Towards a Human-in-the-Loop Library for Tracking Hyperparameter Tuning in Deep Learning Development

Renan Souza[1,2], Liliane Neves[1], Leonardo Azeredo[1], Ricardo Luiz[1], Elaine Tady[1], Paulo Cavalin[2], Marta Mattoso[1]

[1]COPPE/Federal University of Rio de Janeiro
[2]IBM Research

Abstract. *The development lifecycle of Deep Learning (DL) models requires humans (the model trainers) to analyze and steer the training evolution. They analyze intermediate data, fine-tune hyperparameters, and stop when a resulting model is satisfying. The problem is that existing solutions for DL do not track the trainer actions. There are no explicit data relationship between trainer action with the input data and hyperparameters to the output performance results, throughout the training process. This jeopardizes online training data analyses and post-hoc results reproducibility, reusability, and understanding. This paper presents DL-Steer, our first prototype to aid trainers to fine-tune hyperparameters and for tracking trainer steering actions. Tracked data are stored in a relational database for online and post-hoc data analyses.*

1. Introduction

Training Deep Learning (DL) models is an iterative process that can last for weeks. The training can be seen as a workflow with chaining activities like loading input data, combining hyperparameter variations to train several models, tuning hyperparameters for model fitting, and evaluating resulting models. Based on hyperparameters (*e.g.*, number of layers and neurons, loss functions, dropout rates, learning rate, number of epochs, etc.) and model performance, the training workflow iterates until performance is good enough. Users (DL trainers) need to steer the workflow, for example by incrementally fine-tuning the model hyperparameters. User steering decisions are based on the intermediate data analysis and the model performance. Users often work in a time-consuming trial-and-error manner following the DL training evolution via monitoring tools that plot model performance results. Basic monitoring is limited for user steering. There should be interactive tools to put the DL trainers active in the training execution loop ("Human-in-the-Loop") [Kumar et al. 2016].

There are some "Human-in-the-Loop" approaches for DL trainers, but we did not find any work that tracks user steering actions. In a survey, Kumar *et al.* define an architecture for Model Selection Management Systems (MSMS) to iterate on three activities: feature engineering, algorithm selection, and parameter tuning [Kumar et al. 2016]. They propose declarative interfaces for the users to interact with MSMS, and consider provenance data management an import direction and a challenge to represent which and how training data should be captured. Xin *et al.* discuss user steering challenges, such as intermediate results reuse, analysis of the impact of changes, automated recommendations for the user, and reduced feedback between user action and its effect [Xin et al. 2018]. They propose HELIX as a declarative system aimed at addressing the challenge for intermediate results reuse. Miao *et al.* also highlight several

research challenges addressing data management in deep learning, such as understanding and comparing models, model tuning, model versioning, and parameter archiving. They propose ModelHub to address some of these challenges, particularly it has a versioning system and a declarative language to enable users to fine-tune hyperparameters [Miao et al. 2017]. These works highlight that resulting data analysis jointly with user fine-tunings is a challenge. None of them keeps the track of user-steered actions for online analysis with an explicit data relationship with their results, intermediate data and input hyperparameters throughout the whole training process. Keeping these data available for online and *post-hoc* data analyses is important for training as well as for *post-hoc* results reproducibility, reusability, and understanding of the user steering actions.

We present a first prototype of DL-Steer, a library that has three main features. First, it collects input values for deep learning model hyperparameters and relates them to the trained model performance output results (*e.g.*, accuracy and others). These input and output values can be monitored and queried by the user during the training. Second, based on this online data analysis, DL-Steer enables users to steer the training by fine-tuning hyperparameter values. Third, during the training, it captures the user steering actions and correlates them with intermediate data, input hyperparameters, and resulting model metadata and stores them in a relational DBMS, as they are being generated. These three features allow for joint analysis of steering actions, input and output; and facilitates understanding the impacts of specific user-steered model fine-tunes on the results during and after the training. All these data are stored and related in a provenance database that extends W3C PROV concepts. In our case study, DL-Steer supports users to fine-tune the model and evaluate the impact of steering actions during the training. Queries show the impact on the trained models with the tunings of specific DL hyperparameters.

