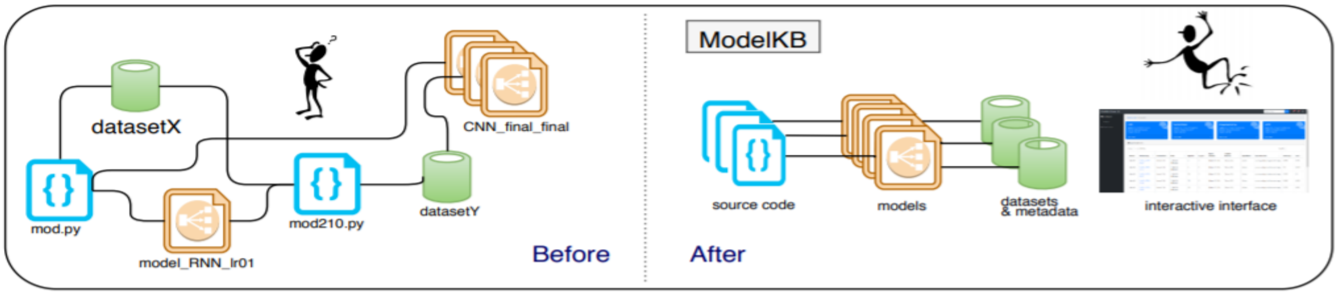
Automated Management of Deep Learning Experiments

Gharib Gharibi, Vijay Walunj, Rakan Alanazi, Sirisha Rella, Yugyung Lee
ggk89@mail.umkc.edu
University of Missouri-Kansas City
Kansas City, MO



ABSTRACT

Developing a deep learning model is an iterative, experimental process that produces tens to hundreds of models before arriving at a satisfactory result. While there has been a surge in the number of software tools that aim to facilitate deep learning, the process of managing the models and their artifacts is still surprisingly challenging and time-consuming. Existing model-management solutions are either tailored for commercial platforms or require significant code changes. In this paper, we introduce a lightweight system, named ModelKB, that can automatically extract and manage the model's metadata and provenance information (e.g., the used datasets and hyperparameters). Our overarching goal is to automate the management of deep learning experiments with minimal user intervention. Moreover, ModelKB provides a stepping stone to facilitate model selection and reproducibility.

KEYWORDS: Data management, deep learning, software automation

DEEM'30, June 30, 2019, Amsterdam, Netherlands
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6797-4/19/06…$15.00
https://doi.org/10.1145/3329486.3329495

ACM Reference Format:
Gharib Gharibi, Vijay Walunj, Rakan Alanazi, Sirisha Rella, Yugyung Lee. 2019. Automated Management of Deep Learning Experiments. In *International Workshop on Data Management for End-to-End Machine Learning (DEEM'30), June 30, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3329486.3329495

1 INTRODUCTION

Background: Deep learning [8] has improved the state-of-the-art results in an ever-growing number of domains such as computer vision, speech recognition, and business analytics. A *deep learning model*, also known as *deep neural network* (DNN), can be defined as a mapping function that maps raw input data (e.g., images) to the desired output (e.g., classification labels) by optimizing a specific metric (e.g., minimizing a loss function) typically using an iterative approach, such as stochastic gradient descent.

The key aspect of deep learning is that it can automatically learn the features of the input data and map them to the desired output using large amounts of data without human intervention (e.g., feature engineering). Given a prediction task, the practitioner (e.g., data scientist) starts by specifying a dataset, the target output, and a proper optimization metric. Then, she builds a DNN architecture, either from scratch or reusing an existing DNN, and then repeatedly adjusts the architecture and its hyperparameters (e.g., number of layers, learning rate) until reaching a satisfactory result. Due to limited knowledge on how DNNs work, adjusting the hyperparameters is done in a heuristic trial-and-error method resulting in hundreds of training iterations. Each iteration (*experiment*) results in a large set of artifacts, such as the architecture, weights, and snapshots of the training implementation (i.e., source code). These artifacts represent a rich set of data about the training experiments that can be used to analyze, explore, and derive insights (e.g., *what hyperparameters work best with this kind of datasets*).

