Intermittent Human-in-the-Loop Model Selection using Cerebro: A Demonstration

Liangde Li
University of California, San Diego
lil009@ucsd.edu

Supun Nakandala
University of California, San Diego
snakanda@eng.ucsd.edu

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

ABSTRACT

Deep learning (DL) is revolutionizing many fields. However, there is a major bottleneck for the wide adoption of DL: the *pain of model selection*, which requires exploring a large configuration space of model architecture and training hyper-parameters before picking the best model. The two existing popular paradigms for exploring this configuration space pose a false dichotomy. AutoML-based model selection explores configurations with high-throughput but uses human intuition minimally. Alternatively, interactive human-in-the-loop model selection completely relies on human intuition to explore the configuration space but often has very low throughput. To mitigate the above drawbacks, we propose a new paradigm for model selection that we call *intermittent human-in-the-loop model selection*. In this demonstration, we will showcase our approach using five real-world deep learning model selection workloads. A short video of our demonstration can be found here: https://youtu.be/K3THQy5McXc.

PVLDB Reference Format:
Liangde Li, Supun Nakandala, and Arun Kumar. Intermittent Human-in-the-Loop Model Selection using Cerebro: A Demonstration. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Deep learning (DL) is revolutionizing many fields. It is now being used in various domains including e-commerce, web, and even in critical applications such as in healthcare. However, there is a major bottleneck for the wide adoption of DL: the *pain of model selection*. The accuracy of a trained model heavily depends on the model architecture and hyper-parameter values used during training. Thus, practitioners often have to perform a search over the potential configuration (config) space, in order to pick the best model.

Paradigms for Searching the Configuration Space. From our conversations with DL practitioners and our own experience building large-scale DL applications we find two main paradigms: 1) AutoML and 2) interactive human-in-the-loop. In the AutoML paradigm, the user will initiate a model selection workload by specifying a config search space and a canned AutoML procedure. AutoML procedures implement a search heuristic such as Bayesian optimization (e.g., HyperOpt [2]), evolutionary search (e.g., PBT [5]), and random search (e.g., ASHA [9]). It then uses the parallelism available in a cluster (or a single machine) to explore configs with high throughput. As model selection progresses, the user will receive the results of the explored configs. Figure 1 (A) presents an illustration of this paradigm. While there are advanced AutoML procedure implementations of the above-mentioned heuristics, recent surveys [3] have shown that an overwhelming majority of ML researchers and practitioners often use simple techniques like grid (explore all configs) or random (randomly sample configs) search.

In interactive human-in-the-loop model selection [4, 14], the user retains full control over the search process. They will explicitly specify a config (or few configs) to explore and wait until it finishes. Based on the results of the explored configs and human intuition about the search space, they will specify the next config (or set of configs) to explore. Figure 1 (B) illustrates this paradigm.

False Dichotomy of Existing Paradigms. We contrast the above paradigms on two dimensions: 1) model exploration throughput and 2) the ability to use human intuition. As shown in Figure 1 (D), AutoML-based model selection explores configs with high throughput. But the only time it relies on human intuition is during the initial specification of the search space. As a result, it may inefficiently explore the config space and incur significant resource costs, which could have been avoided by a simple human intervention. On the other hand, the human-in-the-loop model selection primarily relies on human intuition but operates at very-low throughput levels due to the inherent limitations of human interactivity. Also, many DL configs are so long-running that false promises of "interactivity" become a prison for DL practitioners that wastes their time. Overall, we see a major gap between AutoML-based and human-in-the-loop model selection paradigms today.

This Work. To overcome the above-mentioned drawbacks, we propose a new paradigm we call *intermittent human-in-the-loop model selection*. It is a hybrid of both AutoML-based and interactive human-in-the-loop model selection. However, unlike the latter, human exploration is not mandatory in our approach. As an analogy, the interactive exploration is akin to instant messaging (IM), whereas our paradigm is akin to email threads or Slack channels. Without interactivity, the former becomes not usable. But our approach is more flexible due to asynchronous, spread-out-over-time yet stateful exchanges that can still subsume full interactivity. We implement our paradigm in CEREBRO, a new platform for resource-efficient deep learning model selection [11]. We extend CEREBRO with a graphical user interface, a REST API, and change existing components to support our new paradigm. In this demonstration, we will allow the audience to use CEREBRO to perform intermittent human-in-the-loop model selection using 5 real-world DL model selection workloads. Our paradigm is an ideal fit for DL model selection workloads due to their long-running nature,

