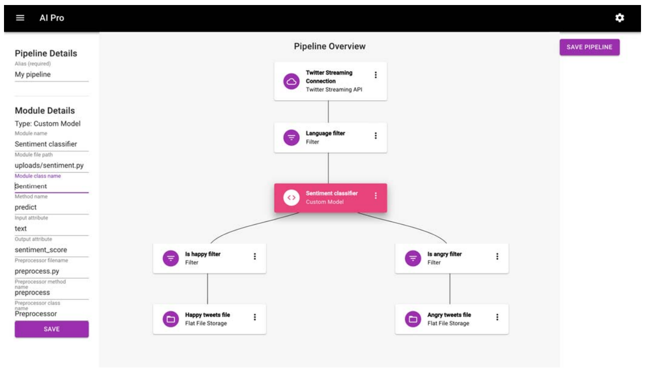
***Abstract*—We present AI Pro, an open-source framework for data processing with Artificial Intelligence (AI) models. Our framework empowers its users with immense capability to transform raw data into meaningful information with a simple configuration file. AI Pro's configuration file generates a data pipeline from start to finish with as many data transformations as desired. AI Pro supports major deep learning frameworks and Open Neural Network Exchange (ONNX), which allows users to choose models from any AI frameworks supported by ONNX. Its wide range of features and user friendly web interface grants everyone the opportunity to broaden their AI application horizons, irrespective of the user's technical expertise. AI Pro has all the quintessential features to perform *end-to-end* data processing, which we demonstrate using two real world scenarios.**

***Index Terms*—data processing, pipeline, framework, realtime, AI processing**

I. Introduction

Recent advances in data science empower businesses more than ever before, but businesses must be able to keep pace with the demand for resources in order to benefit. Technology giants predict that demand for data scientists will grow by 28% by 2020. More than half of the related job positions will be in finance, insurance, professional services and healthcare[1]. Reports suggest that academic institutions will not be able fulfill the demand of skilled workers[2]. This gap between demand and supply will inflate the median salary of data scientists. Smaller companies will find it hard to attract and afford skillful data scientists and will look for alternative solutions and tools. During this period AI will defeat its purpose - to empower every sector and service - if it is not within everyone's reach. Empowering employees with powerful, yet easy-to-use tools is one of the best strategies to confront an increasingly challenging future.

The world is noticing an adoption of AI from many companies now and we believe them to have aggressive plans for the future [3]. These companies are actively looking for process automation, cognitive insights, and trend and time series predictions for optimizing their businesses & services. AI frameworks like Tensorflow, Torch (PyTorch), Caffe, and Keras are empowering data scientists to build complex AI models, solving a wide range of problems in the aforementioned fields. Researchers and collaborators have trained and made publicly available many models in computer vision, natural language processing, neural machine translation, and speech recognition. These models are ready to make predictions on arbitrary input data. However, putting these models into practice requires coding expertise to deploy and execute. Creating data pipelines with any such AI model is not



Pipeline Details
Alias (required)
My pipeline

Module Details
Type: Custom Model
Module name
Sentiment classifier
Module file path
uploads/sentiment.py
Module class name
Sentiment
Method name
predict
Input attribute
text
Output attribute
sentiment_score
Preprocessor filename
preprocess.py
Preprocessor method name
preprocess
Preprocessor class name
Preprocessor

Figure 1: Screenshot of AI Pro.

straightforward and involves overcoming tremendous technical challenges. To address this bottleneck, we have developed **AI Pro**, an open source data processing framework.

AI Pro empowers its users to create data processing pipelines without a single line of code. A configuration file is enough to set up a data pipeline with multiple AI models. Hence, we rightly attribute it *configuration as code*. Our intuitive web-based user interface creates, starts, stops, and monitors end-to-end pipelines, which enhances AI Pro's usability to an unrivaled level. AI Pro expresses each data flow pipeline as a Directed Acyclic Graph (DAG) with data source, data sink (storage), models, and other computational units as *nodes or entities*. Directed edges connecting *entities* represent the nature of data flow between them. For experts, AI Pro offers pluggable custom features for data manipulation and modification.