Deep Learning Modeling Challenges: In addition to the tradition software development challenges, the large number of iterative experiments introduce a new set of challenges: managing the modeling lifecycle (i.e., tracking, storing, query, comparing, and reproducing experiments) [7, 10, 13, 15, 18]. The interviews conducted in both [19] and [20] revealed that the majority of the interviewed data scientists still use *manual* methods to manage their experiments such as text files, spreadsheets, and folder hierarchies, which are expensive, time-consuming and error-prone. Manually managing the modeling experiments hinder other critical deep learning tasks including model comparison, sharing, and reproducibility.

While there have been some recent attempts to address model management in both academia and industry, such as ModelHub [9], ModelDB [21], Runway [19], MLflow [22] and others [11, 16, 17], most of these approaches either require a considerable amount of code changes to instrument the tracking process or are limited to a specific commercial platform.

Contributions: To this end, we present a prototype system, named ModelKB (Model Knowledge Base), to automate the management of deep learning experiments. In particular, ModelKB can automatically extract, store, and manage the metadata and provenance information of the modeling experiments. Our overarching goal is to automate the management process with minimal user intervention within the user's favorite deep learning platform. This allows data scientists to focus on the modeling process without having to worry about managing their experiments or changing their favorite platforms and IDEs.

To summarize, our main contribution is a deep learning management system–ModelKB which can (1) automatically extract and store the model's metadata and experiments' artifacts; (2) visualize, query, and compare the modeling experiments; and (3) reproduce experiments when needed. ModelKB *will* also include a repository that facilitates sharing models remotely (a future work). It is important to note that ModelKB is not a modeling platform rather a complementary system that can automatically manage the experiments in their native platforms.

2 SYSTEM OVERVIEW

2.1 Architecture

Figure 1 illustrates an overview of ModelKB's architecture that consists of five main components: Metadata Extractor, Local Repository, Code Generator, Graphical User Interface, and a set of REST-APIs to invoke ModelKB remotely. These components are managed using the ModelKB Engine. ModelKB organizes the extracted metadata in a hierarchy of Projects and Experiments. A project represents the overall application at hand (e.g., object detection) and it is made of multiple experiments. An Experiment represents a single training run and its artifacts.

ModelKB was developed to enable automatic management of the experiments with minimal user intervention. The existing management approaches mostly require the user to instrument the implementation using several logging functions. In contrast, ModelKB uses Abstract Syntax Trees [14] to automatically locate and extract the metadata and provenance information, which does not require running the experiment (e.g., used dataset, hyperparameters). In order to extract the metadata and artifacts that are generated during and after the training, ModelKB provides wrapper functions that extend the core functions of the modeling platform. Currently, we support TensorFlow [1] and Keras [2]. For example, in the case of Keras, we extended the functions of `fit` (fits a model to a dataset), `fit_generator` (fits the data using a generator), and `predict` (makes predictions using a trained model). Thus, to automatically manage the experiments in Keras, the developer needs to (1) import ModelKB to their workspace and (2) use our extended functions instead of the native functions (i.e., use `xfit` instead of `fit` and `xpredict` instead of `predict`).

