Reimagining Deep Learning Systems Through the Lens of Data Systems

Arun Kumar
University of California, San Diego
La Jolla, CA, USA
akk018@ucsd.edu

ABSTRACT

The high-profile success of Deep Learning (DL) at Big Tech companies, including recent Large Language Models (LLMs) such as the GPT and Llama families, has led to high demand among Web companies, consumer app companies, enterprises, healthcare, domain sciences, and even digital humanities and arts to adopt modern DL for their applications. The scale of DL workloads, domain-specific datasets, and publicly available pre-trained base models keeps growing. Naturally, tackling issues of *scalability*, *usability*, and *resource/cost efficiency* of DL systems are critical to democratizing modern DL-powered AI. We find that some key lessons from the decades of work on data system design, implementation, and optimization–when adapted prudently–can go a long way toward that goal. Specifically, our work shows that new analogues of *multi-query optimization* for DL systems can substantially reduce runtimes and costs, while improving ease of use. This article lays out how we reimagine DL workloads that way and summarizes the technical contributions powering this transformation.

PVLDB Reference Format:
Arun Kumar. Reimagining Deep Learning Systems Through the Lens of Data Systems. PVLDB, 17(12): 4531-4535, 2024.
doi:10.14778/3685800.3685914

1 RESEARCH GAPS IN PRIOR ART

Democratizing DL systems is not a new goal–it is shared across many research communities, including at least distributed systems, high-performance computing (HPC), compilers, and computer architecture. But we observed that much of the prior work *miss the forest for the trees*! More precisely, most work on systems issues in DL had focused on making *single-model execution* for training or inference faster. While that is certainly useful, it is not sufficient because it does not account for *actual end-user behaviors* in DL practice. From our conversations with 30+ AI/data science practitioners across diverse settings, we observed that this gap often leads to high wastage of GPU resources, reduced DL user productivity, and in turn, higher overall runtimes and costs.

2 BACKGROUND AND OUR VISION

Let us start with some AI 101-style facts. We will then make the connection to data systems-style techniques.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685914

DL is a highly empirical endeavor, with numerous types of knobs that affect accuracy, reliability, runtimes, and costs. Since the Pareto tradeoffs on those (and other) metrics are application specific, there is no single universal setting for such knobs. Thus, the ability to rapidly tweak such knobs for one's own data and application is critical to getting "shippable" models. In ML technical parlance, such knob tuning is collectively called *model selection*: an inevitable overarching process to balance overfitting vs. underfitting when building any ML model for an application's data distribution [7].

In DL, model selection involves 3 main types of knobs: (1) alter the data input and/or output representation, since DL operates on tensorized data; (2) alter the neural computational graph architecture, including when transfer learning from pre-trained models; and (3) tune learning hyperparameters, which is common in classical ML too. Naturally, all this often leads to many combinations of configurations being tried, regardless of whether they are specified manually, with AutoML heuristics, or something in between.

Herein lies our fundamental reimagination: we cast such multi-configuration DL workloads as "multi-query" execution for DL "queries" on DL systems, akin to relational queries on RDBMSs. Figure 1(A) makes this intellectual stack analogy more precise.

With the above analogy in mind, we have been laying the technical foundations of *multi-query optimizations for DL workloads on DL systems*. Our techniques are mostly complementary to work from those other communities, since they exist at other levels in the stack. As we DB folks know and love, query optimization takes a more *holistic worldview* on improving data systems by exploiting the *physical, logical, and/or semantic properties* of the workloads.

Our techniques also span the gamut of optimizing *compute, memory/storage, and/or network resources*. It includes better co-placement of data and computations, reducing redundancies in data movement and/or computations, new forms of hybrid parallelism for computations, and semantics-aware optimization of computations. Overall, our techniques help reduce runtimes, costs, and energy footprints of DL, while raising user productivity and ultimately helping democratize modern AI.