Consider a scenario in which a software company has several servers to power their website and wants to perform realtime log anomaly detection. Open source log anomaly detection models like DeepLog and Loglizer [4], [5] can be used, however expertise is definitely required for their integration and monitoring. AI Pro takes care of such integration processes - connecting log data with AI models - with simple configuration steps, without the need for expert integration personnel. With proper configuration, output from the anomaly detection module can be sent to data stores for information display and further action.

Now consider another scenario involving classification and filtering capabilities. Credit card companies use fraud transaction detection tools to distinguish authentic transactions from fraudulent and harmful transactions. The data paths taken by authentic vs fraudulent transactions are different - authentic transactions pass through systems normally, but potentially fraudulent transactions require additional steps for verification. Authentic transactions can be further processed to determine

[†]These authors contributed equally to this work.

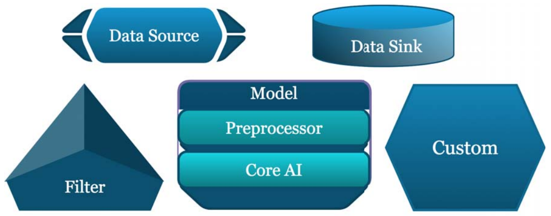


Figure 2: Different types of Entities in AI Pro.

the category of transactions to optimize customer experience and provide recommendation based on a user's orientation. AI Pro gives ample support to analyze several data paths in a single data pipeline.

AI Pro supports Open Neural Network Exchange (ONNX) [6], which allows its users to choose AI models from all different AI frameworks, such as Tensorflow, Caffe, Mxnet, Pytorch, Chainer etc. To the best of our knowledge, there is currently no open source data processing framework like AI Pro. To better understand the importance of AI Pro and our contribution, we briefly list our contributions as follows:

- AI Pro is an Open Source AI Processing Framework.
- It supports batch and stream processing.
- AI Pro uses *configuration as code*.
- Supports ONNX [6] and major AI frameworks.
- AI Pro uses the message queue communication paradigm for asynchronous, parallel processing.
- Web UI for pipeline management and monitoring.

We start our discussion with the system overview in Section II. In Section III, we present AI Pro's features in detail. We briefly provide two practical demo scenarios in Section IV to illustrate its potential. In Section V we present future work and conclusion.

II. System Overview

We have released the first version of AI Pro on Github[1]. In this section, we discuss the main components of AI Pro. To create a data pipeline, the end user creates the specifications of the pipeline in the web UI, which then generates a config file internally. The pipeline configuration processor then takes that file to generate the pipeline at deploy time. The trend of representing a data pipeline as a *directed acyclic graph* (DAG) is advantageous in many ways and is widely accepted by industry [7], [8], [9]. AI Pro's config file is a DAG of *nodes* or *entities* connected according to the flow of data. To remain consistent with notation, we will continue to refer to *node* as an *entity*. Data flow between entities is represented with *directed edges*.

Entity. We define entity as an abstract component responsible for either ingestion, transformation, addition, removal, or storage of *data elements* (e.g. JSON objects) in the process of data flow.

We now present some of the basic types of entities in AI Pro and briefly describe them as follows:

(a) ***Data source entity:*** AI Pro's data pipeline always starts with a *data source entity*. Out of the box, AI Pro provides support for various types of data sources, such as (i) files, (ii) streaming APIs, and (iii) NoSQL Databases. End users just need to specify the data file's location or provide API attributes depending on the type of data source and AI Pro will take care of the data flow.
Configuration sample for Data Source Entity:

```
{
  "alias" : "Twitter Streaming Source",
  "api_key" : "XXXXXX",
  "url": "http://example.com/api/data/",
}
```

