CEREBRO: A System to Manage Deep Learning for Relational Data Analytics

Arun Kumar
University of California, San Diego
arunkk@eng.ucsd.edu

Deep learning a.k.a *deep neural networks* (DNNs) are pushing the state of the art in AI tasks such as image and speech recognition [3]. Leading Web companies are betting big on DNNs. Beyond all the hype, a key question remains: *Will deep learning transform relational data analytics, which is critical for multi-billion dollar tasks like customer churn prediction in enterprise domains?* Machine learning (ML) over relational data systems is a major focus of our community; thus, this is an existential question for this line of work.

Conversations with data scientists at many enterprise companies revealed that there is indeed broad interest in DNNs, but there are at least two major non-economic challenges to their adoption. First, unlike multimedia, relational data typically have *interpretable* features. Preserving interpretability is key for business purposes but DNNs obscure it. Fortunately, ML folks are already tackling this challenge [1]. Second, unlike images/speech, where models are more reusable across domains, relational data have different schemas and features across domains even for the same task; this forces data scientists to perform the painful process of *model engineering* on their data: tuning the number of layers and neurons, their connectivity, and hyper-parameters. Like *feature engineering*, these decisions depend on several factors: data schema, attributes' properties, database dependencies, time budget, resources, etc. This process involves data and model transformations, tracking such changes, and archiving the models for *inference* and auditing. Alas, existing DNN tools provide little support for this iterative process. This forces data scientists to manage such information mostly manually and lowers their productivity. Also, since DNN tools are separate from production systems such as RDBMSs, inference becomes a cumbersome task straddling two systems.

We propose Cerebro, *the first data system to support the process of model engineering for deep learning over relational data*. Our goal is three-fold: (A) make it easier and faster to build high quality DNNs over relational data, (B) archive DNNs and manage their provenance to help with auditing and with (A), and (C) integrate DNN inference with an RDBMS to make it easier to use DNNs in production.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

We plan to prototype Cerebro on top of TensorFlow using the abstraction of a *model selection triple* (MST) [2]. We abstract a DNN as a triple (FE, AS, PT) that is changed iteratively: FE (feature engineering) represents data transformations at the input layer (e.g., joins and aggregates), AS (algorithm selection) represents the graphical structure (hidden layers), and PT (parameter tuning) represents the hyper-parameters and activation functions. This abstraction helps us explore alternate physical layouts for large DNNs. We plan to use PostgreSQL for storage in our first design to exploit its indexing/querying capabilities. We will add a simple embedded domain specific language (DSL) to declare MST changes, capture popular DNN structures, and make it easier for us to store and search over DNNs. We also envision providing *model recommendations* based on data properties, past DNNs, and MST changes in a model engineering *session*. We do not aim to support all possible DNNs in our first design; rather, we will support a large class of popular DNNs for *classification* and make our design extensible.

Cerebro raises several new research questions at the intersection of data management and ML; we outline some and discuss why they are interesting from a data management perspective. (i) What is the "data model" to represent DNN MSTs? An effective representation should provide opportunities to optimize physical layout and indexing. Such storage-runtime trade-offs are reminescent of RDBMS storage systems. (ii) What DNN structures and operations should the DSL capture? This also involves determining how they interact with existing operations in TensorFlow, similar to declarative dataflow languages. (iii) How best to exploit DNN MSTs to reduce runtimes across iterations? This involves new cost-based optimizations to avoid redundancy in both data and model transformations and optimizing execution over both GPUs and CPUs. (iv) How to optimize DNN inference in an RDBMS for large data? This involves jointly optimizing database accesses with DNN computations, exploiting any database dependencies, and managing both data and model parallelism. (v) How to recommend DNN structures and MST changes for a new task/dataset using an archive of DNNs? This involves characterizing the effects of different DNN MSTs on accuracy and devising recommendation algorithms, potentially using ML.

1. REFERENCES

[1] Z. Che et al. Distilling Knowledge from Deep Networks with Applications to Healthcare. *CoRR*, abs/1512.03542, 2015.
[2] A. Kumar et al. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *SIGMOD Rec.*, 44(4):17–22, Dec. 2015.
[3] Y. Lecun et al. Deep Learning. *Nature*, 521:436–444, 5 2015.