BRIDGING THE ML-HUMAN GAP IN SCIENTIFIC DATA NAVIGATION

by

Nicholas S Carey

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

August, 2020

Abstract

Off-the-shelf ML libraries combined with accessible scientific computing infrastructures continue to find new avenues for automation and augmentation of researcher work in the lab. Widely applicable pre-trained neural networks have greatly reduced the barrier of entry toward applying classification models, leaving the main challenge to be the translation of domain expert knowledge into machine intelligence. I have developed several specialized models solving specific lab problems with minimal training regimens by building atop published general-purpose frameworks. Applications include reinforcement-guided molecular dynamics simulations, human reaction-based dataset navigation through machine-readable P300 brain waves, and floating-zone furnace user guidance through classification of live boron-carbide crystal growth video feed. Evaluation of these purpose-built models constructed with limited, expensive training data is achieved in a combination of the established domain metrics with statistical techniques.

ABSTRACT

Primary Reader and Advisor: Tamas Budavari

Secondary Reader: David Elbert

Acknowledgments

I would like to acknowledge the SciServer team and the Institute for Data Intensive Engineering and Science (IDIES) at JHU for providing a platform upon which this thesis is built upon. Without the vision of Alex Szalay, Tamas Budavari, and David Elbert, this work would not have been possible. I'd also like to thank the PARADIM lab and Tyrel McQueen for supporting the work showcased in Chapter 4. The Army Research Lab's Center for Adaptive Soldier Technologies (CAST) provided the Brain-Machine Interface hardware that enabled the research shown in Chapter 5.

I acknowledge Sarana Nutanong, Yanif Ahmad, Alex Szalay, and Thomas B. Woolf as co-authors of the work shown in Chapter 2, Yanif Ahmad, Tamas Budavari, and Alex Szalay as co-authors of the work in Chapter 3, and Adam Phelan, Ali Rachidi, Connor Krill, Pheobe Appel, Jessica Zahn, Matthew Hudes, Brian Schuster, Tyrel M. McQueen and David Elbert as co-authors of the work shown in Chapter 4.

Contents

List of Tables

List of Figures

Chapter 1

Data Navigation

Rapid expanse of AI applications combined with wide-spread access to demographic and consumer data streams have revolutionized world-wide business practices. Similar opportunity has been realized in some scientific areas like astronomy and mapping the human genome, but hindered in others due to a lack of or inaccessibility to large amounts of pertinent data. Indeed, machine learning specifically has been widely touted as central to harnessing the data revolution (HDR) [1], but ML augmentation of workloads have not yet been realized in many wider scientific fields. In order to spread the data revolution, the tenets outlined in the National Science Foundation HDR initiative must be addressed and translated into solutions that show attainable advances in disciplines where tradition and working culture may block the availability of necessary resources for general ML

augmentation in the lab.

This dissertation focuses on interdisciplinary applications from materials sciences, molecular dynamics, distributed algorithms and human-computer interaction to develop methodologies bridging the ML-human gap, allowing rapid applied science advances by efficient use of limited data resources. Each chapter shows an example of a data navigation process; data is collected, stored, and linked to a computational analysis shown to a user whose feedback directs the next exploratory iteration. The work in this thesis seeks to accelerate the exploration loop and provide navigational tools to assist the user in finding relevant data views.

The solutions and models I've developed were largely platformed on common consumer hardware paired with freely available, generally applicable online computing frameworks. When first applying models, the procurement, development, and evaluation of training datasets specific to the task at hand is the most expensive and time-consuming process involved. Collection of relevant training data is particularly difficult when lab equipment access is restricted or expensive. While much of the libraries and hardware I used to develop ML solutions were off-the-shelf and freely available, the gathering of quantities of exemplar data from a floating zone furnace or a medical-grade electroencephalogram (EEG) headset had to be scheduled, organized, and linked with computing infrastructure. I developed dis-

tributed systems and interfaced libraries in order to collect, store, and analyse the data from these sensors. A central finding is the power of utilizing published datasets and publicly available generalized pre-trained models to greatly reduce training needs when adapting prototype models to new purpose, lowering the barrier-of-entry and raising the potential for automation. Chapters 4 and 5 are examples of work that would prove to be prohibitively expensive without the use of such pre-trained models.

With the exception of Chapter 2, the work in this dissertation was developed and prototyped on the SciServer [2] scientific computing platform, which was created as a part of the NSF Data Infrastructure Building Blocks [3] program. The SciServer infrastructure combines database, file storage, as well as collaborative scripting and computing resources under a single interface accessible through the browser. Software are container-ized, facilitating the configuration of libraries and the reproduction of computing environments. Most importantly, when applied to lab solutions SciServer provides an effective platform for realizing the NSF HDR objectives.

My work on the Materials in Extreme Dynamic Environments Data Science Cloud (MEDE-DSC) [4], built on top of SciServer, is one such example of tools built for harnessing the data revolution in new fields. MEDE-DSC combines data-science tools with collaborative data sharing, specifically tailored for the needs of the Materials Domain while addressing the

strategic goals of the Materials Genome Initiative [5]. As a part of MEDE-DSC, I developed a computing environment hosted in SciServer consisting of pre-installed data science tools combined with software popular among the interviewed materials scientists at JHU's HEMI and MEDE organizations. [6] Intended to bypass the set-up and configuration work inherent in employing data science tools in materials research, the MEDE-DSC platform eventually served as a springboard for prototyping the floating-zone furnace user-guidance model detailed in Chapter 4.

In summary, this thesis investigates methods meant to bring new, relevant data perspectives to analysts. Chapter 2 describes a molecular dynamics simulation platform, where the user is guided towards interesting protein interactions by the reinforcement learner managing the simulation. Chapter 3 details an extreme value summary statistics approach to a distributed top-k elements algorithm. These first two chapters contain distributed systems built to assist users searching a data field, while the final two chapters are focused on classifiers directly engaged in data exploration and navigation. Chapter 4 is an overview of a floating zone furnace classifier, directing the furnace operator in achieving a stable crystal growth. Finally, Chapter 5 explains a prototype brain-computer interface system for navigation of datasets based on human reactions.

Chapter 2

Adaptive Exploration for Large-Scale Protein Analysis in the Molecular Dynamics Database

2.1 Abstract

Molecular dynamics (MD) simulations generate detailed time-series data of all-atom motions. These simulations are leading users of the worlds most powerful supercomputers, and are standard bearers for a wide range of high performance computing (HPC) methods. However, MD data exploration

and analysis is in its infancy in terms of scalability, ease-of-use, and ultimately its ability to answer grand challenge science questions. This demonstration introduces the *Molecular Dynamics Database* (MDDB) project at Johns Hopkins, to study the co-design of database methods for deep on-the-fly exploratory MD analyses with HPC simulations. Data exploration in MD suffers from a human bottleneck, where the laborious administration of simulations leaves little room for domain experts to focus on tackling science questions. MDDB exploits the data-rich nature of MD simulations to provide adaptive control of the exploration process with machine learning techniques, specifically reinforcement learning (RL). We present MDDBs data and queries, architecture, and its use of RL methods. Our audience will cooperate with our steering algorithm and science partners, and witness MDDBs abilities to signicantly reduce exploration times and direct computation resources to where they best address science questions.