(b) ***Model entity:*** A model entity is an AI model whose job is to make predictions based on some arbitrary input. Since it's very common for AI models to require input in a specific format to work, model entities encapsulate two sub-entities: preprocessor and core AI models.
  (i) *Preprocessor:* This entity selects a specific part of data as an input to the AI model, then performs any transformations necessary to prepare the data for the format required in the AI model. Some of these formats and transformations may include vector normalization, matrix manipulation, unit/metric conversion, or string list concatenation, as required by the parameters of model's *predict function*. The output of the preprocessor is then handed over to the core AI entity for processing.
  (ii) *Core AI:* A core AI entity is a pre-trained AI model ready to make predictions. These core AI models can either be traditional machine learning models or deep learning models built with Tensorflow, Keras, Pytorch, Caffe or any other frameworks supported by ONNX. Before starting any data pipeline, AI Pro initializes all core AI entities by installing required libraries and loading pre-trained models into memory. The end user just needs to provide the name of the *predict function* in the model to make it work.

We have provided some examples of the model entity in our repository e.g. tweet sentiment analysis, tweet classification models etc. We have also started an initiative to create a *Model Zoo* for AI Pro by encapsulating popular AI models (with prediction functions) and providing support for them in AI Pro.

Configuration sample for Model Entity:

```
{
  "alias" : "Sentiment model, custom",
  "input_attribute" : "text",
  "module_file_path" : "uploads/sentiment.py",
  "method_name" : "predict",
  "module_classname" : "SentimenModel",
  "preprocessor_filename" : "tweet_preprocessor.py",
  "preprocessor_methodname": "preprocess"
  "output_attribute" : "sentiment",
}
```

(c) ***Filter entity:*** A filter entity controls data flow in the DAG by evaluating data attribute values on criteria prespecified when the pipeline configuration is built. Depending on the output of a filter, it can be used to split the data flow into separate paths in the DAG, or even discard certain data elements that fail to meet predetermined criteria. This entity is particularly useful

[1]https://github.com/InitialDLab/AIPro

with the predicted output of classification models to channelize data elements to respective child entities for further processing.

Configuration sample for Filter Entity:

```
{
  "alias" : "Language filter",
  "attribute" : "lang",
  "value" : "en",
  "condition" : "=="
}
```

(d) ***Storage entity:*** A data pipeline can have multiple storage entities that store processed data at different locations. AI Pro currently supports three types of storage entities: (a) Regular file, (b) Database (c) Standard I/O. AI Pro supports many standard databases, such as MongoDB, PostgreSQL, and MySQL.

Configuration sample for storage entity:

```
{
"db" : "geotwitter",
"collection" : "tweets",
"alias" : "Tweets Mongo Connection",
"host" : "localhost",
"type" : "MongoDB",
"port" : 27017
}
```

(e) ***Custom entity:*** Experts can create custom entities for customized transformation of data elements. One example of such a custom entity that is included in AI Pro is geo-location mapping. It maps latitude and longitude to location name and country.

Edges AI Pro communicates between entities with an asynchronous message queue paradigm. It is highly available, scalable, and fault-tolerant. It supports more than one entity to be a consumer of a message queue, which makes it easier to duplicate data elements from one parent to multiple child entities.

The communication between entities is horizontally scalable when more resources are fed into it. It should also be noted that the same instance of a pipeline as well as entities within a single pipeline can run in parallel on different machines in order to scale data flow.

These system components are the integral parts of AI Pro. Each component is extensible, and the processing of one component is independent of another.

III. Feature Overview

In this section we describe some of the features of AI Pro that make it an approachable and user-oriented system. An overview of the system is presented in the project website [2] and a demo video [3].

Web User Interface (Web UI): AI Pro provides a user-friendly web interface for all of its operations, as shown in figure 1. The web interface enables non-experts to create their own pipelines and provides examples and tutorials to help them maintain and build their pipelines. For advanced users and core developers, there is a command line interface for in-depth operations with finer-grained control.

