Measuring, Quantifying, and Predicting the Cost-Accuracy Tradeoff

Matt Baughman, Nifesh Chakubaji
Department of Computer Science
University of Chicago
Chicago, USA
{mbaughman, nifesh}@uchicago.edu

Hong-Linh Truong, Krists Kreics
Department of Computer Science
Aalto University
Espoo, Finland
{linh.truong, krists.kreics}@aalto.fi

Kyle Chard, Ian Foster
Department of CS & DSL Division
UChicago & Argonne Nat Lab
Chicago, USA
{chard, foster}@uchicago.edu

***Abstract*—Exponentially increasing data volumes, coupled with new modes of analysis have created significant new opportunities for data scientists. However, the stochastic nature of many data science techniques results in tradeoffs between costs and accuracy. For example, machine learning algorithms can be trained iteratively and indefinitely with diminishing returns in terms of accuracy. In this paper we explore the cost-accuracy tradeoff through three representative examples: we vary the number of models in an ensemble, the number of epochs used to train a machine learning model, and the amount of data used to train a machine learning model. We highlight the feasibility and benefits of being able to measure, quantify, and predict cost-accuracy tradeoffs by demonstrating the presence and usability of these tradeoffs in two different case studies.**

***Index Terms*—performance, optimization, big data analytics, machine learning, quality of analytics**

I. Introduction

Data science analyses are often more continuous than they are discrete. That is, due to their stochastic nature, the results of an analysis may change over time and potentially they can be improved with additional computation or data. For example, when training a neural network model, the accuracy of the model is often dependent on the amount and quality of training data combined with the length of training time. This situation demonstrates one opportunity to dynamically measure and manage the tradeoff between cost (e.g., execution time, resource consumption, monetary cost) and performance (e.g., accuracy). For our analyses, we focus on the tradeoff between execution time (cost) and accuracy (performance) as the most common example observed in data science.

Machine learning practitioners have long faced the challenge of determining when a model is sufficiently well-trained and often they observe diminishing returns (in terms of accuracy) when training models. Thus, practitioners are faced with the decision between favoring execution time or accuracy, or some mix of the two. Further, this tradeoff is application-dependent, and thus there is no general-purpose approach for measuring and optimally balancing the tradeoff. Often, the decision is based on the use of thresholds. For example, when a model achieves a specified accuracy, exceeds an allowable duration, or fails to demonstrate marginal improvements.

While it is desirable to be able to make decisions that balance tradeoffs between expediency and accuracy, doing so is a significant challenge for two reasons. First, there is no general way to dynamically capture and optimize for cost and accuracy; and second; there is currently no commonly-accepted mechanism for describing and implementing objectives in such a space. Our goal in this paper is to suggest a roadmap for managing this general type of cost-accuracy tradeoff through initial experiments with real-world applications.

In this paper, we first outline the need for managing the cost-accuracy tradeoff by describing three illustrative examples in Section II—ensemble modeling, iterative machine learning training, and varying the amount of data used to train a machine learning model. In Section III we then explore these tradeoffs by varying optimization dimensions for each example and highlight the significant differences in our cost metric—time—and accuracy. In Section IV, we describe a research roadmap for measuring, quantifying, and predicting cost-accuracy tradeoffs. Finally, we present related work in Section V and summarize our contributions in Section VI.

II. Application Scenarios

Many applications are able to trade result quality for expenditure of resources or time needed to compute a result. For example, image processing, machine learning, and iterative simulations allow developers to modify configurations to increase accuracy at the expense of compute time. While methods exist for approximating functions (e.g., loop proliferation, task skipping, and data sampling) a frequent obstacle to the effective practical use of such methods is that users lack the expertise required to estimate how much computation is required to achieve a desired level of accuracy for a particular problem. Here we outline three representative analysis examples of which tradeoffs between compute time and accuracy are important.

A. Ensemble models

In many domains, many alternative applications are designed to perform similar tasks but in different ways. In these cases, each application exhibits different performance characteristics and costs. Often, to improve result quality data scientists employ ensemble-based approaches that allow the use of voting mechanisms to either increase precision or recall.