2.2 Introduction

Molecular dynamics (MD) simulation [7] is a powerful instrument for generating a detailed description of biomolecules (e.g., proteins, lipids, nucleic acids). MD uses a computationally intensive numerical integration in small femtosecond-scale steps to produce spatio-temporal (x,y,z) trajectories

of atoms positions. Trajectories are analyzed as a high-dimensional dataset in terms of the possible conformations (i.e., shapes and poses) that a protein may form.

MD trajectories present a challenging data management problem in that their processing and analysis pipelines are both I/O and CPU intensive. Modern MD simulators are amongst the leading GPU applications and can yield a throughput of 0.1 - 0.3 GB/s per GPU. The ensuing peta-scale dataset is then analyzed with machine learning techniques (e.g., specialized sampling, graphical models, SVMs, etc) to fit a high-dimensional function describing protein energetics. For a large 1-million atom protein system, MD simulations and analyses can take months to years of supercomputer time to generate a rich dataset for a large protein. Despite their cost, trajectory data remains extremely valuable for the simulation scientist. All of the most important biological connections, e.g. drug design, genetics, cancer screening, relate back to protein shapes and kinetics.

We are developing the *Molecular Dynamics Database* (MDDB) at JHU to provide scalable storage and indexing, parallel and incremental query processing, and rich in-database analysis of MD trajectories. MDDB will provide a public analysis service for the biophysics and biochemistry communities, and will require novel database methods for its users to ingest, explore, and extract insights from peta-scale trajectories. In addition to core

database design, our initial efforts on MDDB include multi-scale database view management [8,9], and as the focus of this demonstration, data-driven control and scheduling of expensive MD simulations for a closed-loop data exploration process.

MD simulations sample a proteins possible shapes (or *conformations*) by solving a set of dierential equations governing atom motions. Conformations are associated with free energies in an energy landscape which can be viewed as a *probability distribution* over conformations. Thus MD simulation can be seen as a computationally expensive *sampling algorithm* that can only be evaluated in batch sequential form. The ensuing probability distributions are extremely high-dimensional, sparse, and dicult to use in higher-level analyses such as protein-protein interactions.

A central biophysics task is to derive a Markov chain that compactly represents a proteins energy landscape and the probability distribution it denes. This task is solved by an iterative sampling and energy landscape exploration process. Each iteration consists of a trajectory production step, and trajectory analysis to both revise the Markov chain and to determine simulation parameters to perform further sampling and exploration. Traditionally in MD this is a low frequency, human-in-the-loop iteration. Simulations are run at long timescales that take weeks or months, and are followed by equally long manual analyses by domain experts with small-

scale scripting languages, and statistical packages designed for gigabyte-scale datasets.

MDDB provides a closed-loop in-database control abstraction for *high-frequency* iterative exploration that reduces the computational overheads of costly MD simulations and the experiment management burden. Our focus is on leveraging query processing techniques to best exploit large multi-terabyte trajectory datasets in controlling a sampling algorithm, guiding it to run only *interesting* simulations in *goal-driven* fashion. Our adaptive sampling rapidly constructs a Markov chain that can reproduce an energy landscape, and our abstraction empowers domain experts to specify *data-driven* simulation control mechanisms as *queries*.

Our adaptive control framework is based on reinforcement learning methods that learn control policies from a large dataset that is continually updated by short simulation runs. Our novel contributions include incorporating incremental ETL (Extract-Transform-Load) [10] and query processing methods for efficient data ingestion and scalable policy evaluation over large numbers of environment states and actions. We will showcase MDDBs highly adaptive and responsive MD controller and its greatly accelerated exploration of a complex high-dimensional space of protein structures.

2.3 MDDB Overview

2.3.1 Trajectory Datasets

MD trajectories are currently represented in two forms in MDDB. First, as described in the introduction, a trajectory can simply be stored as a time-series of 3D-coordinates for efficient ingestion. This format matches the direct outputs of MD simulators, providing a high data loading throughput. Our second format is analysis-oriented, where we store protein shapes (called *conformations*) as a sequence of phi-psi dihedral angles describing the protein backbone. Figure 1 illustrates a conformation of the *alanine dipeptide* protein described using two dihedral angles (ϕ, ψ). For biological reasons, dihedral angles marked (*) need not be stored.

Phi-psi sequences are considerably more compact than the raw format. Furthermore, we can exploit their translation and rotation invariant properties for query processing. Indeed, phi-psi sequences are popular low-level features when applying machine learning techniques to molecular datasets.

MDDB represents phi-psi sequences, and their concatenation into trajectories as columnar relations. We currently store all trajectories across multiple proteins in a single columnar relation, with selective decomposition into a feature-oriented star schema that stores information across all trajectories for a specic protein phi-psi sequence. MDDB can apply a wide

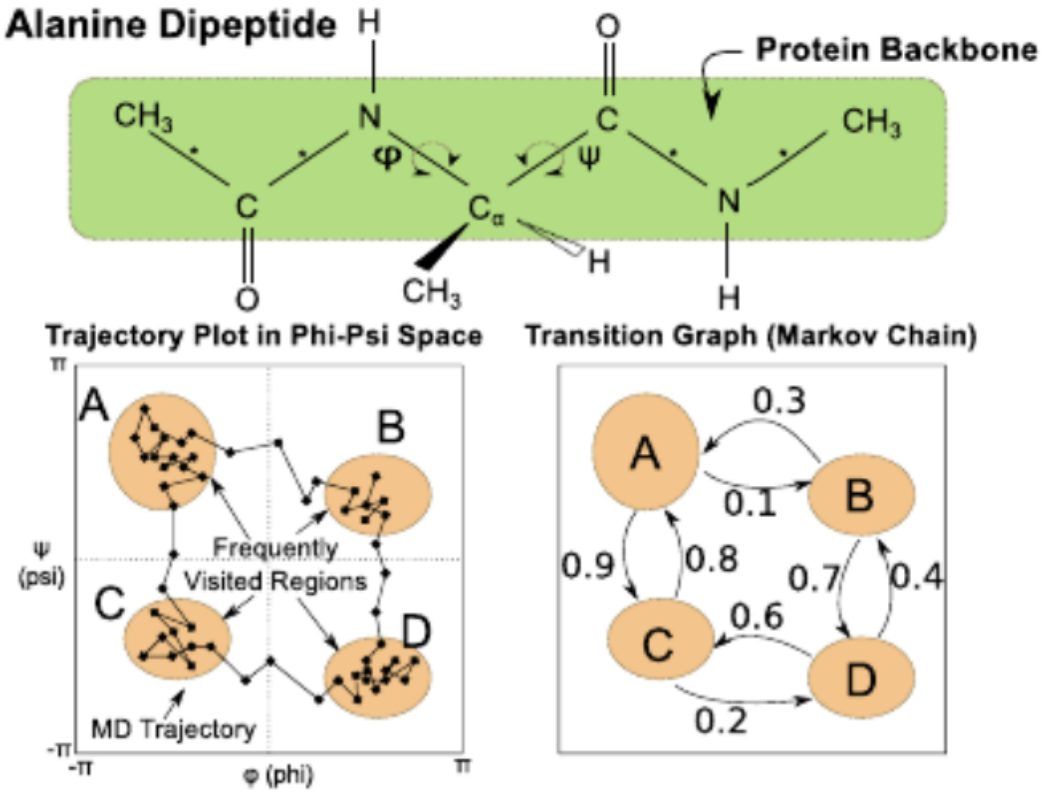

Figure 2.1: Protein conformations in phi-psi space.

range of analyses on phi-psi values, for example Figure 1 shows analyses that: (i) identify energy wells by clustering conformations frequently assumed by the protein; (ii) identify pathways (transitions) between these energy wells; (iii) empirically construct and evaluate Markov chains that encode transition probability distributions of protein conformations.