3 TYPES OF DL WORKLOADS

Based on our conversations with DL practitioners, our in-depth study of the DL literature, and our first-hand experiences with large-scale DL use cases in domain sciences, we decompose multi-query execution in DL workloads into 3 main groups:

Creation/Updates: This group involves *hyperparameter tuning* and *neural architecture design/tuning*. The former is inevitable in almost all DL scenarios. The latter is particularly common for DL on time series data, other semistructured sequences, and some tabular

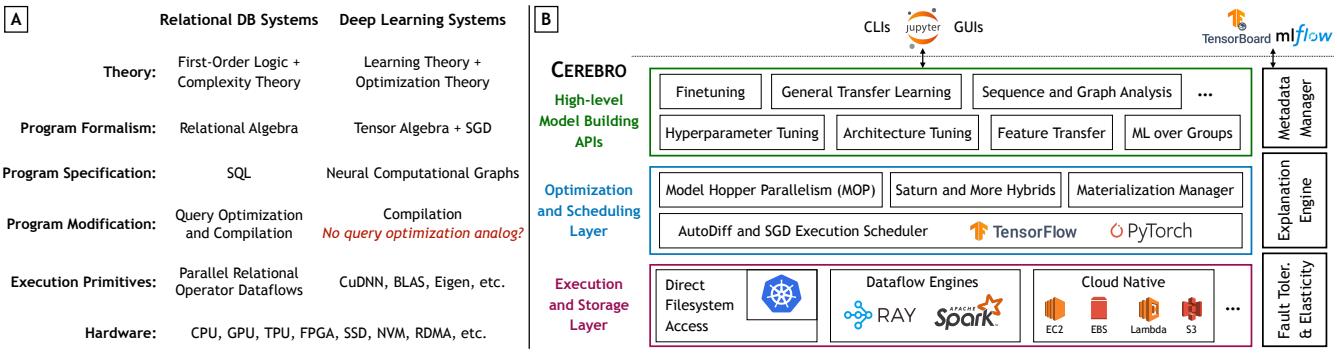

Figure 1: (A) Intellectual stack analogy of RDBMS and DL systems. (B) Our envisioned model selection-first architecture.

data tasks. These workloads also involve *data-centric tweaking*, e.g., subsetting the data on some features, subsetting the features of the data (especially for multimodal use cases), and/or adding/updating the (labeled) data over time.

Transfer: This group involves *feature transfer*, viz., using pre-trained models as featurizers for raw data, *finetuning*, viz., continued training of a pre-trained model on new data, and more general *transfer learning*, viz., dropping/modifying a pre-trained model's layers and/or weights, adding new task-specific layers, mixing and matching pre-trained layers with new layers, etc. Such workloads are now the norm for image, video, and text data using CNNs, Transformers, and/or Diffusion models downloaded from model hubs such as HuggingFace. Due to task-specific customization, this group typically uses the first group as an inner loop, e.g., to compare alternate layers of features, new layers, and/or hyperparameters.

Prediction: This group involves repeated inference requests made to the trained and deployed models. Batching of requests at deployment time is common. Repeated inference with pre-trained models also arises in the second group, representing an overlap between these workloads. Finally, many prediction "explanation" schemes also alter the data and perform repeated inference.

4 TECHNICAL CONTRIBUTIONS

With the analogy and the DL workload characterizations in mind, we now dive into our series of novel multi-query optimization (MQO) and related techniques for DL systems. We group the techniques by the abstraction level of the optimization. To the best of our knowledge, these techniques are all a first of their kind in the whole DL systems landscape. These techniques were published across 8 full research papers at SIGMOD/VLDB, 1 CIDR paper, 2 invited TODS/SIGMOD Record papers, 3 demo/workshop papers at SIGMOD/VLDB/MLSys, and 5 public health journal papers based on DL models built using our systems on large datasets collected by our collaborators.

4.1 Physical-level Hybrid Parallelism

In **Cerebro-MOP** [16] we tackled a key bottleneck for DL training and model selection: scaling to large datasets. Prior art such as PyTorch DDP and Horovod for data-parallel training on a cluster are too communication-heavy and highly sub-optimal for model selection because they do not exploit the multi-model degree of parallelism. Breaking this (false) dichotomy of task-parallelism and data-parallelism we created a novel hybrid: *model hopper parallelism* (MOP). The basic idea of MOP is to enable scheduling of different models on different shards of the data concurrently and coordinating in a resource-aware and fault-tolerant manner across epochs and models. All models learn on all the data in a sequential-equivalent manner, thus not hurting accuracy. MOP is the first known form of *bulk asynchronous parallelism*, fundamentally advancing the parallel data systems literature. Theoretically and empirically we showed MOP is effectively resource-optimal in this setting. We also hybridized MOP with Horovod/DDP for cases with fewer models than workers.

