ModelHub: Deep Learning Lifecycle Management

Hui Miao, Ang Li, Larry S. Davis, Amol Deshpande
Department of Computer Science, University of Maryland, College Park, MD 20742
{hui, angli, lsd, amol}@cs.umd.edu

***Abstract*—Deep learning has improved the state-of-the-art results in many domains, leading to the development of several systems for facilitating deep learning. Current systems, however, mainly focus on model building and training phases, while the issues of lifecycle management are largely ignored. Deep learning modeling lifecycle contains a rich set of artifacts and frequently conducted tasks; dealing with them is cumbersome and left to the users. To address these issues in a comprehensive manner, we demonstrate ModelHub, which includes a novel model versioning system (`dlv`); a domain-specific language for searching through model space (`DQL`); and a hosted service (ModelHub).**

Video: https://youtu.be/4JVehm5Ohg4

I. Motivation and Background

Deep neural networks (DNN) have dramatically improved the state-of-the-art results for many important fields in recent years [1]. Learned using massive amounts of training data, DNN models have superior generalization capabilities, and the intermediate layers in many deep learning models have been proven useful in providing effective semantic features that can be used with other learning techniques. However, there are many critical large-scale data management issues in learning, storing, sharing, and using deep learning models, which are largely ignored by researchers today, but are coming to the forefront with the increased use of deep learning in a variety of domains. Deep learning modeling lifecycle contains a rich set of artifacts, e.g., learned parameters and training logs, and frequently conducted tasks, e.g., to understand the model behaviors and to try out new models. Dealing with such artifacts and tasks is cumbersome and left to the users. To address these issues in a comprehensive manner, we propose and demonstrate ModelHub, which includes a novel model versioning system (`dlv`); a provenance management system; a domain-specific language for searching through model space (`DQL`); and a hosted service (ModelHub) to store learned models, explore existing models, and share models.

DNN Modeling Lifecycle and Challenges: Compared with the traditional approach of feature engineering followed by model learning [2], deep learning is an end-to-end learning approach, i.e., the features are not given by a human but are learned in an automated fashion from the input data. Moreover, the features are complex and have a hierarchy along with the network representation. This requires less domain expertise and experience from the modeler, but understanding and explaining the learned models is difficult. Thus, when developing new models, changing the learned model (its network structure and hyper-parameters) becomes an empirical search task.

This work was supported in part by NSF under grants 1513972 and 1513443, and in part by the Office of Naval Research under grant N000141612713 entitled 'Visual Common Sense Reasoning for Multi-agent Activity Prediction and Recognition.'

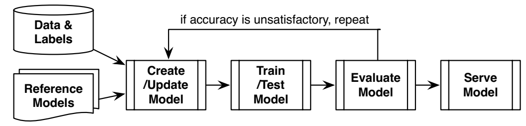

Fig. 1: Deep Learning Modeling Lifecycle

In Fig. 1, we show a typical deep learning modeling lifecycle. Given a prediction task, a modeler often starts from well-known models that have been successful in similar task domains; she then specifies input training data and output loss functions, and repeatedly adjusts the DNN on operators and connections like Lego bricks, tunes model hyper-parameters, trains and evaluates the model, and repeats this loop until prediction accuracy does not improve. Due to a lack of understanding about why models work, the adjustments and tuning inside the loop are driven by heuristics, e.g., adjusting hyper-parameters that appear to have a significant impact on the learned weights, applying novel layers or tricks seen in recent empirical studies, and so on. Thus, many similar models are trained and compared, and a series of model variants need to be explored and developed. Due to the expensive learning/training phase, each iteration of the modeling loop takes a long period of time and produces many (checkpointed) snapshots of the model. The modeling lifecycle exposes several systems and data management challenges, including: a) managing many similar models and artifacts, and their versions, b) large storage footprints of learned parameters, c) understanding and comparing models, d) abstracting repetitive steps in adjusting models, and e) sharing models with others.

Related Work & Contributions: There have been several high-profile deep learning systems in recent years, but those typically focus on training aspects (e.g., distributed training, utilizing GPUs, etc.) [3], [4]. The data management and lifecycle management challenges discussed above have been largely ignored so far, but are becoming critical as the use of deep learning permeates through a variety of application domains, since those pose a high barrier to entry for many potential users. In the database community, there has been increasing work on developing general-purpose systems for supporting machine learning, including pushing predictive models into databases [5], accelerating learning in databases [2], [6], and managing modeling lifecycles and serving predictive models in advanced ways [7], [8], [9]. ModelHub is motivated by similar principles; aside from a specific focus on deep learning models, ModelHub also supports *versioning* as a first-class construct [10] which differentiates it from that work.

II. ModelHub Overview

Here we briefly describe the key functionality of ModelHub, and refer to [11] for full technical details. ModelHub consists of five key components (Fig. 2) that we discuss next.