2.3.2 Query Workload

MDDB supports two substantially different categories of query workloads, one a natural fit for DBMS and a second that requires coupling in-database functionality with a range of external software components. The rst category consists of query templates that capture domain-specific ad-hoc and exploratory science questions, as used in traditional interactive sampling algorithms. These templates compose SPJAG queries, frequently apply geometric computations, and exhibit varying join degrees and nesting depth in terms of correlated subqueries. The second workload arises from data-dependent iterative algorithms that produce substantial quantities of derived and intermediate data while solving search problems, or in converging to a fixpoint or termination condition. Adaptive control and its use of reinforcement learning matches this workload pattern, as well as several analysis algorithms used internally in MDDB, such as k-means clustering, MCMC inference and replica exchange (a form of parallel tempering).

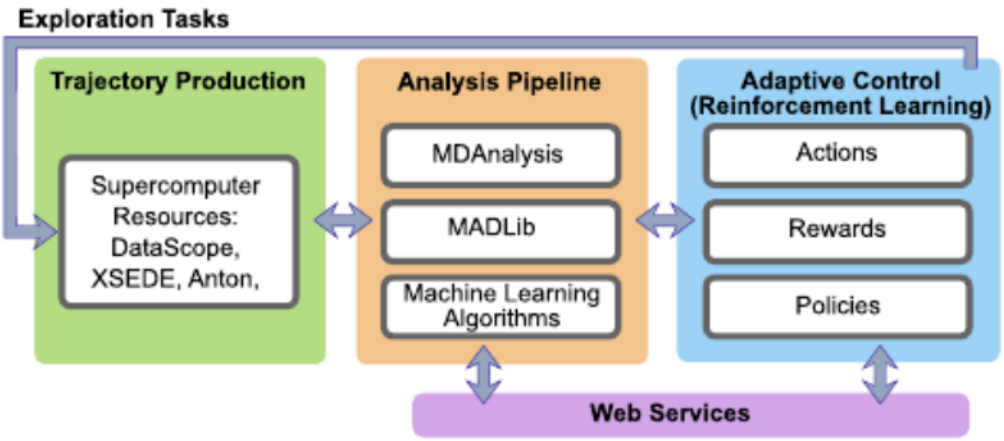

Figure 2.2: MDDB system architecture

2.3.3 System Architecture

MDDB uses off-the-shelf software and hardware components to minimize prototyping times through extensive software reuse, and to enable other bio-physics groups to easily reproduce our setup. MDDB can run on the PostgreSQL and Greenplum DBMS, with basic processing on protein systems provided by the MDAnalysis package [11] developed by the Woolf Lab, and data analysis from the MADLib library. MDAnalysis crucially provides data import and export functionality from the majority of the popular MD codes, and we use six major MD simulators in-house in our own trajectory production (CHARMM, Amber, NAMD, Gromacs, LAMMPS, Desmond). MDDB integrates with job scheduling software, specically Gearman, to manage trajectory production and to support highly dynamic degrees of parallelism in our iterative algorithms.

Figure 2 illustrates the setup of our MDDB instance, and its deployment on our 10 node commodity cluster with approximately 150 cores, 700 GB RAM and 100TB storage capacity. MDDB couples its DBMS with HPC resources for large-scale trajectory production, and a web application software stack to expose a biophysics web service and visualization capabilities. MDDB uses institution-level and national leadership-class computing resources for trajectory production, including JHUs DataScope system, as well as XSEDE and PSCs Anton resources. Our DataScope system provides a substantial oating point capability, with 90 nodes providing Tesla-class GPUs and SSD storage for pipelined simulation and ingestion. Currently, we transfer datasets from supercomputer resources synchronously after simulations, and we are exploring bulk network transfer techniques to maximize data ingestion efficiency for MDDB from wide-area sources.

2.4 Adaptive Control Framework

MDDB uses reinforcement learning (RL) techniques to control and implement a continuous, adaptive sampling algorithm. Figure 3 illustrates our adaptive controller as a learning agent which: (i) observes the state of the environment(e.g., the data collected so far) in order to come up with an appropriate simulation setup; (ii) interacts with the environment by

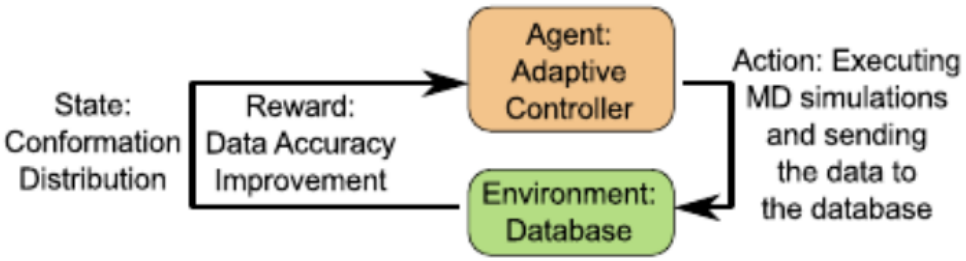

Figure 2.3: Adaptive simulation control as a reinforcement learning problem.

executing different simulations; (iii) observes the reward associated with performing this action at the current state and the new state; (iv) starts over. As learning progresses, the agent tries to find an action policy which maps each state of the environment to the best action available through statistical prediction and inference on the rewards obtained by performing different actions at different states. The goal of the adaptive controller is to maximize the total reward obtained from a sequence of actions over a long period of time.

2.4.1 Database Support for Reinforcement Learning

RL problems face several scalability challenges that can naturally be encoded as and overlap with DBMS challenges. RL scenarios typically operate in environments consisting of an extremely large number of states,

with unknown stochastic reward and state transition distributions. Thus RL algorithms often have to memoize and keep statistics over the combinations of states, actions, and rewards encountered in order to support policy evaluation, improvement and both value function and action-value function approximation.

DBMS encodings of RL algorithms can benet in the following ways. First, the RL environment and its states are often dened in terms of transformations of observations, and are thus derived data. These transformations can often easily and naturally be expressed through SQL queries. Next, the intermediate data and state in RL algorithms can be sizeable, and can benet from the scalable query processing and indexing provided by DBMS. We believe this is especially the case for a wide range of policy evaluation and improvement, many of which can be represented as iterative aggregation queries. Finally, we can bring to bear a wide range of incremental processing techniques, including advanced view maintenance methods [12] alongside query processing, to support the online execution of RL algorithms. This can be particularly fruitful for MCMC-based policy rollout [13] as well as gradient-based methods [14].

In the remainder of this section we will describe how the MD sampling and exploration (MDex) problem is dened as an RL problem with an environment, actions, and rewards.