One example is the variant calling analysis of genomes which aims to identify single nucleotide variants within an individual genome relative to the population at large. There are dozens of interchangeable variant callers, with no scientific consensus on the "best" variant caller for a given situation. Each variant caller has vastly different performance characteristics, with execution time ranging from minutes to hours. It has become common practice for repositories, such as the Genomic Data Commons (GDC), to release variant datasets processed with multiple variant callers (e.g., GDC releases datasets processed with four variant callers). Similarly, it is common for researchers to combine results from several variant callers using ensemble-based approaches to improve accuracy [1].

To illustrate these opportunities, we refer to the use of ensemble-based methods by combining data from four well-known variant callers using data shared via the GDC [2]. Figure 1 shows the disjoint set of variants identified by the four variant callers on the aggregate of breast cancer (BRCA) samples. Each of the variant calling pipelines executed for between 10 and 500 minutes. These results highlight not only the potential benefits of ensemble approaches for improving precision and/or recall, but also that, in some cases, little benefit is to be gained by applying some variant callers, thereby establishing our tradeoff.

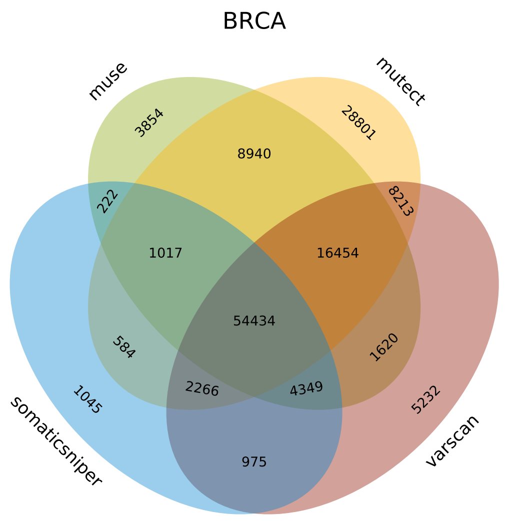

Fig. 1. Venn diagram for each variant caller for BRCA (from Leung et al. [2])

B. Iterative machine learning training

There is a general class of scientific applications including machine learning and monte carlo simulations that rely on iterative statistical methods. These methods essentially refine a model over discrete steps toward some objective function. The number of phases used may be varied depending on resource availability, objective, and the general type of learning algorithm.

One specific example to illustrate the properties of this type of algorithm is a simple, two-layer dense network for the classification of handwritten digits. Figure 2 demonstrates the relationship between time and loss (the performance metric of this model) present during neural network training and, more broadly, stochastic machine learning training. This exhibits a nonlinear tradeoff, given the reduction of marginal accuracy improvements over the course of training, as one would expect.

C. Training data for machine learning

The training time of machine learning algorithms can be influenced significantly by the amount of data used for training. However, it is not evident in all cases that more data is necessarily always better. Many aspects affect whether this may be true, including the uniqueness of the inputs, the breadth of possibilities the algorithm is expected to learn, and the overall performance and learning method of the algorithm itself.

In the case of simple image classifiers, it is reasonable that, of the training data, half of the images of one category may be representative of the other half. Following, this may be true of 25% of the data, or perhaps even 10%. In addition to the execution duration of the learning algorithm, the quantity of data affects questions of data sufficiency, particularly in machine learning applications that are being pioneered for the first time or are iteratively updated. How much data is enough data? How much accuracy will be gained by retraining with slightly more or different data?

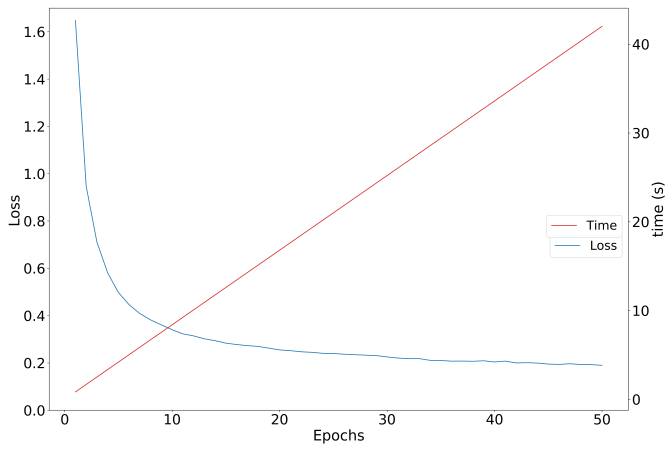

Fig. 2. Time and loss ("performance") of neural network training over many iterations.