[2] https://www.cs.utah.edu/~deb/aipro
[3] https://youtu.be/e6imr87kdB4

Easy Pipeline Configuration: AI Pro's motto of *configuration as code* is fulfilled by a simple & intuitive pipeline configurator. As the user adds entities one by one to create a pipeline, AI Pro's user interface aids them to fill relevant attributes based on the entity type. It then generates a config file from at deploy time to start the pipeline.

Pipeline Management: Currently, AI Pro provides support to start, stop, modify, and delete a pipeline from the web interface. Users can use AI Pro to test different models just by swapping out different entities in the configuration. This feature minimizes turnaround time for developers to figure out appropriate models.

Status Monitoring: AI Pro makes it easy for the end user to see throughput and status of a pipeline. Monitoring the status of individual edges is also available, making it easier to debug the point of failure if something goes wrong.

Open Source and ONNX support: We support Free and Open-Source Software (FOSS) for the numerous benefits it brings to the software community. We believe in community driven software and hope AI Pro will attract collaborators from different domains to make it even more user-friendly. We will also continue to port AI models, especially the models that support ONNX, into *AI Pro's Model Zoo* for zero-hassle modularity. ONNX empowers the open AI ecosystem, and our goal is to align with it.

To adhere to the limitation of space in this demo paper, we restrict our discussion of features here.

IV. Demo Scenarios

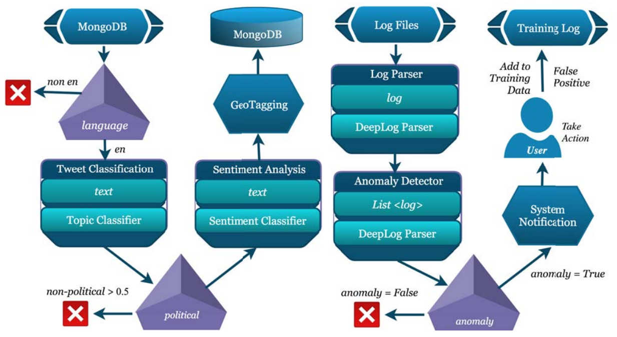
In this section we present two demo scenarios of AI Pro to illustrate how it simplifies data processing while working with AI. The first demo scenario is *Spatio-temporal Sentiment Analysis on the 2016 US Presidential Election with Tweets*. In the second demo scenario, we present a system of *System Log Anomaly Detection*.

A. Spatio-Temporal Sentiment Analysis with Tweets:

This demo scenario shows how the AI Pro framework is used to predict voting patterns over time for the 2016 US Presidential Election by classifying political tweets and then predicting the sentiment of tweets that are classified as political. From there, tweets are supplemented with state and county names using the geotagging entity so that voter predictions can be visualized in aggregate by state and county. A pictorial DAG representation of this scenario is presented in Figure 3a.

Entity Descriptions:

1) *Data source*: The pipeline starts with a MongoDB Database as the *data source*. In the configuration file, we list the required attributes to connect to our database in MongoDB and added an optional attribute *projection* to collect only the text, language, date, and geolocation from each individual tweet.
2) *Political filter*: Data then flows to a *filter entity* to only include tweets that were written in English.



(a) Sentiment Analysis. (b) Log Anomaly Detection.

Figure 3: System Architecture of Demo Scenarios.

3) *Tweet classification model*: The tweet classification model classifies the text into one of three categories: democratic, republican, or non-political. Each data element is then passed to the next filter in the pipeline.
4) *Filter entity:* Based on the output of the tweet classification model, this filter determines which data elements pass to the next entity and which are dropped. We only keep tweets that are classified as democratic or republican, and pass them to the sentiment analysis model.
5) *Sentiment analysis model entity*: Tweets that passed the above filtering conditions are then fed into a pre-trained sentiment analysis model. This model predicts how strongly positive or negative the emotion is in the tweet, on a scale of 0 to 1, where 0 is negative and 1 is positive. We also keep track of whether the tweet is democratic or republican, and use that to show sentiment towards each political party in order to approximate future potential voting trends.
6) *Geotagging custom entity*: All data elements are then passed through a geotagging entity, where it maps a tweet to a county, state and country based on its latitude and longitude attributes. It also appends those attributes to each data element.
7) *Data storage entity*: Enriched data elements with information from AI models and custom entities are now stored in a MongoDB with this entity.