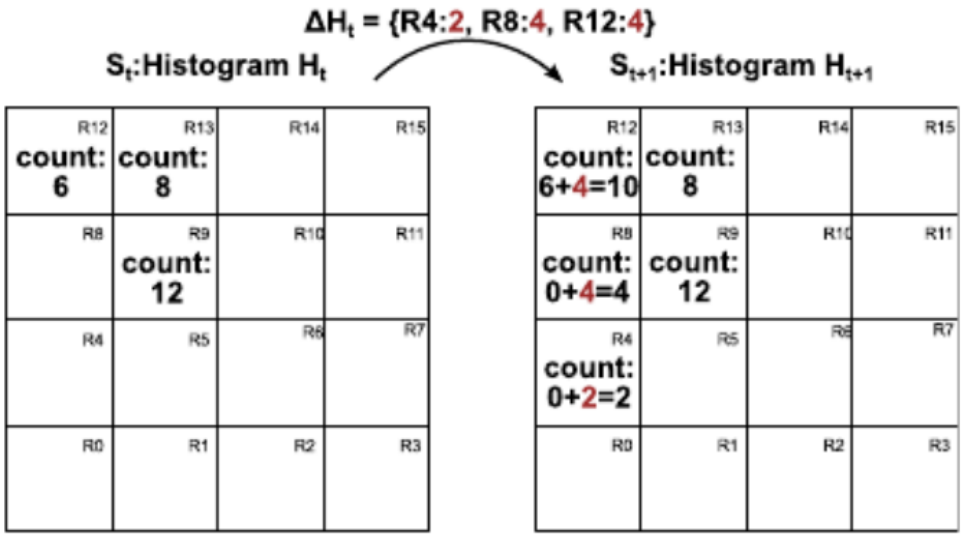

Figure 2.4: Transition between two environment states S_t and $S_t + 1$.

2.4.2 MDex Environment: Conformation Distribution

We dene the MDex environment as a set of states where each state represents the data distribution of conformations in phi-psi space. We use summarization of the conformation distribution to manage and to reduce the complexity of the state space. Specically, we use a histogram to maintain the visit counts of different regions in the phi-psi space. The current MDex environment state is simply represented using the histogram as shown in Figure 4. Our summary data structure is only used internally within the adaptive sampling algorithm. Up on convergence, we construct a Markov chain via post-processing of the complete trajectory dataset.

2.4.3 MDex Actions: Simulation Restart Decisions

For our set of actions, we consider two alternatives that influence the choice of our learning algorithm and the complexity of the learning process. Actions span the methods we can apply to select the starting points of MD simulations in terms of phi-psi values amongst other domain-specific simulation parameters and inputs (e.g., boundary conditions). For the former, examples include starting from an energy well, a sparsely populated region, or a randomly chosen point in the phi-psi space. Our two alternatives present the exibility of investigating reinforcement learning for MD under a discrete set of actions, and also a continuous-valued set of actions. Our actions are implemented as database queries that submit new jobs to our scheduling software.

2.4.4 MDex Rewards: Quality Improvement of the Data Obtained After Performing an Action

The reward after performing an action indicates how much the new trajectory data obtained from the action improves the accuracy of the Markov

chain in capturing the probability distribution represented by the energy landscape, and correspondingly the dynamics of the protein. This is done by measuring how close an MD output trajectory is to a random walk on a flat energy surface when using an MD simulator with an inverted energy landscape constructed using obtained trajectory data. The reward of the action performed at time t is the accuracy improvement in the Markov chain between the histogram summary at step $t + 1$ and step t.

2.4.5 MDex Policy: Solving for the Optimal Action Policy

We consider two families of RL algorithms, namely temporal difference (TD) learning and policy gradient methods. These methods reflect contrasting efficiency and flexibility in their usage, to allow us to investigate the suitability of RL in controlling sampling. TD learning concentrates on efficient, low overhead incremental learning and can readily be combined with prediction function approximation, and we view it as the low-cost method. Policy gradients present a generalized solution that can apply in both discrete and continuous scenarios, but involve more heavyweight gradient computations and ensure that actions are chosen within a local neighborhood of prior action. We implement these RL algorithms as user-dened

functions, with the policy itself implemented as a parameterized query.

2.5 Demonstration Scenario

2.5.1 Demo Setup

Figure 2 illustrates an example setup for our system which handles simulation jobs, data analysis, control and a web frontend. We plan to show that our system is capable of delegating computationally intensive tasks in the pipeline, such as MD simulations and analyses, to other machines. Specifically, we plan to use JHUs DataScope [15] for simulations and data analysis tasks. In addition, we will show the use of MDDB with an Anton machine [16] at the Pittsburgh Supercomputer Center. This is a massively parallel supercomputer for MD simulations of macro-molecules.

2.5.2 Web Frontend

We will also provide a web frontend interface (similar to the screenshot in Figure 5) to allow a user to interact with MDDB and monitor the progress of the adaptive control loop. The user will be able to interact with MDDB by choosing a protein system from a library of drug design targets, specifying the number of parallel simulation jobs, and then starting MD simulations.