III. Practical evaluations of cost-accuracy tradeoffs

To explore the potential benefits of quantifying and predicting cost-accuracy tradeoffs and common methods, we review two target use cases. We first explore the training of machine learning models by varying the number of iterations and quantity of training data available. We then explore the exploitation of domain knowledge to intelligently select data to be used for training as part of the preprocessing stages of a retail forecasting pipeline.

A. Deep Learning case study

As described above, deep learning is a perfect test case for computational tradeoffs with respect to time and accuracy due to the flexible nature of deep learning training. The tradeoffs we examine here are two forms of time vs. accuracy as determined by a) the number of samples from the data or b) the number of iterations, or *epochs*, we use to perform the training.

1) Methods: To investigate the role of tradeoffs in deep learning, we have implemented and trained three different image classifiers for the CIFAR-10 and CIFAR-100 datasets. The first two models consisted of three convolutional layers with 32, 32, and 64 kernels respectively and two fully connected layers with 512 units and then 10 or 100 units (for CIFAR-10 and -100 respectively). The third model was trained solely on CIFAR-100 and consisted of 6 convolutional layers with 2x128, 2x256, and 2x512 kernels per layer and two fully connected dense layers with 1000 and 100 units. For each of these models, we trained on 20%, 40%, 60%, 80%, and 100% of the available data, which we randomly selected in two ways: 1) selecting randomly from across the entire dataset and 2) selecting evenly from each class. We repeated each of these tests ten times for rigor.

For training time context (which is meant only to function as an initial example), we conducted all of these tests on a consumer-grade desktop with a latest-generation 6-core Intel CPU clocked at 3.9GHz, a Nvidia RTX 2070 Super, and 32GB of RAM. All data was stored locally on an NVMe SSD.

2) Evaluation of tradeoff results: In our experiments, we found some results that applied across all of our experiments and some that were more localized. Across all of our experiments, we observed that the difference between training and validation accuracy does not begin to diverge until after 40 epochs regardless of the proportion of data used (see Figure 3) and that, for all proportions of data, validation accuracy increases with respect to 100% data usage as the number of iterations increases (see Figure 4).

However, another result we saw from training our models is that accuracy per hour (time) seems always to favor the higher data percentages. On the other hand, maximizing this accuracy per hour measurement definitively will favor early halting of the algorithm, though this is to be expected. In other words, we have found that training a model on more data for a smaller number of iterations will lead to higher validation accuracies than training a smaller quantity of data for more epochs (see Figure 5).

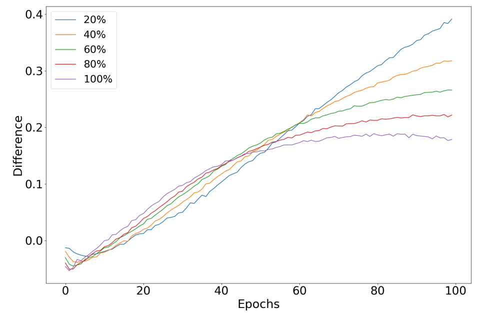

Fig. 3. Difference in accuracy during training between different proportions of training data.

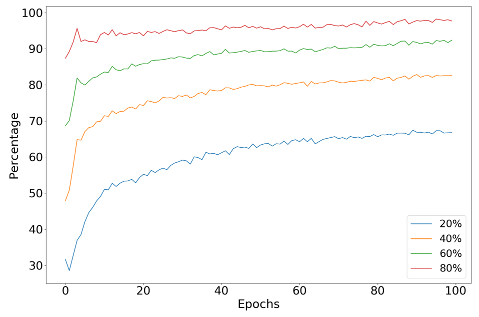

Fig. 4. The proportional accuracy by each data percentage over time with respect to using all data in the training set.