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

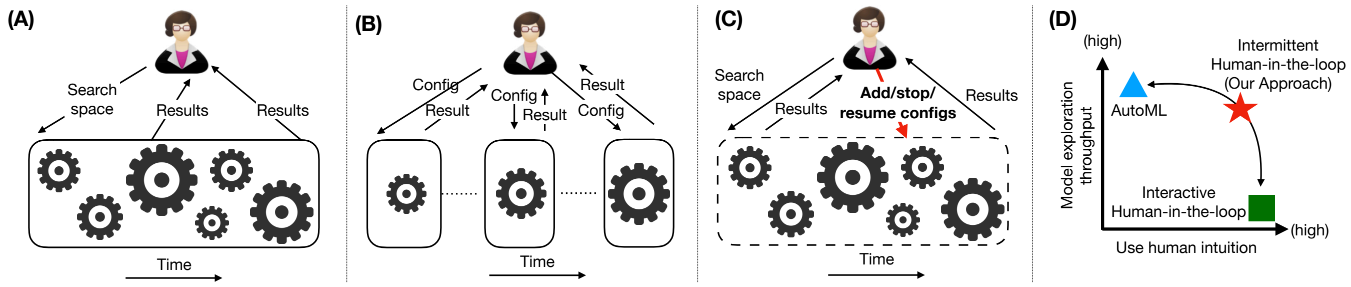


Figure 1: A) AutoML-based model selection. B) Interactive human-in-the-loop model selection. C) Our paradigm of *intermittent human-in-the-loop model selection.* D) Qualitative comparison of different paradigms.

and thus, we focus on DL for now. But it is readily applicable to any other ML model family too. A short video of our demonstration can be found here: https://youtu.be/K3THQy5McXc.

2 TECHNICAL CONTRIBUTIONS

2.1 New Paradigm for Model Selection

Our intermittent human-in-the-loop paradigm breaks the false dichotomy of AutoML-based and interactive human-in-the-loop model selection. It is motivated by both observations about model selection practice [6] and our experience in training DL models for public health applications [7]. It starts similar to the AutoML-based paradigm where the user specifies the search space and picks a canned AutoML procedure like Grid, Random, or even a more advanced one like HyperOpt. However, instead of passively waiting by just consuming the results of explored configs, we enable the user to steer the model selection process. User can now *create* new individual configs or batch of configs using a refined search space, *stop* running configs, and *resume* stopped configs.

Creating new configs outside the control of the AutoML procedure enables the user to inject human intuition into the overall model selection process. New configs can also be created by first cloning an existing configuration along with its trained parameters and then by tweaking only some of the hyper-parameters like learning rate or batch size. Users can use this feature to make the model training adaptable based on human intuition. They can also dynamically reprioritize the training of some configs over the others by using the stop and resume feature. Thus, as shown in Figure 1 (D) our paradigm can seamlessly navigate the exploration throughput and human intuition usage tradeoff space based on the available user interaction level. In a sense our approach fulfils the desire for “dialogue with the algorithms” we have heard from many ML/DL practitioners, except neither party is forced to respond promptly.