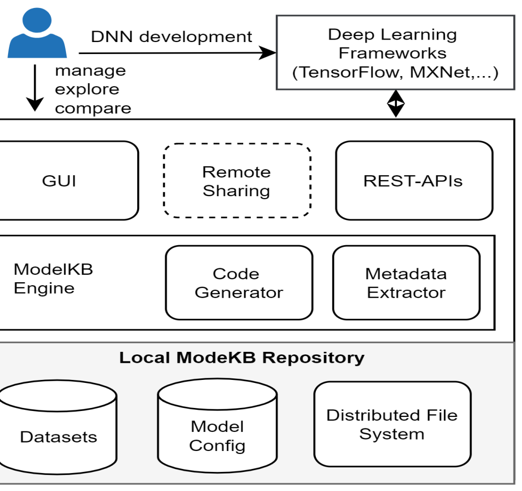


Figure 1: ModelKB Architecture Overview

2.2 Automatic Metadata Extraction

The Metadata Extractor is responsible for automatically extracting the modeling metadata from the source code and passing these metadata to ModelKB Engine, which will store and visualize the extracted metadata. The extractor extracts two types of metadata: First, it uses the AST to extract the metadata that does not require running the experiment, and this does not require any user intervention. Second, it provides function wrappers to extract training and evaluation metadata, and this requires simple code instrumenting (e.g., changing the function name from `fit` to `xfit` in Keras).

Inspired by the work in [16], our data schema can be summarized in four main entities: Model-Metadata, Experiment-Metadata, Prediction-Metadata, and Environment-Metadata. The Model-Metadata captures (1) a link to the dataset entity, (2) a link to the implementation entity, and (3) the model's configuration metadata. We also generate unique identification information for each model (e.g., identifier, timestamp). The Experiment-Metadata captures data about each experiment run (e.g., the loss and accuracy scores over time) in addition to other data that can be added manually by the user. We also evaluate each run and include additional metrics to each experiment, such as the confusion matrix. The Prediction-Metadata stores information on the inference operations that take place after the training process. Moreover, in order to efficiently reproduce the experiments, we store information about the development environment (e.g., version, environment variables) using the Environment-Metadata entity which can be utilized to set up the target environment for reproducibility, sharing, and deployment.

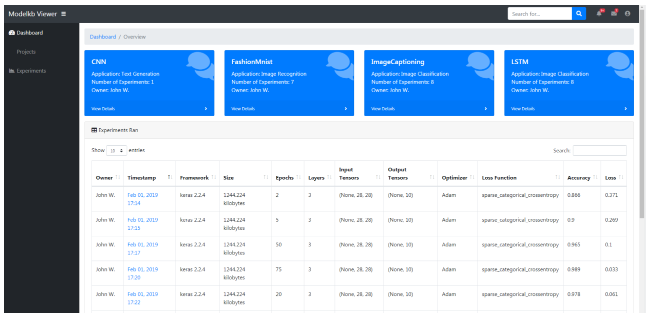
2.3 Experiment Code Generation

ModelKB introduces a novel feature that utilizes code generators to reduce the overhead in storing the source code for each experiment. Particularly, we do not store the training code for each experiment; instead, we developed a code generator that can regenerate the source code using the stored metadata upon request. This can significantly reduce the overhead and space complexity when storing, retrieving, and sharing experiments. The code generator is implemented using a native Python template library and can be easily extended to other programming languages. Our code generator can also be used to generate the architecture implementation from a given architecture description file (e.g., .h5, .JSON).

2.4 Data Visualization

ModelKB provides a local web-based visualization that allows the user to explore, analyze, compare, and test the models. Specifically, ModelKB provides three main views: *Dashboard*, *Project View*, and *Experiment View*. The *Dashboard* (see Figure 2) provides a high-level view of the projects stored in the system and a tabular view of the recent experiments. It allows searching for projects by name, owner, application, and the total number of experiments. The *Project View* lists a summary of all the experiments in the project using a dynamic table. The user can customize the information shown in the table using a drop-down menu. The user can also select different filters to query specific results and compare two experiments side-by-side based on the selected metadata. Clicking on any of the experiments will lead to the *Experiment View*.

The *Experiment View* (see Figure 3) focuses on visualizing one experiment at a time, and it includes three main tabs: *metadata*, *architecture*, and *test*. The *metadata* tab lists all the metadata that the user selects to view, including plots for the accuracy, loss, and a confusion matrix. The *architecture* tab visualizes the architecture of the model in an interactive interface, where the user can click any layer in the architecture to view its configuration. This functionality is provided by an open-source tool, called Netron [12]. The *test* tab allows the user to run on-the-fly predictions directly from the browser. For example, the user can upload an image and predict its label without having to deploy the model, which facilitates model selection and comparison.