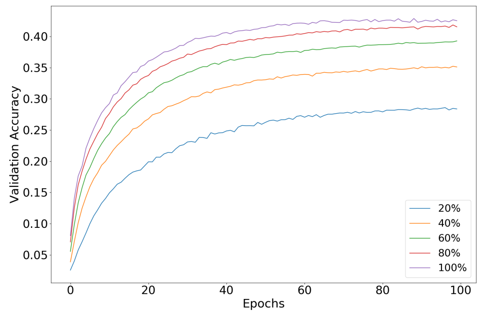

Fig. 5. Validation accuracy of different data proportions over training time.

B. Retail forecast pipelines

Production data analytics pipelines, in general, are comprised of many phases, such as data collection, data preprocessing, training and analytics/machine learning. Each phase has different inputs and outputs that strongly influence the other phases. Further, in production machine learning applications training is conducted on real data and pipelines are designed to produce results for particular purposes. In practice, the time to obtain results is often one of the primary evaluation goals. In this context, we consider the time-accuracy tradeoff for a retail prediction pipeline. The key question is how can we optimize the training time while still being able to produce high-quality results for forecast models. Specifically, in our case study, the domain experts and companies doing retail forecasting look for different ways to determine the tradeoffs between elapsed time, costs and quality of the models.

1) Methods: We first define tradeoff quality for pipelines and their components based on phases: tradeoffs are based on time and quality of the results. We further divide the quality of data aspects into many sub-dimensions based on the specific data analytics/ML pipelines. Here, the approach is to *allow owners of pipelines to define possible tradeoffs* based on their respective domains and to provide evaluation functions for evaluating the quality of different result characteristics.

We use a synthetic retail dataset that describes item promotions and sales in stores over a period of 3 years. We generated two datasets, based on real-world retail data from a collaborating company, containing 500 000 and 1 000 000 rows of sales data. Three sample rows from the dataset can be seen in Table III-B1. In the first test, we examine two cases: tradeoffs between resources for cost and for elapsed time: (i) very high capacity resources with higher costs and (ii) limited capacity for lower costs but need more execution time. We tested with using memory size of computing instances because data analysis tasks have strong requirements of storage for swaps and computations.

We apply a representative forecasting pipelines that comprises a set of models that can be queried to make predictions about a specific location and item. This means that different models are used for each location and item pair. One way to maintain the best possible accuracy per model is to filter out certain locations, items, or time periods.

Our pipeline consists of four different tasks. First, data cleaning removes faulty rows and calculates the sales for each item from raw data. Second, a data transformation step formats the different categorical values so that they can be used in the algorithm. Third, a linear regression step runs over the data and creates a prediction model. Finally, a scoring task scores the resulting model and expresses the quality as the `R` squared score. We perform the forecasts and training using ML pipelines executed in the cloud, for example, in our case, using Apache Airflow in Amazon EC2.

To explore the effect of data reduction we apply two application-specific strategies. The first reduces the number of store locations that are taken into account when training the model. As our dataset contains information about stores from two different cities, we filter the results to one city, thus decreasing the row count twofold. The second uses data from only the last 12 months. This effectively removes the first 24 months of data and decreases the row count by a factor of three. If we apply both reductions, we reduce the number of rows by a factor of six.

We monitored the following metrics:

- *data size*: is based on row count. We can reduce the data size by reducing the number of rows used for training.
- *`R` squared value*: is a suitable metric for linear regression algorithms used in retail forecasting. An acceptable `R` squared value largely depends on specific use cases and tasks. We use the guidelines of the R squared value from [3] which is suitable for marketing research.
- *elapsed time*: is expressed in seconds and calculated on a task per task basis.
- *cost*: is the running cost of a task spent in Amazon EC2.

2) Evaluation of tradeoff results: The results of the pipeline with and without data reduction are shown in Table III-B2. For the dataset with 500 000 rows, the accuracy of the pipeline after reductions was very similar to the regular pipeline without these reductions. However, the reduced pipeline was approximately 40% faster.

We observed a real improvement for this strategy with the larger dataset of 1 million rows. With reduced data, we observe the total accuracy 13.3% lower and the elapsed time was 44% shorter. This improvement came with a cost towards the quality of results. The R squared value was 9.5% lower at 77% and the row count was 6 times smaller. This means that with a lowered cost we could get good enough results for 50% of the total store locations.