2.2 UIs for Intermittent Specification

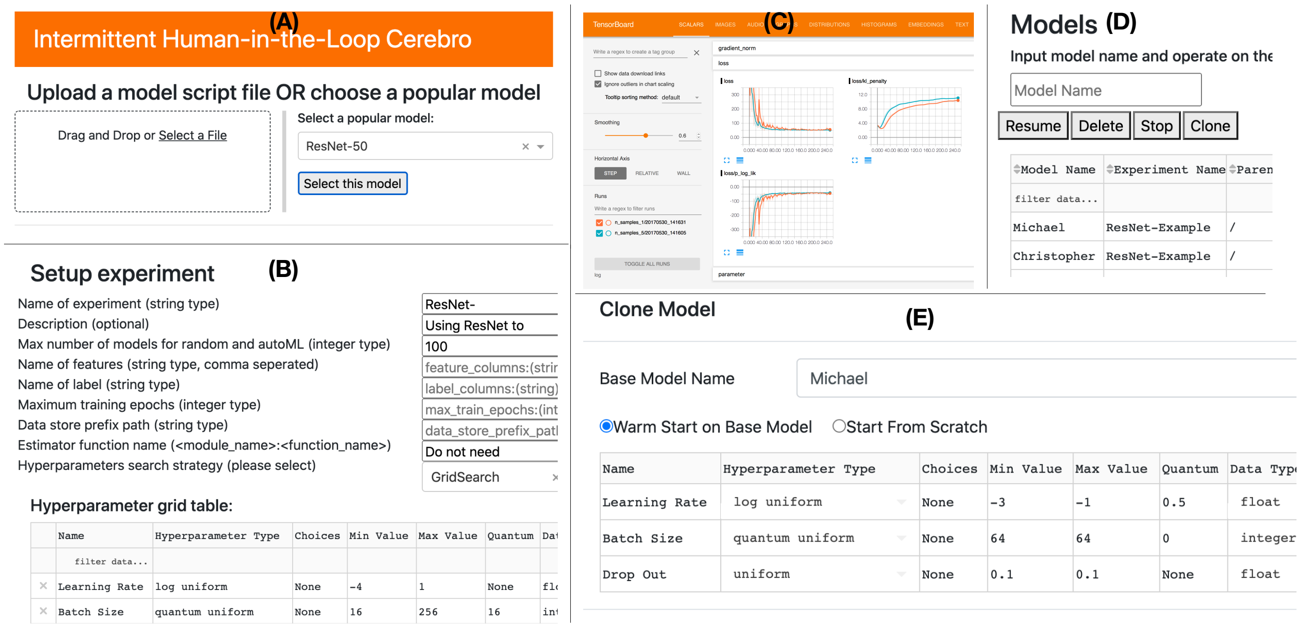
System UI provides graphical controls that enable the user to perform intermittent human-in-the-loop model selection. It is implemented using Python Dash visualization library and runs in a web browser which makes it portable. It is integrated with a backend REST API to perform the user-requested actions.

The user will start interacting with our system by either picking a canned ML model from a roster or by uploading a Python script defining a custom ML model using the UI shown in Figure 2 (A). We currently support 4 popular DL models in our roster: ResNet50, MobileNet, BERT-base, and DistilBert. New models can be easily added to the roster. Also, the custom script option can support arbitrary Keras models. After picking a model, the user will be then prompted with the UI shown in Figure 2 (B) to specify a name, description, AutoML search procedure, names of features and label columns, the path to the training data, and the maximum number of training epochs for any model. If a custom script is uploaded, the user is required to specify the entry point function name in that script. This entry point function should take a dictionary of config values as input and return a compiled Keras model as output. The user is also required to specify the search spaces for the available configs. The list of available configs is fixed for a canned model. For a custom model, it can be defined manually. After specifying these values, the user can launch the model selection workload. While the workload is running, the user can visualize model training and validation metrics, such as loss and accuracy, through an embedded TensorBoard UI as shown in Figure 2 (C). User can also add/stop/resume configs using the controls shown in Figure 2 (D) or create a new drill-down workload on a refined search space using the UI shown in Figure 2 (E).

2.3 Decoupled System Architecture

Our paradigm translates to two key system design decisions: (1) *decoupling* the specification of what configs to explore from scheduling their training and (2) being able to *multiplex* the training of many configs on the fly on the same cluster. Otherwise, it is simply not possible to run multiple model selection workloads at the same time or even increase the model selection throughput of a single workload without provisioning more resources. While resource provisioning has become easy with cloud computing, cloud users also often need to limit their resource usages due to cost concerns. For others like domain science users, it may be simply not possible to provision more resources such as in fixed-sized campus clusters.