CNN
Application: Text Generation
Number of Experiments: 1
Owner: John W.

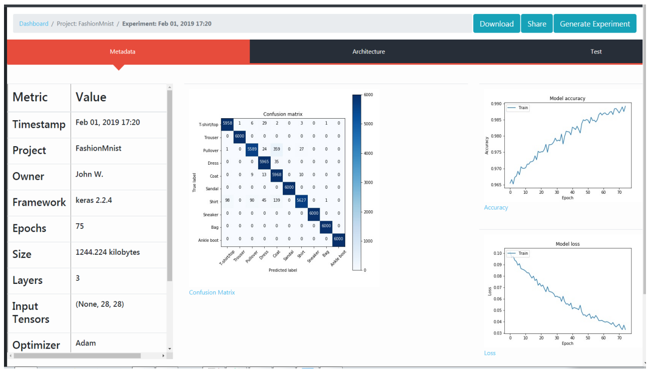
FashionMnist
Application: Image Recognition
Number of Experiments: 7
Owner: John W.

ImageCaptioning
Application: Image Classification
Number of Experiments: 8
Owner: John W.

LSTM
Application: Image Classification
Number of Experiments: 8
Owner: John W.

Owner	Timestamp	Framework	Size	Epochs	Layers	Input Tensors	Output Tensors	Optimizer	Loss Function	Accuracy	Loss
John W.	Feb 01, 2019 17:14	keras 2.2.4	1244.224 kilobytes	2	3	(None, 28, 28)	(None, 10)	Adam	sparse_categorical_crossentropy	0.866	0.371
John W.	Feb 01, 2019 17:15	keras 2.2.4	1244.224 kilobytes	5	3	(None, 28, 28)	(None, 10)	Adam	sparse_categorical_crossentropy	0.9	0.269
John W.	Feb 01, 2019 17:17	keras 2.2.4	1244.224 kilobytes	50	3	(None, 28, 28)	(None, 10)	Adam	sparse_categorical_crossentropy	0.965	0.1
John W.	Feb 01, 2019 17:20	keras 2.2.4	1244.224 kilobytes	75	3	(None, 28, 28)	(None, 10)	Adam	sparse_categorical_crossentropy	0.989	0.033
John W.	Feb 01, 2019 17:22	keras 2.2.4	1244.224 kilobytes	20	3	(None, 28, 28)	(None, 10)	Adam	sparse_categorical_crossentropy	0.978	0.061

Figure 2: ModelKB Dashboard



Metric	Value
Timestamp	Feb 01, 2019 17:20
Project	FashionMnist
Owner	John W.
Framework	keras 2.2.4
Epochs	75
Size	1244.224 kilobytes
Layers	3
Input Tensors	(None, 28, 28)
Optimizer	Adam

Figure 3: ModelKB Experiment View

3 RELATED WORK

The recent surge in machine learning has led to several efforts from the communities of data management, software engineering, and machine learning to develop methods and tools that facilitate the end-to-end management of machine learning. ModelDB [21] is one of the early systems that aimed at addressing model management issues, and it comes very close to our solution in its functionality. However, ModelDB is tailored for machine learning models built in *scikit-learn* and *spark.ml*, which provides limited support for DNNs.

ModelHub [10] is a high-profile deep learning management system that proposes a domain specific language to allow easy exploration of models, a model versioning system, and a deep-learning-specific storage system. It also provides a cloud-based repository. Schelter et al., [16] provide an automated tool to extract the model's metadata and provenance with an interactive visualization to query and compare experiments.