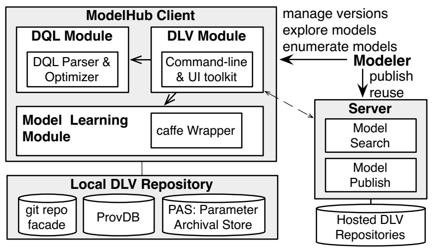

Fig. 2: ModelHub System Architecture

(1) `DLV` is a novel versioning system (VCS), implemented as a command-line tool (`dlv`), that serves as an interface to interact with the rest of the components. Use of a specialized VCS instead of a general-purpose one like `git` allows us to better portray and query the internal structure of the artifacts generated in a modeling lifecycle, e.g., network definitions, training logs, binary weights, and relationships between models. When managing DNN models in the VCS repository, a *model version* consists of a network definition, a collection of weight parameters, extracted metadata (e.g., hyper-parameters, accuracy and loss information during the training phase), and a collections of files used together with the model instance (e.g., scripts, datasets). A set of *model version*s and the lineage of the *model versions* are captured when the user runs the different `dlv` commands that update the repository. `dlv` features a novel parameter storage (*PAS*) that archives model versions using *deltas* in a multi-resolution fashion, and an approximate progressive query/inference evaluation algorithm [11].

(2) `DQL` is a model enumeration and hyper-parameter tuning domain-specific language (`DQL`), that we propose to serve as an abstraction layer, to help modelers focus on the creation of the models instead of repetitive steps in the lifecycle. The query facilities can be categorized into two types: a) model exploration queries and b) model enumeration queries. **Model exploration queries** interact with the models in a repository, and the users use them to understand a particular model, to query lineages of the models, and to compare several models. For usability, similar to other VCS, we design it as query templates via `dlv` sub-command with options and render results in HTML front end when needed; examples include `dlv list` to find models and lineages, `dlv desc` to understand models via metadata and plots, `dlv diff` to compare models side by side, and `dlv eval` to evaluate managed models. **Model enumeration queries** are used to explore variations of currently available models in a repository by changing network structures or tuning hyper-parameters. `DQL` supports several operations that need to be done in order to derive new models: 1) Select models from the repository to improve; 2) Slice particular models to get reusable components; 3) Construct new models by mutating the existing ones; 4) Try the new models on different hyper-parameters and pick good ones to save and work with. When enumerating models, `dlv` also allows stopping exploration of bad models early.

(3) The *model learning module* interacts with external deep learning tools that the modeler uses for training and testing. In the demonstration, we implement a concrete model learning module on top of `caffe`, which is a popular deep learning training system for computer vision modelers [3].

(4) `ProvDB` is a standalone provenance management module used by `dlv` to store provenance metadata and other information discussed above. `ProvDB` is built on top of Neo4j, and provides several visual interfaces to explore the stored information. In particular, it allows a user to easily and visually compare different models along several different dimensions.

(5) Finally, the ModelHub service is a hosted toolkit to support publishing, discovering and reusing models, and serves similar role for DNN models as *github* for software development.

III. Demonstration Plan

In the demonstration, we introduce the current version of our ModelHub system and illustrate how it manages the modeling lifecycle and DNN models. We prepare a collection of development models for face recognition. The goal is to train a DNN on a large face dataset (CASIA-WebFace), which contains 494,414 face images from 10,575 subjects. The DNN models have been created over a period of time, and were not managed in ModelHub by default. Along with the models, the experiment spreadsheets and setup scripts are shown as they are. The conference attendees can use the ModelHub system to walk through the DNN lifecycle via real models. Interacting with `dlv`, the users are shown the full list of query facilities it currently supports. We use `dlv` to demonstrate how to manage models using `dlv add` and `dlv commit` and the benefit of metadata management with versioning ability. Furthermore, the managed models can be explored and compared using `dlv desc` and `dlv diff` in an interactive HTML-based GUI. After using `dlv` to manage and understand the models, our demonstration includes enumerating new models using `DQL`, and searching through network architectures and hyper-parameter values. The conference attendees can also interact with the system through `ProvDB`, and can use it to explore the provenance and visually compare derivations of different models. Finally, we show how to publish and download models from ModelHub. Modified face recognition models are published via `dlv publish`. Besides the face dataset, we also prepare well known models and host them beforehand in a ModelHub instance. The attendees are allowed to discover and download models via `dlv search` and `pull`.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, 2015.

[2] C. Zhang, A. Kumar, and C. Ré, "Materialization optimizations for feature selection workloads," in *SIGMOD*, 2014.

[3] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *ACM MM*, 2014.

[4] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems," in *OSDI*, 2016.

[5] X. Feng, A. Kumar, B. Recht, and C. Ré, "Towards a unified architecture for in-RDBMS analytics," in *SIGMOD*, 2012.

[6] A. Kumar, J. Naughton, and J. M. Patel, "Learning generalized linear models over normalized data," in *SIGMOD*, 2015.

[7] D. Crankshaw *et al.*, "The missing piece in complex analytics: Low latency, scalable model management and serving with Velox," in *CIDR*, 2015.

[8] A. Kumar *et al.*, "Model selection management systems: The next frontier of advanced analytics," *SIGMOD Record*, 2015.

[9] M. Vartak *et al.*, "ModelDB: a system for machine learning model management," in *SIGMOD HILDA Workshop*, 2016.

[10] A. Bhardwaj *et al.*, "Datahub: Collaborative data science & dataset version management at scale," in *CIDR*, 2015.

[11] H. Miao, A. Li, L. Davis, and A. Deshpande, "Towards unified data and lifecycle management for deep learning," in *ICDE*, 2017.