Enriched data, stored in MongoDB, is now ready for analytics. We used Spatial In-Memory Big-data Analytics (Simba) [10] for online analytics and presented it with an interactive web interface [4].

B. System Log Anomaly Detection:

Log anomaly detection is an essential step towards building a secure and reliable system. This demo is an interesting scenario where the log anomaly detection model continuously learns as we discover false positives from the system and use those to retrain the model. [11]. A schematic representation of the system architecture is presented in Figure 3b.

Entity Descriptions:

1) *Data source entity*: The log entries from a log file are fed into the system through this entity. The data elements contain timestamp and log as attributes.

[4]`http://estorm.org`

2) *Log parser model*: The preprocessor inputs log text to the log parser, which outputs the log key and a list of parameter values. The data elements attach these attributes to pass it to the child entity.
3) *Anomaly detection model*: The anomaly detection model takes input as a batch of keys and a corresponding list of parameter values. The preprocessor creates chronologically sorted batches of data elements to feed into the anomaly detection model. If the model detects any anomaly, its output is true, otherwise the output is false. This data is then saved under the *anomaly* attribute.
4) *Anomaly filter*: If the attribute value of *anomaly* is true, a notification is sent to an end user with the anomalous log messages.
5) *Training data source*: When end users find the anomaly detected to be a false positive, end users write it to a training data file. The training data source then reads false positives entries and feeds them to the anomaly detection model in training mode instead of predict mode.

These interesting scenarios show the power of AI Pro. AI Pro can also be used for scenarios like realtime object detection in the surrounding environment, with a speech synthesis module to aid the hearing and vision impaired.

V. Conclusion

We presented AI Pro, an open source AI data processing framework for the community. With the demo scenarios we have presented, we believe that AI Pro has the potential to help the industry flourish with the many benefits of AI. In future, we will continue to make it even more user friendly and port as many AI models to *AI Pro's Model Zoo* as possible for plug and play potential.

References

[1] S. Miller and D. Hughes, "The quant crunch: How the demand for data science skills is disrupting the job market," *Burning Glass Technologies*, 2017.
[2] InsideBigdata, "Infographic: The data scientist shortage - insidebigdata," accessed 8 Nov 2018. [Online]. Available: https://insidebigdata.com/2018/08/19/infographic-data-scientist-shortage/
[3] HBR, "3 things ai can already do for your company," accessed 10 Nov 2018. [Online]. Available: https://hbr.org/2018/01/artificial-intelligence-for-the-real-world
[4] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: system log analysis for anomaly detection," in *Software Reliability Engineering (ISSRE) 2016*. IEEE, 2016.
[5] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection." in *USENIX Annual Technical Conference*, 2010.
[6] onnx.ai, "Open neural network exchange (onnx)," Sep 2017. [Online]. Available: https://onnx.ai
[7] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "Knime-the konstanz information miner: version 2.0 and beyond," *SIGKDD explorations Newsletter*, 2009.
[8] T. Kosar and M. Livny, "Stork: Making data placement a first class citizen in the grid," in *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*. IEEE, 2004.
[9] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy, and C. Curino, "Apache tez: A unifying framework for modeling and building data processing applications," in *SIGMOD 2015*. ACM, 2015.
[10] D. Xie, F. Li, B. Yao, G. Li, L. Zhou, and M. Guo, "Simba: Efficient in-memory spatial analytics," in *SIGMOD 2016*. ACM, 2016.
[11] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *SIGSAC 2017*. ACM, 2017.