2. DL-Steer: a steering library for hyperparameter tuning in DL

DL-Steer follows the concepts of Computational Steering, which have been under investigation by multidisciplinary research fields for years [Bauer et al. 2016, Mulder et al. 1999]. The two fundamental concepts of Computational Steering are monitoring of evolution of a computational process; and user-steered dynamic adaptations of the computational process. In addition to these two concepts, in past works we have shown the importance of provenance data management for Computational Steering in: online data analyses to aid users in understanding the intermediate data being generated to help on dynamic adaptation decisions; and for keeping track with online evaluation of user steering actions [Souza et al. 2017, Souza and Mattoso 2018]. DL-Steer builds on these directions to support DL training workflows, particularly for fine-tuning hyperparameters during model training. We introduce DL-Steer through three main steer steps, as follows.

(i) Instrumentation of scripts. In this first prototype, we aim at users that write Python scripts with function calls to TensorFlow [Abadi et al. 2016], a highly parallel and widely used library for machine learning, particularly for DL. Python scripts with TensorFlow are largely found in open source code repositories. Initially, the users insert function calls to DL-Steer library in strategic points of their script. Inserting calls in scripts is the main approach in Computational Steering [Bauer et al. 2016], because users do not want to wrap they code in a new specific system for steering. Users add function calls to DL-Steer at strategic points of the script. In GitHub (https://github.com/hpcdb/DL-Steer), we show an exemplary pseudocode instrumented using DL-Steer library. The strategic points serve for data capture and steering.

(ii) Data capture and steerable points. *Data capture points* identify where to capture the intermediate training results during the workflow execution. DL-Steer captures the training data and correlates them to input hyperparameter values and models results, and inserts all of them in a provenance database online during the training. The provenance database is managed by MonetDB, a relational column-oriented DBMS, efficient for data analytics. *Data steerable points* determine where the user can dynamically fine-tune hyperparameter values that will be used for training. They are implemented as a queue data structure. The queue interfaces to Redis, a Key/Value lightweight in-memory store. The queue data managed in Redis are accessible both by the running script and by external steering actions. Each item in the queue contains a combination of hyperparameter variations that is iteratively evaluated *(while there are items in the queue to evaluate)* as a model fit. During the training, the user can modify how hyperparameters should vary inside an item. To prevent inconsistencies, DL-Steer guarantees that only the items that have not been evaluated yet can be modified by the user. When the user decides to modify the queue, the steering action takes place only in the next iteration. Provenance data of the steering action are captured, related to current training data being generated, and stored in the provenance database in MonetDB. The DL-Steer data schema follows W3C PROV provenance data model specializations, such as our previous works [Souza et al. 2017, Souza and Mattoso 2018]. Online data analyses over the training data are driven by provenance-based queries.

(iii) Data steering during training. During the DL training workflow execution, the user can either run *ad-hoc* analytical queries on MonetDB directly (via its SQL query interface) or use a graphical interface that queries it at specified time intervals and plots evolution of training results. DL-Steer has a graphical interface and a command line interface for steering actions like modifying queue items. Hyperparameters in each item are represented as keys and values, where the keys are hyperparameters to be evaluated in the training and the values represent how they vary. In case of numeric lists of variations, the user specifies the initial and final interval values and the incremental step. For example, the key/value structure `{"learning_rate": [0.1, 1.0, 0.1], "epochs": [10, 50, 10]}`, sets the values of the hyperparameters learning rate and number of epochs.

3. Case Study

In this preliminary DL-Steer case study, the user identifies data capture and steerable points to instrument a Python script that uses TensorFlow for parallel Deep Learning training of an image classifier in GPU. The data capture points are given by a queue of hyperparameter variations with initial values defined by the user and by the trained model performance results. The performance function is defined by the user as a "Gain" function. The steerable point is the queue of hyperparameter variations. At each iteration, an item in this queue is processed in a model fitting and DL-Steer captures the input values used and model output results generated. Moreover, at each iteration, DL-Steer's dashboard queries the provenance database to plot performance results, as shown in Figure 1. In this case study, at iteration 20, the user observes that the Gain function moving average (*i.e.*, the average of the last *k* iterations) is 0.28 and the hyperparameter learning rate is varying in the interval `[0.01, 0.05]` with step 0.01. Then, at iteration 21, the user does the first steering action to modify the learning rate interval to `[0.03, 0.07]`, with the same step 0.01. This action impacts the gain, as observed in the moving average at 10 iterations later: 0.48 (an improvement of 0.2). At the iteration 54, the moving average is 0.51 and the hyperparameter batch size is 100. Then, the user decides for

another action at iteration 55 to modify the batch size to 1000. This also impacts the gain, as the moving average 10 iterations later increased to 0.55 (improvement of 0.04). Finally, after 75 iterations, the user observes that the iteration 67 generated a model with a satisfying gain function value of 0.84. The user then decides to stop the training. Input hyperparameters and performance model results of all models, including the best model, and the related user steering data are all stored in the DL-Steer's provenance database.