Saturn [14] and **Hydra** [13] tackled the complementary setting of small dataset but large model that does not fit on a single GPU's memory. Prior art such as Google GPipe, Microsoft DeepSpeed, and PyTorch FSDP handle large models by sharding them across GPUs. Again, their approach of training one model at a time is too communication-heavy and sub-optimal in the model selection context because they too do not exploit multi-model degree of parallelism. They also require tuning many system parameters that are unintuitive for end-users of DL who may not have systems expertise. We devised two systems in this context: Hydra and its successor Saturn.

Hydra broke the (false) dichotomy of task- and model-parallelism to create a novel hybrid: *shard alternator parallelism* (SHARP), also a new form of bulk asynchronous parallelism. Analytically and empirically, we showed Hydra is near resource-optimal and significantly faster than those prior tools for large-model DL. But Hydra requires our own model sharding layer, which is painful to maintain as the underlying DL tools evolve.

Saturn, Hydra's successor, tackles those issues by operating as a *meta-level hybrid-parallel optimizer* on top of unmodified PyTorch FSDP and GPipe. It selects the exact parallelization scheme to use for a given workload, akin to how a relational query optimizer picks between alternate physical relational operator implementations. We unified such automated selection of parallel execution with auto-tuning of their system parameters and GPU apportioning across models in a model selection workload. Put together, Saturn

raises GPU utilization by 40%-50% compared to current practice of one-at-a-time single-model parallelism on such tools.

Kingpin [10] targets a common ML/DL practice: building separate models for data subgroups (e.g., per state). We observed that this is essentially a form of GROUP BY, albeit for ML model selection instead of SQL aggregates. Prior art manually materializes data subsets for such GROUP BY execution, which is sub-optimal due to wasted memory/storage and/or high communication costs. Inspired by how RDBMSs implement hashing-based GROUP BY operators, we proposed another novel hybrid of task- and data-parallelism: *gradient accumulation parallelism* (GAP). Analytically and empirically, we showed that GAP and associated techniques that build on it are substantially more resource-efficient for not just DL but also GBDT and GLMs. GAP is yet another novel form of bulk asynchronous parallelism.

Lotan [24] targets graph neural networks (GNNs), which capture non-IID data. GNNs are increasingly common in both domain sciences and Web companies, but their graph-oriented data access patterns are more complex than the sequential scans for SGD on IID data. The prior art was bespoke custom graph systems for GNNs. But we asked a seemingly simple question: *How far can we push an existing graph analytics engine to scale GNNs?* Perhaps surprisingly, we found that graph engines can indeed help scale GNN workloads defined via DL tools. Our prototype named Lotan unified PyTorch and GraphX to enable *decoupled scaling and parallelization*: put DL arithmetic on the DL tool and graph gather-scatter operations on the graph engine. To reason about our optimizations, we devised a novel physical operator algebra for GNN dataflows that is more coarse-grained than the tensor algebra of DL tools but fine-grained enough for us to optimize. That also enabled us to devise the first-known operator batching for GNNs, another new form of MQO in DL. Empirically, we found Lotan to be the most scalable system for GNNs, surpassing even industrial tools from Alibaba, while offering competitive time-to-accuracy on large GNN benchmarks.

4.2 Query Rewrites and Materialized Views

Vista [20] is the first system to study and optimize a now-ubiquitous DL workload: transfer learning. In image/video analytics, DL users transfer features from pre-trained CNNs to help boost accuracy for their new target task, but they often need to compare different layers because their accuracy can differ significantly. This causes *redundant computations* during CNN re-inference in the layers that are common. Thus, we proposed a novel query rewriting and MQO scheme to share those layers by automatically creating "materialized views" and injecting them into the workload. We showed that Vista's approach offers up to 5x speedups on real image analytics workloads. This is the first time subsets of neural computational graphs were reimagined as view definitions that can be materialized and reused, inspired by how subsets of relational queries are defined and reused as materialized views.