Overall the application-aware data reduction strategy proved effective and shows that *cost-accuracy tradeoffs may be more intelligently made based on knowledge of the application domain*.

IV. Discussion and Research Roadmap

We first summarize the implications of our results and then outline the future directions of our research.

A. Discussion

We see above that there is a definite diminishing value in the iterative training of neural networks, as well as the diminishing value in data addition. One prime area of interest regarding this work is its applications in efficient and intelligent neural architecture search. As new problem domains and sciences begin exploring the potential applications of deep learning within their respective fields of study, new types and styles of neural networks will be necessary to serve as adequate solutions to these new problems.

For example, in evaluating model performance, we developed a simple decay model to predict the training loss of the model given different quantities of “input” epochs (see Fig 6. From this simple model, we could easily create a simple tool that could, given a certain objective confidence level, act as an early stopping condition for evaluating how well a model may perform. We show this potential in Fig. 7 which illustrates the mean predictive error of our loss model as a function of the number of epochs used for training.

Additionally, by integrating inter-iteration weight sharing [4], we could use these partially trained networks to achieve even greater accuracy, as well as increased advantages in execution duration.

B. Future Directions

We briefly outline our future research directions based on our preliminary results described above.

1) Quality of Analytics (QoA): We suggest that a first step is to create a conceptual framework for defining Quality of Analytics (QoA). This framework should allow us to define functions for evaluating attributes that are specific to domains such that other domain-independent attributes like time and cost can be automatically determined. It should also be flexible, such that different quality measures can be integrated.

TABLE I
THREE SAMPLE ROWS FROM RETAIL DATASET (SIMPLIFIED COLUMN NAMES)

date	id	name	volume	price	cost	promo	category_net	margin	category1	category2	location	sales
07/01/2018	100	Chicken	38144.0	3.79	2.7	0	451692.0	0.25	Meat	Food	Helsinki	144565.76
14/01/2018	100	Chicken	36420.0	3.79	2.66	0	414342.0	0.25	Meat	Food	Helsinki	138031.8
21/01/2018	100	Chicken	35322.0	3.79	2.66	0	381854.0	0.25	Meat	Food	Helsinki	133870.38

TABLE II
COMPARISON OF DATA REDUCTIONS IN PIPELINE TRAINING.

	500k dataset		1m dataset	
Data Used	500k	83,322	1m	833,322
Time (s)	1185	725	2130	1200

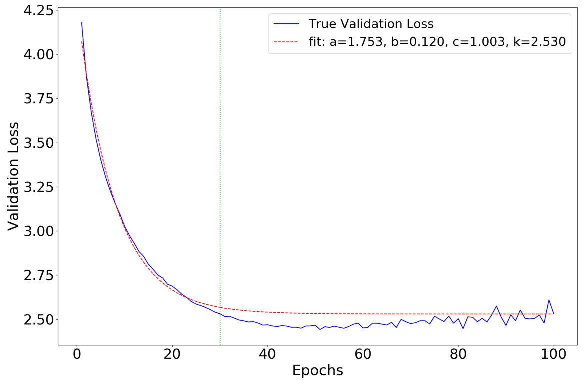

Fig. 6. Predicted loss after 30 epochs and observed loss for training our CIFAR-100 model.

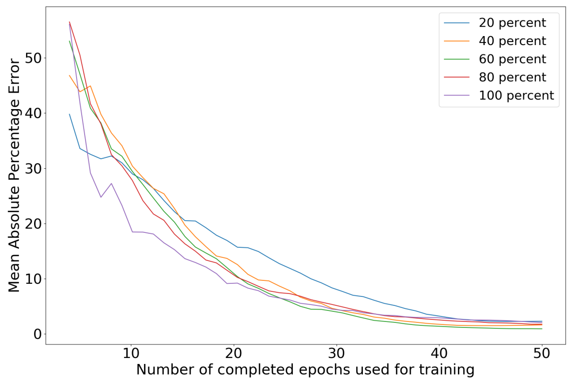

Fig. 7. Mean absolute percentage error of our loss prediction model given a number of epochs.

This way will allow us to incorporate knowledge from domain experts into tradeoff analytics.