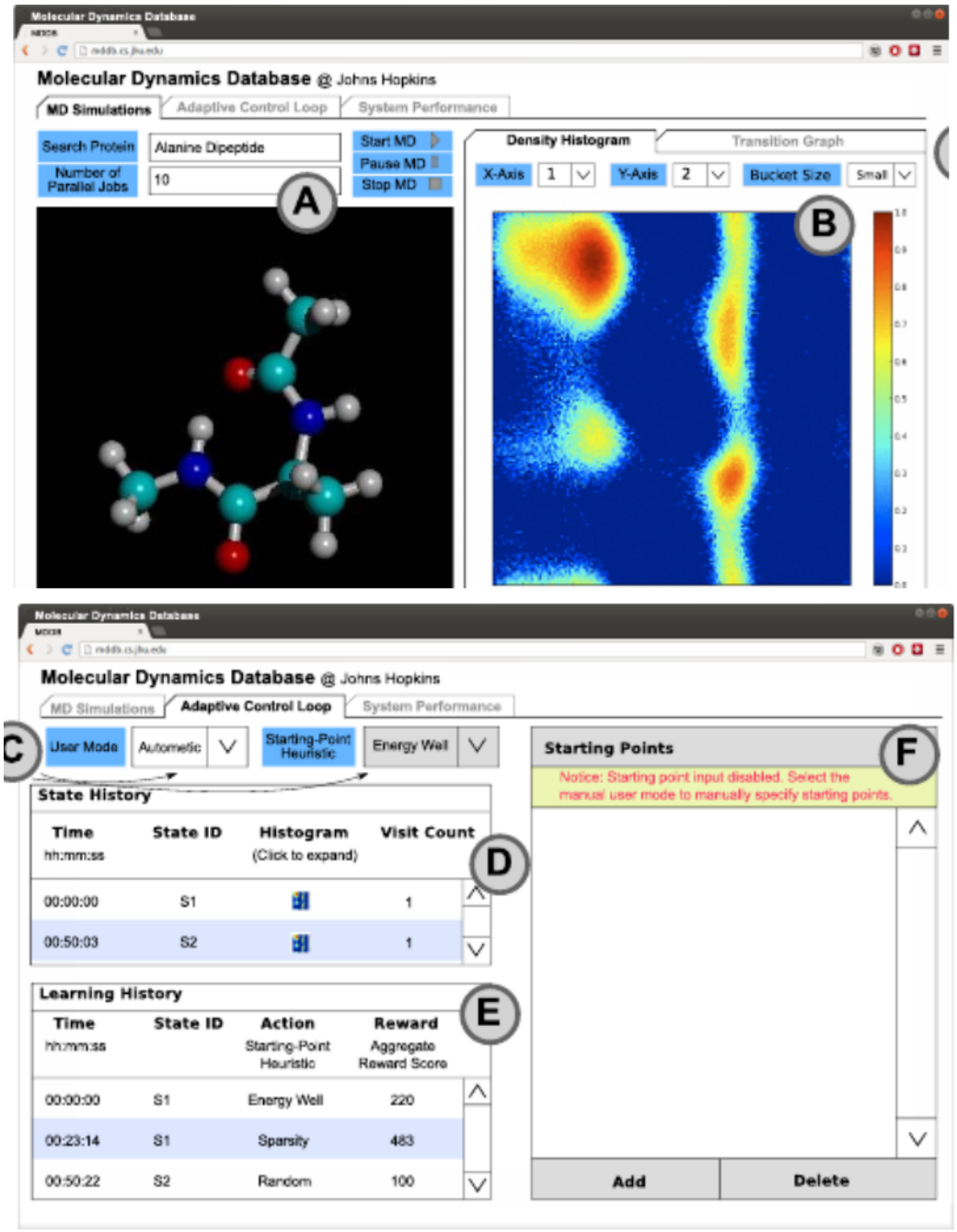

Figure 2.5: Web frontend. (A) Protein selection and MD worker allocation. (B) Progress visualization (density histogram and transition graph). (C) User mode and starting-point heuristic selection. (D) Environment state history. (E) Learning history in terms of states, actions, and rewards. (F) Starting point input.

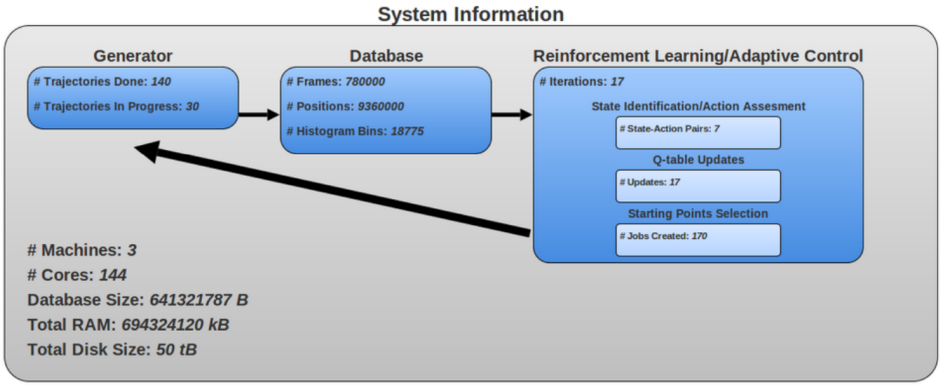

Figure 2.6: Generator runs protein simulation programs like Charmm or Amber. The database stores summarized data-stream generated by MD simulations. The Reinforcement Learning module examines current simulation results and redirects simulators toward interesting unencountered protein states

The progress of the adaptive control loop is shown as a density histogram and a transition graph. The density histogram discretizes the phi-psi space into histogram regions and maintains the visit counts for each region across all MD trajectories. When the dimensionality (the number of phi-psi angles) is greater than 2, the interface will provide options for the user to choose a combination of two phi-psi angles from which to plot a density histogram. The transition graph shows the transition probability distribution for each energy well discovered so far and the stability of each transition probability (i.e.,the percentage change in transition probabilities).

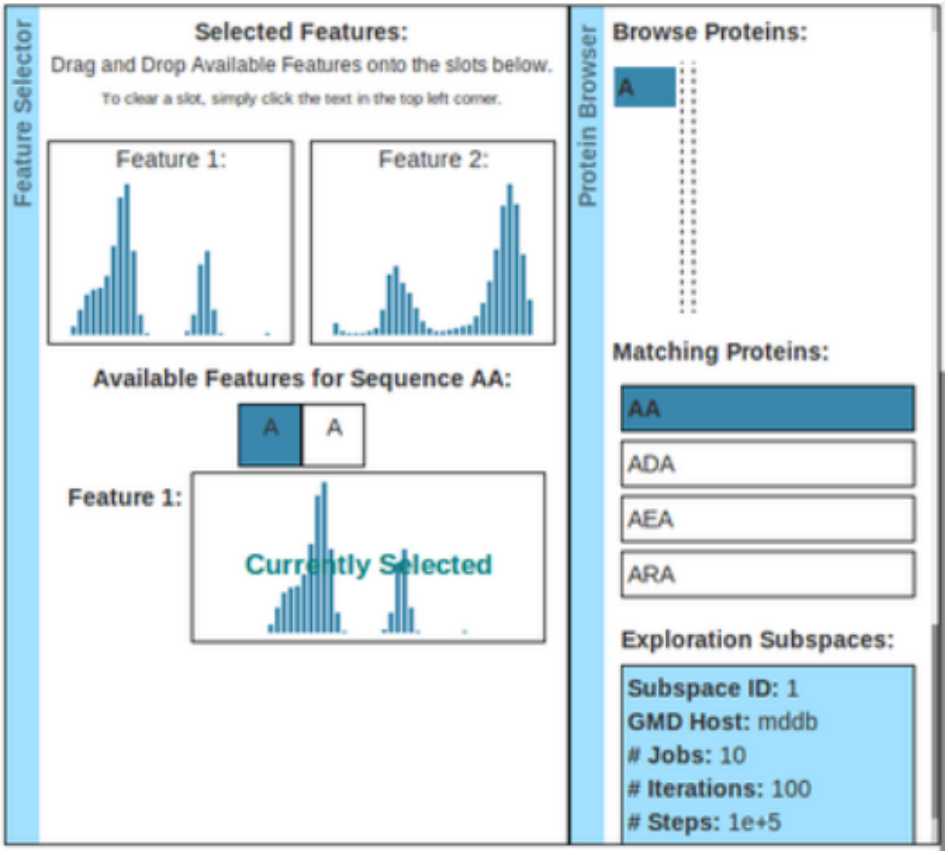

Figure 2.7: Feature Selection User Interface

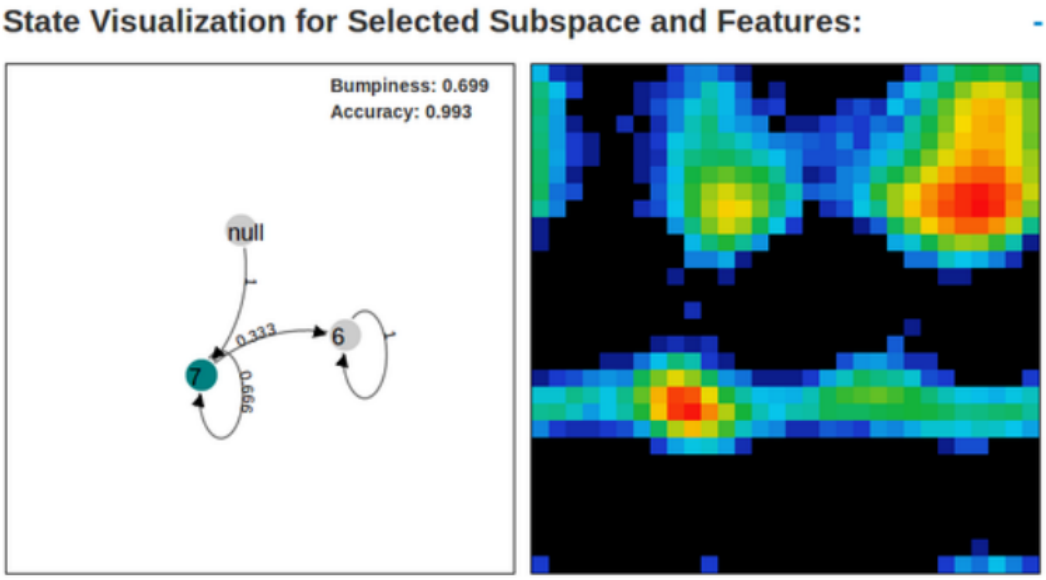

Figure 2.8: Goal of MD job is to find stable Markov chain representation of all protein states and transitions. This is an effective summary of protein behavior. Also shown is a heatmap/histogram representation of protein state energy landscape where hotspots are energy wells (low energy states) that proteins are most likely to reside in.

