Debuggable Machine Learning with ConX and Comet.ml

Douglas Blank and Cecelia Shao (Comet.ml)
Presented at ICLR 2019 Debugging Machine Learning Models Workshop

Machine Learning (ML) models can take a variety of forms including Evolutionary Systems, Bayisian Networks, Reinforcement Learning, Deep Learning (DL), and many others. Certainly, these programs can have the standard types of computing errors and require standard types of debugging tools and skills. However, ML models can have their own unique types of semantic and runtime errors as well. Some of these "ML runtime errors" are similar to standard errors. However some "errors" could simply be differences between what the modeler intended and what they instantiated, or be caused by unexpected dataset formats and data distributions. In this presentation, we demonstrate tools to show how to make debuggable ML. Specifically, we demonstrate an open source DL package for designing and building DL models, called ConX [1], and a commercial ML product built for logging, visualizing, and managing ML experiments, called Comet.ml [2].

Most ML modeling tools are designed to run as a script rather than be used interactively. Furthermore, being able to interactive inspect intermediate computations (such as hidden layer activations in a DL model) often requires detailed knowledge of the model's internals. When a DL model is constructed in the Python open source neural network library ConX, it automatically builds all of the intermediate models between the input and output layers. This allows an interactive inspection at each step in the processing, similar to a "stepper" in a traditional debugger. The modeler can visually examine the activations (as a vector or image) at each step through the network, or display an entire network's activations and weights provided by a dataset abstraction layer. ConX is designed to work interactively in a Jupyter Notebook [3]. Many tensor- or matrix-based ML tools are overly flexible. For example, many frameworks allow any shape of tensor to be used as input and target, or generally allow any for any error function to be used to compute loss of an output layer with generally any activation function. However, this often does not make any sense. As such, ConX gives appropriate warnings when a user is doing something that appears to be wrong.

Deterministic training with proper version control can help make this training process auditable and dramatically improve the experience of debugging, especially in collaborative settings where practitioners working on different versions of a model may make several changes to a project’s files [4]. In order to reproduce the condition that caused model error, it is necessary to keep track of data transformations and splits, model architecture changes, hyperparameter configurations, and other critical experiment artifacts [5]. Comet.ml enables the automation of large portions of the experiment environment, so that the documentation of system metrics, software dependencies, source code, and hyperparameters occurs simultaneously with the training process. With the experiment data in Comet.ml as a foundational, central ‘source of truth’, users can leverage visualizations of their experiment’s performance to see their model is converging on an optimal solution and compare the source code and environment details between experiments to identify potential errors and optimizations.