Nautilus [21] expands upon Vista's direction to all types of DL architectures beyond CNNs, including Transformers and DAG structures, as well as for more general transfer learning schemes such as layer-wise freezing. Given a set of model finetuning or transfer learning tasks, Nautilus automatically constructs a "super-model" that shares the common layers, a novel form of MQO we call "model fusion." It reduces both redundant computations and intermediate data movement between DRAM and GPU. We also devised incremental view maintenance-style techniques to support evolving training data, say, when labeled data is added over time. Overall, Nautilus offers up to 5x speedups by trading off some storage space and also raises GPU utilization substantially. With the recent boom of generative AI workloads that transfer learn from pre-trained Transformers (e.g., the Llama family) or Diffusion models (e.g., Stable Diffusion), we believe such MQO techniques to make the process faster and cheaper can help more users tailor generative AI to their in-house (and often private) labeled data.

4.3 Incremental View Maintenance

Krypton [15] targets a popular form of "explanation" for a CNN's predictions on an image: hide a patch of pixels, perform re-inference, slide the patch to do re-inferences, and get a heatmap of which pixels "matter" most. We observed that most features created by the CNN will not change much due to spatial locality. So, we reimagined this as a form of incremental view maintenance (IVM): materialize all CNN features on the original image and incrementally update only those that actually change. This is the first known form of IVM for neural computational graphs in DL systems. Going further, we also batched the patched re-inference requests to share computations and data movement between DRAM and GPU–the first-known *merger of IVM with MQO* in the data systems literature. We showed all this works on both CPUs and GPUs, yielding even 10x speedups on real radiology and image analytics datasets. This work was accorded a SIGMOD'19 Best Paper Honorable Mention and a SIGMOD Research Highlight Award [18]. In an invited longer paper, we expanded Krypton's IVM framework to also optimize video stream inference, exploiting the pixel redundancy across adjacent frames [17].

4.4 Semantics-based Approximations

Also in **Krypton**, we built on top of our IVM framework to propose two CNN semantics-based approximation heuristics to raise speedups further. One is *projective field thresholding*: clip the breadth-wise spread of changes across CNN layers to reduce FLOPs count, inspired by vision neuroscience. The other is *adaptive drill-down*: reduce heatmap resolution for "uninteresting" regions, inspired by BI practices of drilling down or rolling up in data cubes.

4.5 System Design and Implementation

We enjoy building systems to prototype our ideas and using them for real workloads when possible. All the techniques listed above have been prototyped as open source tools on top of PyTorch or TensorFlow. We have also demonstrated some of our tools at VLDB [9] and MLSys [22]. Our system design and implementation philosophy has 3 key guidelines to maximize potential for impact:

(1) *Do not change the DL tool's internal code.* This ensures backward compatibility as those tools evolve and lets us piggyback on complementary advances made by the Big Tech firms on their compilers or hardware support. This decision proved helpful with

PyTorch in particular, which has grown massively in the last few years in terms of both features and user base.

(2) *Overload already popular user-facing APIs* such as Keras and AutoML heuristics for specifying the model selection workloads. Specifically, we have built support for grid search, random search, and heuristics such as Hyperopt and ASHA. All this can lower friction for DL users to adopt our tools. Of course, for novel capabilities not present in those APIs, e.g., like for Nautilus or Krypton, we devised our own intuitive high-level APIs in Python.

(3) *Be ecumenical on backend infrastructure.* For system prototyping, our work has spanned filesystem acccess on Kubernetes clusters, mediated access via a data processing system (Spark, Ray, and Greenplum), and direct cloud IaaS access. None of our techniques are dependent on a particular data backend. This enables portability across stacks and meeting DL users where they are.

The above said, we have also assembled some of these techniques into a *loosely coupled but unified stack* we call the Cerebro platform [8, 9]. Figure 1(B) shows its layered architecture. It is under active development, and an initial version on a Kubernetes cluster is being rolled out to more domain scientists at UCSD via the Voyager Supercomputer at the San Diego Supercomputer Center (SDSC). Cerebro is the first model selection-first DL platform to enable full out-of-the-box scalability on all axes of interest: datasets, models, tasks, and resources.

5 PRACTICAL ADOPTION SO FAR

Apart from building systems and publishing about our techniques, we have also worked with relevant collaborators in both domain sciences and industry to translate some of this work to practice.