2.5.3 Audience Interactions

As a demonstration, our system will provide a rich interactive experience and allows for different levels of interactions which exposes the internal logic of the system at multiple levels. The user will be able to interact with the system in three different user modes.

- Automatic: This mode requires limited involvement from the user side. In the automatic mode, the adaptive controller learns the best starting-points heuristic in different situations. Users can monitor the progress through the density histogram and transition graph.

- Semi-automatic: This mode requires the user to specify the heuristic for choosing starting points to change it as they see fit by observing the distribution of the data collected so far using the density histogram and transition graph. The user will be able to compare their action to the one that would have been chosen by the adaptive controller in the automatic mode.

- Manual: This mode requires the user to act as a starting point heuristic by specifying a set of starting points. The system will keep sampling around these starting points until they are replaced by other points.

To showcase the effectiveness of adaptive control, we will compare the

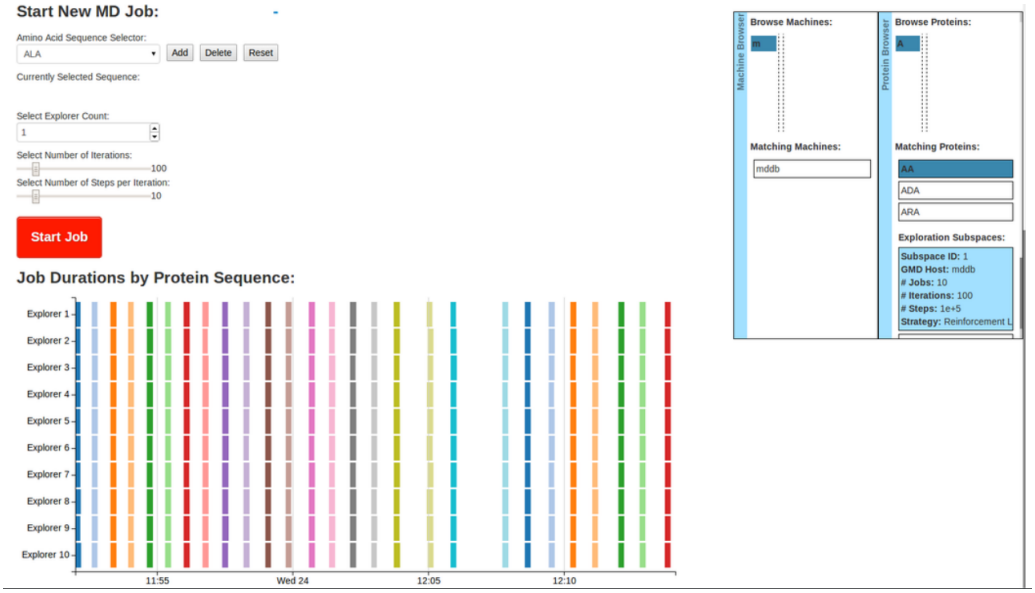

Figure 2.9: Web interface allows users to easily start new protein simulations. Ongoing and past simulations' progress and hardware impact monitored in real-time. Colored bars represent protein simulation execution. Between each iteration, adaptive control or reinforcement learning module assesses best way to guide simulation

cumulative reward obtained across user modes.

2.5.4 Demo Takeaways

Our demo will introduce MDDB to the database community, and in addition to the adaptive control mechanism shown here, will raise awareness of a large open science dataset (currently at 25TB and growing) that will be made available for DBMS research at the time of the demo.

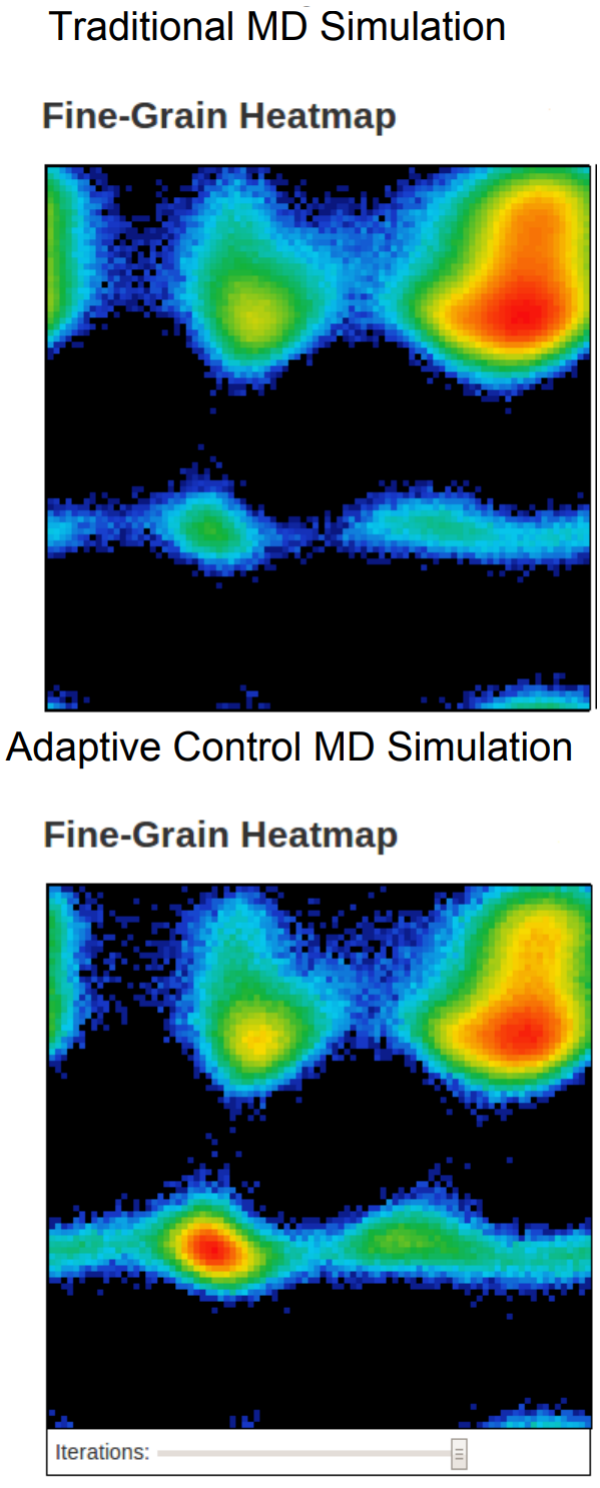

Figure 2.10: MD simulation with reinforcement learning more fully explores protein state space.

Chapter 3

Extreme Value Summary Statistics in a Distributed Top-K Algorithm

Extremely large (peta-scale) data collections are generally partitioned into millions of containers (disks/volumes/files) which are essentially unmovable due to their aggregate size. They are stored over a large distributed cloud of machines, with computing co-located with the data. Given this data layout, even simple tasks are difficult to perform and naive algorithms can easily become quite expensive. We present a one pass, communications-efficient technique useful for both estimating upper order quantiles and selecting the largest k elements across a highly distributed dataset or stream.