2) Monitoring and mechanisms for measuring QoA: We need to develop systems for measuring accuracy and costs. That could be done by leveraging existing monitoring tools. However, there may well be other features that need to be monitored (e.g., the sensitivity of data used to train a machine learning model). Further, capturing QoA through a multi-analysis pipeline may require new methods of measurement from individual components. We need to address mechanisms for capturing/quantifying measurements that can be used for evaluating domain-specific attributes like accuracy in different domains. Due to the complexity of domain-specific attributes that need to invoke domain experts, we should allow annotation and instrumentation techniques that allow experts to provide input to measurements.

3) Models for predicting QoA: One of the biggest areas of research is to develop new models that are capable of predicting cost and accuracy, from a variety of application types and domains, and in different environments (different configurations, parameters, hardware, accelerators, etc.). There is extensive work on predicting various aspects of this problem, and thus, we focus on the reuse of existing methods and the integration of diverse features.

4) Methods for adaptation and optimization: We need to enable users to explore cost/accuracy tradeoffs such that they can inform application development and use. Furthermore, methods for enabling users to state cost/accuracy goals in programs should be allowed to trigger adaptation actions. From the system viewpoint, to implement possible actions for changing pipelines, we should define primitive action models, in which actions can be mapped to tradeoff conditions. Initial work on primitive action models [5] could be a starting point.

5) Generalization with Neural Architecture Search Integration: As mentioned above, one key area we plan to explore is in the area of neural architecture search (NAS). The tradeoffs between required data, predicted model performance, accuracy, and compute cost covered in this paper represent a significantly complex optimization space where the optimization objectives of an end-user could impactfully manifest in the execution of an integrated NAS pipeline.

6) Integration with cost-aware computing work: Our prior work has focused strongly on different approaches to cost-aware and distributed computing, both in traditional HPC environments and in the cloud. Specific examples include the prediction of cloud costs given workflow execution duration [6], cost-aware cloud provisioning [7], prediction and optimization of execution characteristics in the cloud, and automating efficient experimental design of workflow parameterizations [8].

V. RELATED WORK

Balancing the cost of computing has been an important area of investigation in fields such as IoT and power-aware computing where limited resources make consideration of cost more important. In IoT for example, edge devices may screen for redundancies in information and transfer only selective or aggregate data. In power-aware scenarios, developers may choose to lower precision or underutilize cores in an effort to reduce power consumption.

Machine learning has motivated more recent work in exploiting cost-accuracy tradeoffs. For example, researchers have shown significant advantages when training neural networks by not unnecessarily retraining an algorithm over the same data [9]. Additional methods of machine learning and training optimizations—such as model pruning [10], hyperparameter selection [11], and using hierarchical, ensemble models [12]—also provide opportunities to explore these tradeoffs. The most common approach for reducing costs when training models is via reduction of training data, for example by applying selection criteria to obtain representative and diverse training data [13], [14]. Benchmarks have also been used to test and compare the cost and accuracy of ML models [15]. These methods have shown great promise for subsampling training data at each epoch, reducing computation time by an order of magnitude or more than brute force training.

Other works on the selection of compute resources [16] indicate the significant difference that intelligent selection of compute resources can make. Additionally, efficient black-box optimization of experiment parameters has been shown to significantly reduce the necessity of exhaustive experimentation regarding workflow configurations [8].

VI. Conclusions and Future Work

We have conducted a preliminary investigation of the benefits of balancing the cost-accuracy tradeoff in applications. Specifically, we focused on time and accuracy by illustrating how the number of methods applied in an ensemble model, the amount of training data used for machine learning, and the length of training for machine learning models can impact the cost and accuracy of the model.

To seek for common approaches to solving tradeoff problems, we developed and analyzed two specific case studies to illustrate these cost-accuracy tradeoffs in the context of machine learning problems. The first case study demonstrated how variations in the quantity of data used to train a common computer vision model alter both the training time and the training accuracy. Additionally, we show in this example that manually altering training duration also directly effects model accuracy. These tradeoffs are then illustrated as they relate to overall model accuracy, such as would be used in standard training, as well as to the predictability of model accuracy with respect to training duration. The second case study illustrates a manifestation of the cost-accuracy tradeoff in a production machine learning workflow development cycle by altering the quantity of data exposed to the workflow during training and evaluation. This real-world example of these tradeoffs highlights the extreme complexity of tradeoff optimization due to the extreme nonlinearity observed between execution duration and data availability.