We implement our paradigm in Cerebro. Cerebro uses a novel parallel execution strategy for stochastic gradient descent (SGD)-based training (e.g., like in DL) called *model hopper parallelism* (MOP) that ensures SGD properties. SGD reads training data *sequentially* and repeats it for several iterations. A single iteration is also called an *epoch* of training. MOP breaks a single epoch of training on partitioned data into multiple sub-units called *sub-epochs*; a sub-epoch operates on a single data partition. Given a set of configs, MOP schedules them using an *epoch-level* scheduling template where all configs are trained for the current epoch before training the next epoch for any config. Also, it multiplexes the training of the configs by asynchronously scheduling sub-epochs on workers.



Intermittent Human-in-the-Loop Cerebro

Upload a model script file OR choose a popular model

Drag and Drop or Select a File

Select a popular model: ResNet-50

Select this model

Setup experiment

Name of experiment (string type)	ResNet-
Description (optional)	Using ResNet to
Max number of models for random and autoML (integer type)	100
Name of features (string type, comma seperated)	feature_columns:(strir
Name of label (string type)	label_columns:(string)
Maximum training epochs (integer type)	max_train_epochs:(int
Data store prefix path (string type)	data_store_prefix_patl
Estimator function name (<module_name>:<function_name>)	Do not need
Hyperparameters search strategy (please select)	GridSearch

Hyperparameter grid table:

	Name	Hyperparameter Type	Choices	Min Value	Max Value	Quantum	Dat
	filter data...						
×	Learning Rate	log uniform	None	-4	1	None	flo
×	Batch Size	quantum uniform	None	16	256	16	int

Models

Input model name and operate on the

Model Name

Resume Delete Stop Clone

Model Name	Experiment Name	Paren
filter data...		
Michael	ResNet-Example	/
Christopher	ResNet-Example	/

Clone Model

Base Model Name: Michael

[✓ Warm Start on Base Model] Start From Scratch

Name	Hyperparameter Type	Choices	Min Value	Max Value	Quantum	Data Typ
Learning Rate	log uniform	None	-3	-1	0.5	float
Batch Size	quantum uniform	None	64	64	0	integer
Drop Out	uniform	None	0.1	0.1	None	float

Figure 2: User interface for intermittent human-in-the-loop model selection. (A) UI to either pick a canned ML model (e.g., ResNet50) or upload a script file defining a custom model. (B) UI to specify experiment metadata, training data information, and configuration search space. (C) Visualizing the learning curves using embedded TensorBoard. (D) UI listing all configs and controls to add/stop/resume configs. (E) UI to create a drill-down model selection workload.

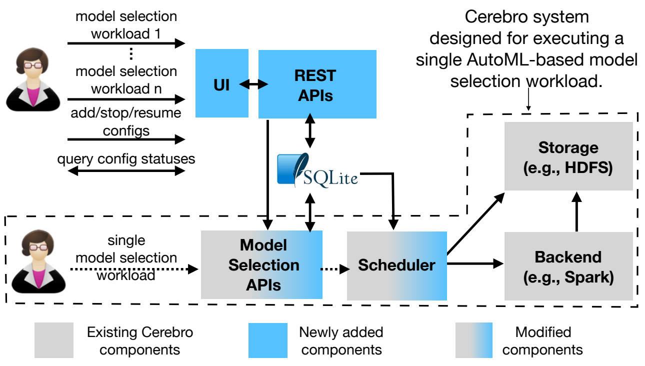

model selection workload 1
⋮
model selection workload n
add/stop/resume configs
query config statuses
UI
REST APIs
SQLite
Cerebro system designed for executing a single AutoML-based model selection workload.
Storage (e.g., HDFS)
single model selection workload
Model Selection APIs
Scheduler
Backend (e.g., Spark)
Existing Cerebro components
Newly added components
Modified components

Figure 3: High-level system architecture diagram of Cerebro along with the changes/additions to support our intermittent human-in-the-loop model selection paradigm.

The scheduler ensures that both a config is trained sequentially and on all partitions. Overall, MOP significantly increases the model selection throughput without provisioning more resources. Originally, Cerebro was designed to execute a single AutoML-based model selection workload at a time. Figure 3 presents Cerebro system architecture. More details about the Cerebro system can be found in our VLDB 2020 paper [11].