(1) In an NIH-funded collaboration with UCSD Public Health researchers, we used Cerebro to analyze TB-scale labeled time series data from body-worn accelerometers to study human sedentary behaviors and their impact on health. Cerebro's high-throughput model selection functionality enabled us to raise balanced accuracy for a binary classification task (sitting or not) from their prior art of 76% to a massive 92%. The models built with Cerebro are now the state of the art in their field. This collaboration has led to 5 full papers in prominent public health journals [3–6, 19]. Our models and code are all open sourced [2], and they are already being used by external public health researchers in this field in the USA and Australia. In a follow-on NIH-funded project, we are extending such modeling and usage of Cerebro and Nautilus to more tasks and cohorts. This includes using our pre-trained models as the base models for transfer learning to reduce both the amount of labeled data needed and compute resources needed.

(2) Pivotal/VMware adopted MOP for DL workloads on data resident on Greenplum and shipped it as part of their Apache MADlib library for in-RDBMS DL [1, 11, 12]. They used this tool for some enterprise analytics use cases involving multimodal data analytics combining relational and NLP/vision/time series use cases. In turn, this collaboration led to additional interesting research on the larger tradeoff space of how to bridge the gap between DL tools and data resident in parallel data warehouses/lakehouses [23].

(3) In another domain science collaboration at UCSD, we have applied techniques from Hydra to computational physics use cases involving fluid dynamics simulations. More use cases in public health, political science, and neuroscience involving time series, text, and video data are also in the works.

(4) Some Big Tech companies, including Meta and Netflix, have explored adopting some of our techniques and tools (specifically Hydra, Saturn, and/or MOP) for internal usage. Their use cases include computational advertising and recommendation systems.

(5) Finally, we have co-founded a new startup named RapidFire AI that is building the industry's first "agile DL platform" in the cloud to help democratize DL-based AI. Our product applies techniques from this research and also extends it along new directions. As we roll out our product to customers, we hope to learn more from their practical challenges to help inform future research at this fast-growing and important intersection of the DB and AI worlds.

SPEAKER BIOGRAPHY

Arun Kumar is an Associate Professor in Computer Science and Engineering and the Halıcıoğlu Data Science Institute at the University of California, San Diego. His primary research interests are in data management and systems for ML/AI-based data analytics. Systems and ideas based on his research have been shipped as part of products from, or used internally by, multiple cloud, Web, and database systems companies, including Google, Meta, Oracle, and VMware. He is a recipient of three SIGMOD research paper awards, six distinguished reviewer/metareviewer awards from SIGMOD/VLDB, the IEEE TCDE Rising Star Award, an NSF CAREER Award, a UCSD oSTEM Faculty of the Year Award, and research award gifts from Amazon, Google, Oracle, and VMware. His first PhD graduate received the ACM SIGMOD Jim Gray Doctoral Dissertation Award. He is also Cofounder and CTO of a startup named RapidFire AI that is building the world's first agile DL platform to make it easier, faster, and cheaper for companies of all sizes to be more successful in translating AI to business impact.

ACKNOWLEDGMENTS

This work was supported in part by a Hellman Fellowship, the NIDDK of the NIH under award number R01DK114945, an NSF CAREER Award under award number 1942724, and gifts from VMware. The content is solely the responsibility of the author and does not necessarily represent the views of any of these organizations. None of this work would have been possible without the passion and creativity of the amazing students who drove this research agenda forward: Supun Nakandala, Yuhao Zhang, Kabir Nagrecha, Side Li, Liangde Li, and Advitya Gemawat. We thank the members of UC San Diego's Database Lab, Center for Networked Systems, Loki Natarajan and our public health collaborators, Frank McQuillan and the Apache MADlib/Greenplum team at VMware, and Amitava Majumdar and the SDSC Voyager team for their collaborations and/or feedback on parts of this work. We thank Yannis Papakonstantinou, Raghu Ramakrishnan, Samuel Madden, and Volker Markl for their strong support of the award nomination. Finally, we thank the VLDB 2024 Awards Committee for the honor and the VLDB 2024 team for this opportunity to speak more about our work.

REFERENCES

[1] [n.d.]. Apache MADlib DL Model Selection with MOP. http://madlib.apache.org/docs/latest/group__grp__keras__run__model__selection.html.