In our future work, the integration of established cost-aware computing approaches with exposure to the cost-accuracy tradeoffs in machine learning demonstrated in this paper will allow for full-cycle cost-aware machine learning workflow that will allow for automated profiling and optimization in production environments.

Acknowledgments

This work was partially supported by the National Science Foundation (NSF: 1816611). We thank Teppo Luukkonen and Mikko Ervasti from Sellforte for their support in developing the retail forecast use case.

References

[1] V. Trubetskoy Et al., “Consensus Genotyper for Exome Sequencing (CGES): improving the quality of exome variant genotypes,” *Bioinformatics*, vol. 31, no. 2, pp. 187–193, 2014.

[2] K. Leung, M. Kimball, J. Pitt, A. Woodard, and K. Chard, “Walking the cost-accuracy tightrope: balancing trade-offs indata-intensive genomics,” 2019.

[3] J. F. Hair, C. M. Ringle, and M. Sarstedt, “Pls-sem: Indeed a silver bullet,” *Journal of Marketing Theory and Practice*, vol. 19, no. 2, pp. 139–152, 2011.

[4] M. Baughman and J. Wozniak, “Enhancing neural architecture search with inter-epoch crossover and speciation.” 2019.

[5] T. Nguyen, H. L. Truong, G. Copil, D. Le, D. Moldovan, and S. Dustdar, “On developing and operating of data elasticity management process,” in *Service-Oriented Computing - 13th International Conference, ICSOC 2015, Goa, India, November 16-19, 2015, Proceedings*, ser. Lecture Notes in Computer Science, A. Barros, D. Grigori, N. C. Narendra, and H. K. Dam, Eds., vol. 9435. Springer, 2015, pp. 105–119.

[6] M. Baughman, C. Haas, R. Wolski, I. Foster, and K. Chard, “Predicting amazon spot prices with lstm networks,” in *Proceedings of the 9th Workshop on Scientific Cloud Computing*. ACM, 2018, p. 1.

[7] R. Chard, K. Chard, R. Wolski, R. Madduri, B. Ng, K. Bubendorfer, and I. Foster, “Cost-aware cloud profiling, prediction, and provisioning as a service,” *IEEE Cloud Computing*, vol. 4, no. 4, pp. 48–59, 2017.

[8] C. Wu, T. Summer, Z. Li, A. Woodard, R. Chard, M. Baughman, Y. Babuji, K. Chard, J. Pitt, and I. Foster, “Paraopt: Automated application parameterization and optimization for the cloud,” in *CloudCom*, 2019.

[9] L. Ning, H. Guan, and X. Shen, “Adaptive deep reuse: Accelerating cnn training on the fly,” in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1538–1549.

[10] S. Lym, E. Choukse, S. Zangeneh, W. Wen, S. Sanghavi, and M. Erez, “Prunetrain: Fast neural network training by dynamic sparse model reconfiguration,” 2019.

[11] L. N. Smith and N. Topin, “Super-convergence: Very fast training of residual networks using large learning rates,” 2018.

[12] C. S. Miranda and F. J. Von Zuben, “Reducing the training time of neural networks by partitioning,” *arXiv preprint arXiv:1511.02954*, 2015.

[13] P. Zhu, D. A. E. Acar, N. Feng, P. Jain, and V. Saligrama, “Cost aware inference for iot devices,” in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 2770–2779.

[14] M. J. Van Grinsven, B. van Ginneken, C. B. Hoyng, T. Theelen, and C. I. Sánchez, “Fast convolutional neural network training using selective data sampling: Application to hemorrhage detection in color fundus images,” *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1273–1284, 2016.

[15] C. Coleman, D. Kang, D. Narayanan, L. Nardi, T. Zhao, J. Zhang, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, “Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark,” *SIGOPS Oper. Syst. Rev.*, vol. 53, no. 1, pp. 14–25, Jul. 2019.

[16] M. Baughman, R. Chard, L. T. Ward, J. Pitt, K. Chard, and I. T. Foster, “Profiling and predicting application performance on the cloud.” in *UCC*, 2018, pp. 21–30.