Our novel approach draws its foundations from Extreme Value Statistics (EVS) to reason about the statistical relationships between the tail distributions of dataset partitions. The tail of each partition is fitted by the Generalized Pareto Distribution, which captures threshold exceedances. The obtained parameters are communicated to a central coordinator and used to estimate quantiles, or solve for a threshold above which there are approximately k elements. We discuss the computational and bandwidth costs of the algorithm, and demonstrate the accuracy of the method on both a variety of synthetic datasets and a PageRank dataset.

3.1 Introduction

We present DOT-K, a new communication-efficient algorithm for solving a two common tasks in distributed data management: (a) estimating the kth largest element of a data set split over many partitions and (b) subsequently retrieving the largest k elements. More formally, given any real-valued $x_r = f(A_r)$ function defined on (one or more) attributes A_r of row r in the dataset D_i of partition i, our objective is to accurately estimate the kth largest value $f_{(k)}$ over all $r \in D$ the entire collection of partitions such that the size of the

result set is equal to a given k,

$$\left| f(A_r) : f(A_r) > f_{(k)} \right| = k. \tag{3.1}$$

Top-k and quantile queries are relevant to a variety of applications, including outlier detection and general data exploration. Top-k queries are commonly executed in all sorts of database environments, many of which are sensitive to different bottlenecks. The scope of top-k query processing is reflected in the amount of solutions that cater to specific problem environment requirements [17]. The Threshold Algorithm (TA), developed by [18] [19] [20], is a well understood and efficient method of finding the top-k values of a monotonic aggregation function over row attributes; in the TA setting, the top-k query result will be the largest valued outputs from a scoring function that takes several row object attributes as input. The TA-style top-k query has received much attention and has successfully been adapted to a distributed environment by algorithms such as KLEE and TPUT [21] [22]. Distributed TA-style algorithms are appropriate for a column-partitioned database where a single object's attributes may be spread across multiple partitions, such as a heterogeneous multimedia database where a row object's video data is stored at a different node than its associated image or text data. In contrast, we examine the top-k query in a highly distributed,

row-partitioned database system; DOT-K assumes that the data objects over which a top-k query is made are represented in full at their respective partitions. Applications where a single, homogeneous dataset must be split between many parallel nodes are an appropriate use case for DOT-K. An enterprise network tap, in which logs of company net activity are collected and stored at many distributed locations, would be an ideal setting. Another example is a top-k query over a scientific dataset row-partitioned across a large cluster of machines, at a scale where minimizing communications costs between machines becomes a priority.

We focus on consuming minimal network resources when solving top-k queries in a highly distributed environment. Let P be the amount of a dataset containers, or partitions. A naive distributed top-k query would ask each machine to forward their local partition's largest k values and subsequently sort the P lists to find the global top-k result. The naive solution is completely acceptable when P or k is small; however, at a large, datacenter scale the network cost of communicating the P local top-k lists and the computation cost of sorting those lists becomes prohibitive. In DOT-K, we address these costs by summarizing the tail of each dataset partition using an Extreme Value Statistics distribution. Instead of communicating P sets of local top-k lists, each partition forwards the EVS distribution parameters that describe the local top-k values. The central query coordinator

then relates the tail distributions of each partition and calculates an estimate for the global dataset's kth order statistic. Finally, the estimate for the kth largest value is communicated to all dataset partitions and all values greater than the kth order statistic estimate are sent back to the query coordinator.

DOT-K is also communications-efficient at estimating many high-order quantiles in one query. While not applicable to answering medians or any quantile query across the board as in Greenwald-style quantile estimation algorithms, such as those given by Zhang and Wang [23] [24], depending on the DOT-K implementation choice of local EVS distribution parameter estimators, the DOT-K method can achieve similar memory and compute costs as sketch-based quantile estimation algorithms as detailed in Section 3.2.

In Section 2 we examine the necessary Extreme Value Statistics pertinent to DOT-K. In Section 3, we present the DOT-K algorithm and our Threshold Equation. Section 4 outlines our experimental communications and accuracy evaluations of DOT-K, and we conclude with possible improvements in Section 5.

3.2 Extreme Values

Extreme Value Statistics (EVS) is concerned with characterizing the tail distributions, or extreme values, of random variables. EVS has traditionally been used to model extreme environmental phenomena, such as sea levels or wind speeds, as well as weakest-links in reliability modeling [25]. An area of interest in EVS is modeling a distribution's random variables which exceed a threshold. As a top-k query is concerned with the k data elements which exceed the kth order statistic threshold, points-over-threshold EVS has a natural application to top-k query processing. An EVS theroem developed by Pickands, Balkema and de Haan states that the distribution of threshold exceedences of a sequence of independent and identically-distributed random variables with a common continuous underlying distribution function is approximated by the Generalized Pareto Distribution, and that the approximation converges as the tail threshold rises [26] [27] [25]. Therefore, for a large class of common data distributions, the k largest values may be well approximated by a Generalized Pareto Distribution if the kth order statistic is an appropriately high enough threshold. Picking a threshold that defines the tail can be a delicate choice; due to a bias-variance trade-off, a lower threshold results in a worse GPD approximation while a higher threshold limits the amount of available threshold excee-

dences leading to greater parameter estimation uncertainty [28]. However this property bodes well for larger datasets, and benefits DOT-K at extreme scales. The true dataset tail grows with the dataset size, which affords a more accurate tail summarization.

In a distributed query setting, we summarize the extreme values, or the tail, of each partition in order to attain minimal communication costs while still relate data across distributed nodes. Rather than sending samples of tail data, the partitions describe their local largest k values by fitting a GPD and communicating the GPD parameters to the central query coordinator.

The probability distribution function of the Generalized Pareto Distribution

$$p(x|\xi,\sigma,\mu) = \frac{1}{\sigma}\left[1+\xi\cdot\left(\frac{x-\mu}{\sigma}\right)\right]^{-\frac{1}{\xi}} \tag{3.2}$$

is defined by three parameters: the threshold μ, the shape ξ, and the scale σ. There are several methods for estimating the GPD parameters that best model a given set of threshold exceedences. A survey of these methods along with their respective strengths and weaknesses is not within the scope of this paper. Many authors have approached the subject of GPD parameter estimation including Pickands [26], Hosking and Wallis [29], Castillo and Hadi [30], Zhang [31], and Husler [32]. In our experimental implementation of DOT-K, we use a Maximum Likelihood Estimator based approach to fit a GPD to each dataset partition's extreme values. The accuracy of a

DOT-K query result is entirely based upon the accuracy of the GPD summarizations of each dataset partition tail. We assume an extremely large distributed dataset with at least thousands of extreme values within each dataset partition. When fitting 500 or more data elements an MLE-based method is a proven GPD parameter estimation technique [29].