Other existing systems focus on addressing specific challenges. For example, TensorBoard [5] focuses on tracking and visualizing the modeling experiments within Tensorflow. ONNX [6] is a cross-platform model exchange system. Other systems focus on model packaging and reproducibility, including Repo2Docker [4] and CodaLab [3].

Overall, these tools are either tailored to a specific task or require manual code changes to extract the metadata. In contrast, we aim to automate the end-to-end model management by extracting large amounts of metadata automatically without any instrumentation, and then provide function wrappers to instrument functions that invoke the training process to extract training and evaluation metadata with minimal code changes.

4 CONCLUSIONS

In this paper, we discussed our ongoing work towards automating the management of deep learning experiments and presented our lightweight system ModelKB, which is undergoing user studies with deep learning practitioners from our lab and a developer from H&R Block Inc. Our future work aims at expanding ModelKB functionality to other deep learning frameworks and building a cloud-based repository with new features such as *model as a service* for sharing and reproducing deep learning models.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *OSDI*, Vol. 16. 265–283.

[2] François Chollet et al. 2015. Keras. https://keras.io.

[3] CodaLab. 2019. Accelerating reproducible computational research. http://codalab.org/

[4] Jessica Forde, Tim Head, Chris Holdgraf, Yuvi Panda, Gladys Nalvarete, Benjamin Ragan-Kelley, and Erik Sundell. 2018. Reproducible research environments with repo2docker. (2018).

[5] Google. 2019. TensorBoard: Visualizing Learning. https://www.tensorflow.org/guide/summaries_and_tensorboard

[6] ONNX Group. 2019. Open Model Exchange. https://onnx.ai/

[7] Arun Kumar, Robert McCann, Jeffrey Naughton, and Jignesh M Patel. 2016. Model selection management systems: The next frontier of advanced analytics. *ACM SIGMOD Record* 44, 4 (2016), 17–22.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[9] Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande. 2017. ModelHub: Deep Learning Lifecycle Management. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 1393–1394.

[10] Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande. 2017. Towards unified data and lifecycle management for deep learning. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 571–582.

[11] Leonardo Murta, Vanessa Braganholo, Fernando Chirigati, David Koop, and Juliana Freire. 2014. noWorkflow: capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop*. Springer, 71–83.

[12] Netron. 2019. Visualizing Deep Learning Models. https://github.com/lutzroeder/netron

[13] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2017. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1723–1726.

[14] Python. [n. d.]. Abstract Syntax Tree. https://docs.python.org/3/library/ast.html

[15] Sebastian Schelter, Felix Biessmann, Tim Januschowski, David Salinas, Stephan Seufert, Gyuri Szarvas, Manasi Vartak, Samuel Madden, Hui Miao, Amol Deshpande, et al. 2018. On Challenges in Machine Learning Model Management. *Data Engineering* (2018), 5.

[16] Sebastian Schelter, Joos-Hendrik Böse, Johannes Kirschnick, Thoralf Klein, and Stephan Seufert. 2017. Automatically tracking metadata and provenance of machine learning experiments. In *Machine Learning Systems Workshop at NIPS*.

[17] Sebastian Schelter, Joos-Hendrik Böse, Johannes Kirschnick, Thoralf Klein, and Stephan Seufert. 2018. Declarative Metadata Management: A Missing Piece in End-To-End Machine Learning. (2018).

[18] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*. 2503–2511.

[19] Jason Tsay, Todd Mummert, Norman Bobroff, Alan Braz, Peter Westerink, and Martin Hirzel. 2018. Runway: machine learning model experiment management tool.

[20] Manasi Vartak. 2018. *Infrastructure for model management and model diagnosis*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[21] Manasi Vartak, Harihar Subramanyam, Wei-En Lee, Srinidhi Viswanathan, Saadiyah Husnoo, Samuel Madden, and Matei Zaharia. 2016. Model DB: a system for machine learning model management. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM, 14.

[22] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. 2018. Accelerating the Machine Learning Lifecycle with MLflow. *Data Engineering* (2018), 39.