We leverage the epoch-level scheduling template of Cerebro to support our new paradigm. We also add a new graphical user interface (UI), a REST API and update Cerebro's model selection APIs and scheduler to achieve our requirements. UI sends user requests to the model selection APIs through the REST API. We changed the model selection APIs such that they now write the configs to an SQLite database instead of directly interacting with the scheduler. User-created configs are also directly added to this database. The scheduler will then read all the configs that need to be trained from this database and train them for one epoch. After completing training for one epoch it will update the training metrics of the config in the database. And this process will continue. Whenever the user wants to stop (resp. resume) a config, it will be marked as such in the database and will be ignored (resp. considered back) by the scheduler for training. Figure 3 presents the system architecture of the modified Cerebro system.

3 DEMONSTRATION

3.1 Datasets and Workloads

In this demonstration we will enable the user to perform intermittent human-in-the-loop model selection on 5 real-world workloads: ResNet50 and MobileNet fine-tuning for NIH Chest X-Ray images dataset, BERT-base, and DistilBert fine-tuning for Yelp Reviews dataset, and an MLP training on Criteo click-through rate prediction dataset. Models for image and text workloads will be available in the system's model roster. For MLP, we will use the custom model option. The user will be able to pick either Grid Search, Random Search, or HyperOpt as the AutoML procedure. We will be running Cerebro with Spark backend and HDFS storage medium.

We assume the training and validation data are pre-processed and already available on HDFS, which is a reasonable assumption for many real-world ML model selection workloads. Currently, we support Petastorm data format, a widely used data format to store training data for DL training. More details on how to generate training data in Petastorm format can be found in Cerebro system documentation [1]. Also, we will use samples from the above datasets as real-world DL workloads intermittently span hours to even days, while the VLDB Demo slot is only an hour or so.

3.2 Walkthrough

Participants will be first made familiar with the intermittent human-in-the-loop model selection paradigm, Cerebro system architecture, and MOP using a supporting slide deck. This introduction will give the participants the necessary knowledge to understand and appreciate our technical contributions. Participants will be demonstrated 3 scenarios.

Scenario 1: Add/Stop/Resume Configs. We will demonstrate intermittent human-in-the-loop model selection using Grid Search as the AutoML procedure. We will pick a dataset and a canned DL model from our roster. We will then launch the workload after specifying the required information and defining the config search space. Since we pick a canned DL model the available config values are pre-determined and we will only need to specify the search space for those values. Instead of passively waiting until the workload finishes, we will demonstrate how we can use the model training and validation metrics available in the embedded TensorBoard UI to manually stop any non-promising configs, add new configs that we think will be better based on the results of completed configs, clone a config to run it further with a changed learning rate or batch size, and also resume any stopped config.

Scenario 2: Custom DL Model. This is similar to the first scenario except here we will use a custom DL model by uploading a script file. We will demonstrate how to define a custom model using Keras and how to define the config search space for the custom model that it depends on. We will explain how we can use this approach to perform a model selection workload over any config search space including architecture tuning such as tuning the number of layers and the number of neurons, and hyper-parameter tuning such as tuning the learning rate.

Scenario 3: Drill-down Model Selection. Here we will demonstrate how to launch another drill-down model selection workload on a refined search space as part of a parent model selection workload. This is useful when the user wants to explore several configs closer to a promising config in one go, instead of manually adding one config at a time. We will pick a dataset and a canned DL model from our roster and launch a model selection workload using HyperOpt AutoML procedure. Instead of passively waiting and allowing HyperOpt to explore the search space, we will demonstrate how we can launch a drill-down model selection workload on a refined search space based on human intuition.

Finally, participants will be able to use a hosted version of our system. They will be able to pick a dataset and a model from the roster or define a custom model and perform *intermittent human-in-the-loop model selection.*

4 RELATED WORK

While a few commercial software products including Sagemaker Autopilot, Azure Automated ML, and Determined AI have attempted to streamline AutoML-based model selection, unlike our paradigm, none of these products enable the user to steer a model selection process while it is running. Several other works [6, 8, 15] have also emphasized the importance of more human control in AutoML-based model selection. However, to the best of our knowledge, ours is the first system prototype that enables the user to intervene and steer the model selection process on par with the meta-heuristic while it is running. We also provide details of a system architecture to realize this new paradigm. Ideas similar to our stop and resume-based model training reprioritization approach have been explored in relational query processing settings [12, 13]. Our work was inspired in part by such ideas but ours is the first to apply them in the context of ML model selection workloads, with the main novelty here being our focus on iterative training procedures such as SGD and intervention by observing evolving learning curves.