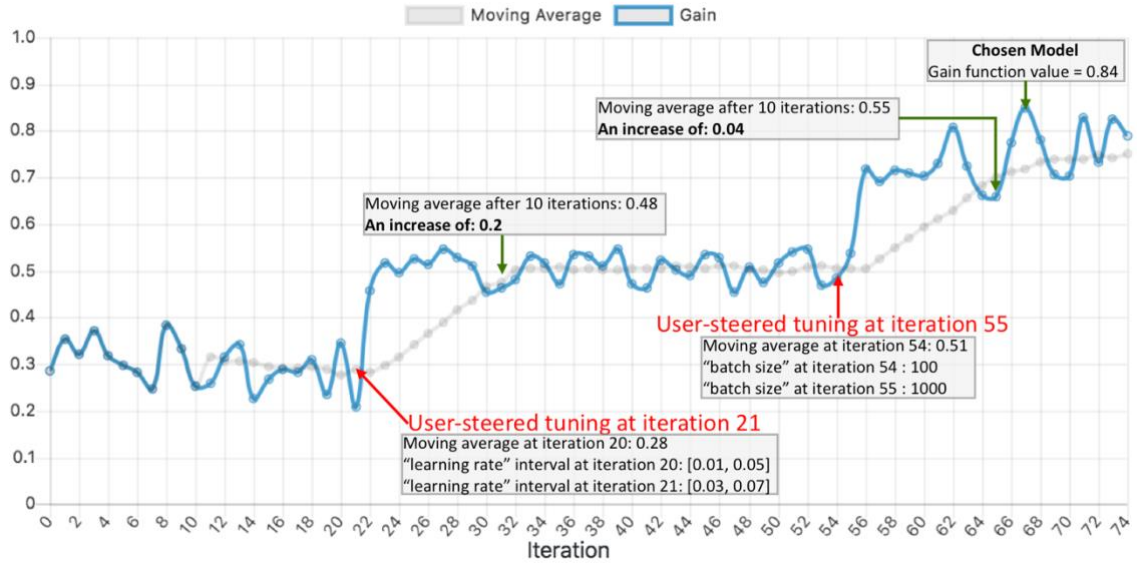

Figure 1. DL training results with user steering visualization.

4. Concluding Remarks

We presented DL-Steer, our first prototype of a library for tracking user-steered hyperparameter tuning during the DL training workflow. It keeps track of steering actions in the models with an explicit data relationship with their results, intermediate data, and input hyperparameters, throughout the training process, all related in a provenance database. This not only contributes for experiment reproducibility, but also helps the user to understand the impact of steering actions in the results. For future work, we plan to explore our solution to analyze variations of different network architectures, loss functions, dropout rates, and other hyperparameters in a large-scale dataset.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., et al., (2016), "TensorFlow: a system for large-scale machine learning". In: *OSDI*, p. 265–283.

Bauer, A. C., H., A., Ahrens J., Childs H., Geveci B., Klasky S., Moreland K., O'Leary P., Vishwanath V., et al., (2016), "In situ methods, infrastructures, and applications on high performance computing platforms", *C. Graphics Forum*, v. 35, n. 3, p. 577–597.

Kumar, A., McCann, R., Naughton, J., Patel, J. M., (2016), "Model selection management systems: the next frontier of advanced analytics", *SIGMOD Record*, v. 44, n. 4, p. 17–22.

Miao, H., Li, A., Davis, L. S., Deshpande, A., (2017), "Towards unified data and lifecycle management for deep learning". In: *ICDE*, p. 571–582.

Mulder, J. D., van Wijk, J. J., van Liere, R., (1999), "A survey of computational steering environments", *FGCS*, v. 15, n. 1, p. 119–129.

Souza, R., Mattoso, M., (2018), "Provenance of dynamic adaptations in user-steered dataflows". In: *IPAW*, p. 1–12.

Souza, R., Silva, V., Coutinho, A. L. G. A., Valduriez, P., Mattoso, M., (2017), "Data reduction in scientific workflows using provenance monitoring and user steering", *FGCS*, v. online, p. 1–34.

Xin, D., Ma, L., Liu, J., Macke, S., Song, S., Parameswaran, A., (2018), "Accelerating human-in-the-loop machine learning: challenges and opportunities". In: *SIGMOD Workshop on Data Management for End-to-End Machine Learning*.