[2] [n.d.]. Deep Postures GitHub repository. https://github.com/ADALabUCSD/DeepPostures.

[3] John Bellettiere et al. 2022. CHAP-Adult: A Reliable and Valid Algorithm to Classify Sitting and Measure Sitting Patterns Using Data from Hip- Worn Accelerometers in Adults Aged 35+. *Journal for the Measurement of Physical Behaviour* (2022).

[4] Jordan A. Carlson et al. 2022. CHAP-child: An open source method for estimating sit-to-stand transitions and seden- tary bout patterns from hip accelerometers among children. *International Journal of Behavioral Nutrition and Physical Activity* (2022).

[5] Mikael Anne Greenwood-Hickman et al. 2021. The CNN Hip Accelerometer Posture (CHAP) Method for Classifying Sitting Patterns from Hip Accelerometers: A Validation Study. *Medicine and Science in Sports and Exercise Journal* (2021).

[6] Paul R. Hibbing et al. 2023. Low movement, deep-learned sitting patterns, and sedentary behavior in the International Study of Childhood Obesity, Lifestyle, and the Environment (ISCOLE). *International Journal of Obesity* (2023).

[7] Arun Kumar et al. 2016. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *SIGMOD Rec.* 44, 4 (May 2016), 17–22.

[8] Arun Kumar et al. 2021. Cerebro: A Layered Data Platform for Scalable Deep Learning. In *CIDR*.

[9] Liangde Li et al. 2021. Intermittent Human-in-the-Loop Model Selection Using Cerebro: A Demonstration. In *VLDB*.

[10] Side Li and Arun Kumar. 2021. Towards an Optimized GROUP by Abstraction for Large-Scale Machine Learning. In *VLDB*.

[11] Frank McQuillan. [n.d.]. Efficient Model Selection for Deep Neural Networks on Massively Parallel Processing Databases. https://archive.fosdem.org/2020/schedule/event/mppdb/.

[12] Frank McQuillan. [n.d.]. Model Selection for Deep Neural Networks on Greenplum Database. https://tanzu.vmware.com/content/blog/model-selection-for-deep-neural-networks-on-greenplum-database.

[13] Kabir Nagrecha and Arun Kumar. 2022. Hydra: A Data System for Large Multi-Model Deep Learning. *Technical report.*

[14] Kabir Nagrecha and Arun Kumar. 2024. Saturn: Resource-Aware Multi-Query Optimization for Multi-Large-Model Deep Learning Workloads. In *VLDB*.

[15] Supun Nakandala et al. 2019. Incremental and Approximate Inference for Faster Occlusion-Based Deep CNN Explanations. In *SIGMOD*.

[16] Supun Nakandala et al. 2020. Cerebro: A Data System for Optimized Deep Learning Model Selection. In *VLDB*.

[17] Supun Nakandala et al. 2020. Incremental and Approximate Computations for Accelerating Deep CNN Inference. *ACM Trans. Database Syst.* 45, 4, Article 16 (Dec. 2020).

[18] Supun Nakandala et al. 2020. Query Optimization for Faster Deep CNN Explanations. *SIGMOD Rec.* 49, 1 (2020), 61–68.

[19] Supun Nakandala et al. 2021. Application of Convolutional Neural Network Algorithms for Advancing Sedentary and Activity Bout Classification. *Journal for the Measurement of Physical Behaviour* (2021).

[20] Supun Nakandala and Arun Kumar. 2020. Vista: Optimized System for Declarative Feature Transfer from Deep CNNs at Scale. In *SIGMOD*.

[21] Supun Nakandala and Arun Kumar. 2022. Nautilus: An Optimized System for Deep Transfer Learning over Evolving Training Datasets. In *SIGMOD*.

[22] Allen Ordookhanians et al. 2019. Demonstration of Krypton: Optimized CNN Inference for Occlusion-based Deep CNN Explanations. In *VLDB*.

[23] Yuhao Zhang et al. 2021. Distributed Deep Learning on Data Systems: A Comparative Analysis of Approaches. In *VLDB*.

[24] Yuhao Zhang and Arun Kumar. 2023. Lotan: Bridging the Gap between GNNs and Scalable Graph Analytics Engines. In *VLDB*.