5 FUTURE RESEARCH DIRECTIONS

First, much work is needed in formalizing this new paradigm and developing new AutoML procedures that explicitly take advantage of human input. Alas, almost all AutoML procedures today are developed without any human interactivity in mind. Some even make assumptions that prohibit human interaction [10]. Second, it would be appealing to add elastic scaling and cloud-native scheduling support to Cerebro, so that it can reduce runtimes subject to monetary constraints. Finally, Cerebro's decoupled architecture can be generalized to support multi-tenancy. With multi-tenancy, multiple concurrent model selection workloads on different datasets can be run on the same infrastructure.

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 a gift from VMware. The content is solely the responsibility of the authors and does not necessarily represent the views of any of these organizations. We thank the members of UC San Diego's Database Lab and Center for Networked Systems, Loki Natarajan and our public health collaborators at UC San Diego, Frank McQuillan and the Apache MADlib/Greenplum team at VMware, Joe Hellerstein, Chris Jermaine, Sam Madden, Sebastian Schelter, and Dan Suciu for their feedback on this work and/or this paper.

REFERENCES

[1] Accessed February 28, 2021. Cerebro Documentation. https://adalabucsd.github.io/cerebro-system/.
[2] James Bergstra et al. 2013. HyperOpt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. In *Proceedings of the 12th Python in Science Conference*, Vol. 13. Citeseer, 20.
[3] Xavier Bouthillier and Gaël Varoquaux. 2020. *Survey of Machine-Learning Experimental Methods at NeurIPS2019 and ICLR2020.* Ph.D. Dissertation. Inria Saclay Ile de France.
[4] Chengliang Chai and Guoliang Li. 2020. Human-in-the-loop Techniques in Machine Learning. *Data Engineering* (2020), 37.
[5] Max Jaderberg et al. 2017. Population Based Training of Neural Networks. *CoRR* abs/1711.09846 (2017). arXiv:1711.09846 http://arxiv.org/abs/1711.09846
[6] Arun Kumar et al. 2016. Model selection management systems: The next frontier of advanced analytics. *ACM SIGMOD Record* 44, 4 (2016), 17–22.
[7] Arun Kumar et al. 2021. Cerebro: A Layered Data Platform for Scalable Deep Learning. In *CIDR.* http://cidrdb.org/cidr2021/papers/cidr2021_paper25.pdf
[8] Doris Jung Lin Lee et al. 2019. A Human-in-the-loop Perspective on AutoML: Milestones and the Road Ahead. *IEEE Data Eng. Bull.* 42, 2 (2019), 59–70. http://sites.computer.org/debull/A19june/p59.pdf
[9] Liam Li et al. 2020. Massively Parallel Hyperparameter Tuning. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020.* mlsys.org. https://proceedings.mlsys.org/book/303.pdf

[10] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR* 18, 1 (2017), 6765–6816.
[11] Supun Nakandala et al. 2020. Cerebro: A Data System for Optimized Deep Learning Model Selection. *Proc. VLDB Endow.* 13, 12 (July 2020), 2159–2173. https://doi.org/10.14778/3407790.3407816
[12] Sivaramakrishnan Narayanan and Florian Waas. 2011. Dynamic Prioritization of Database Queries. In *ICDE*. IEEE, 1232–1241.
[13] Johannes Wust et al. 2013. Dynamic Query Prioritization for In-memory Databases. In *In Memory Data Management and Analysis*. Springer, 56–68.
[14] Doris Xin et al. 2018. Helix: Holistic Optimization for Accelerating Iterative Machine Learning. *Proc. VLDB Endow.* 12, 4 (2018), 446–460. https://doi.org/10.14778/3297753.3297763
[15] Ce Zhang et al. 2016. Materialization Optimizations for Feature Selection Workloads. *ACM Transactions on Database Systems (TODS)* 41, 1 (2016), 1–32.