For a given GPD, one can calculate the threshold level x_m that is exceeded on average once every m observations in the underlying dataset [25]. The threshold level is useful when estimating dataset quantiles. By relating m to the dataset size, we can solve for order statistics. Coles [25] gives the formula to solve for the m-observation return level

$$\zeta_\mu \left[1 + \xi \cdot \left(\frac{x_m - \mu}{\sigma}\right)\right]^{-\frac{1}{\xi}} = \frac{1}{m} \tag{3.3}$$

where ζ_μ is the probability of an observation exceeding the GPD threshold parameter μ, which is estimated by

$$\hat{\zeta_\mu} = \frac{N_\mu}{N} \tag{3.4}$$

where N is the dataset size and N_μ is the number of elements greater than the GPD threshold μ. Since DOT-K fits each partition's local largest k values to a GPD, then the value of ζ_μ is calculated with $N_u = k$ and $N = n_i$ where n_i is the local partition set size. We expand on these ideas and

generalize them to partitioned datasets.

3.3 DOT-K Algorithm

Our goal is to first estimate the kth largest element of the entire data collection D and subsequently retrieve all elements greater than the estimate. This is achieved in the following steps:

1. Model the tail distribution of the quantity of interest in each partition and communicate the results to a central coordinator
2. Using the GPD fits from all partitions, solve for a global threshold that is expected to yield the largest k elements
3. Query the partitions with a global threshold and return all elements that exceed the limit

The rest of this section is concerned with obtaining the global threshold, the local strategies and analysing the communication requirements of the algorithm.

3.3.1 Global Threshold

For each partition i, the coordinator knows not only the GPD parameters but also the number of elements $\{n_i\}$ with which in hand, it can estimate

the tail distribution of the entire collection, $D = \cup D_i$. For that we obtain the m-observation exceedance equation

$$\sum_{i=1}^{p} n_i \, \zeta_{\mu_i} \left[1 + \xi_i \cdot \left(\frac{x_m - \mu_i}{\sigma_i} \right) \right]^{-\frac{1}{\xi_i}} = k \tag{3.5}$$

which simply states that the total estimated number of exceedances above the global threshold x_m is equal to the requested number of elements, k, cf. Eq. (3.3). This one-dimensional equation can be solved numerically using standard off-the-shelf procedures. We note that the interval on which x_m is defined is determined from individual fits that each limit the possible values. Depending on the shape parameters the contraints can bound from both sides. In practice, this is not a limitation but a way to better initialize the numerical solvers.

It is worth pointing out that if any one of the nodes fail to deliver their estimates due to a temporary outage, the coordinator can obtain a global threshold by simply ignoring the missing partitions. This estimate will be more generous than the true solution would have been in the sense that using the obtained threshold on the entire collection would simply return more than the expected number of elements. If the failed partition comes back online for the second pass when the actual selection is performed, the result of the query is still going to be correct at the expense of a few more

communicated data elements.

3.3.2 Quantile Estimation

DOT-K is easily adapted to high order quantile estimation. Once the local GPD parameters are collected at the central coordinator, Eq.(3.5) may be used to estimate order statistics greater than the original kth order statistic by varying k in the equation, at no additional communications cost for queries concerned with the state of the dataset at the time of local GPD parameter collection. Further queries on a stream with an updated underlying dataset require a refresh of the local GPD parameters and a new round of communications.

Quantile DOT-K, if implemented with the more inaccurate but computationally light Method of Moments [33] GPD parameter estimation algorithm, need only track local dataset mean, variance, and size at each distributed partition or stream. While local partition compute and memory cost associated with maintaining these basic statistics would be significantly less than a summary data structure based quantile algorithm, DOT-K query error would undoubtedly increase from the worse Method of Moments technique [33]. Future work includes further experiments with different GPD parameter estimation techniques in order to optimize DOT-K's local compute and memory costs while maintaining query accuracy.

3.3.3 Local Strategies

Like several other distributed top-k algorithms, at each node DOT-K assumes either sorted access to the local partition data [17]. If sorted access is unavailable, a linear scan is performed to find each partition's local top-k values.

In an online distributed stream setting, DOT-K requires summary statistics maintained over the local streams in order to keep up-to-date GPD parameters describing the local tail. For the more accurate but computationally expensive parameter estimators, such as Maximum Likelihood, a length k priority queue data structure maintaining each local stream's top data elements is needed in order to obtain accurate GPD parameters at the time of a top-k query.

However, there may be ways to estimate the GPD fit in each partition using approximate sub-linear algorithms. Wu and Jermaine have discussed an approximate Bayesian method to predict the extreme values [34]. The method is based on the observation that the expected value of the k-th ranked element out of N samples is the (k)-th value out of n sub-samples, where $k/N = k/n$. One can generalize this technique to compute the GPD fit from a small subset of the approximate top ranked (e.g. k = 10, 50, 100) elements. One can also include a few extra elements for redundancy and validation of the fit. This can provide an additional speedup of the other-

wise liner scan of the partition data. We need to explore further that given a sub-sampling rate what subset of the ranks is optimal for the estimating the GPD parameters.

3.4 Experiments and Discussion

We run and discuss two separate sets of experiments in order to evaluate the performance of DOT-K. Our first experiment investigates the communication costs and overhead incurred by DOT-K in a distributed computing environment; the second experiment gauges the quality and accuracy of DOT-K queries at scale over a variety of datasets in a pseudo-distributed environment.

3.4.1 Communication Overhead

DOT-K exhibits minimal communication costs. There are four series of messages between the query coordinator and the dataset partitions. DOT-K starts with a message from the coordinator to the partitions containing the query details, namely, the value for k. After the P partitions calculate the local top-k values and the GPD fits, each partition communicates the three GPD parameters and local partition size to the coordinator. Next, the coordinator forwards the kth order statistic estimate to the partitions,

and finally the partitions send the estimate exceedances to the coordinator. Therefore, there are a total of $4P$ messages with a total of $6P + \hat{k}$ real values transmitted, where $\hat{k}$ is the count of estimated kth order statistic exceedances.

When used to estimate high-order quantiles, DOT-K requires even less communications between nodes. The query starts with a request from the coordinator to the distributed partitions, and communications are finished when the partitions forward the locally fitted GPD parameters back to the coordinator. With a total of $2P$ messages containing a few real values each, the coordinator may estimate any order statistic greater than the kth order statistic.

3.4.1.1 Message Count Experiment Setup

The goal of this experiment is to empirically evaluate our predicted communication costs of DOT-K, for both computing a quantile and a top-k query over a partitioned dataset. We implement DOT-K on Apache Storm [35], an open source distributed stream processing engine currently in use by many industry members including Twitter [36]. Storm treats a distributed program as a directed graph, with computation happening at graph nodes and data transfer along graph edges. DOT-K is ideally mapped to the Storm distributed computing model: there is a node for each distributed dataset

partition, or incoming stream, and there is a central coordinator node that receives the partition summaries, then estimates the kth order statistic and broadcasts that threshold back to the partition nodes. Each Storm node logs whenever it emits or receives messages to and from other nodes; from these logs we recreate all node-to-node messages, and quantify the communications costs incurred by DOT-K.

We deploy eighty Storm workers, or computation nodes, spread over twenty Amazon AWS EC2 'm1.medium' instances. We simulate a streaming environment by randomly scattering the dataset among the nodes and constructing a Storm 'spout' to emit the data values one by one. Each node runs a Storm computation 'bolt' that maintains a priority queue of the largest-valued data elements emitted from the stream and estimates local GPD parameters with data held in the priority queue. Note that even if two nodes reside on the same instance, they still communicate using the Storm message service and the communication will be logged and counted in our evaluation. The experiment records the total number of individual messages sent between computation nodes during the course of both DOT-K quantile and top-k queries executed at scale varying between ten and eighty nodes. Figure 3.1 shows the global message count associated with executing DOT-K at a given parallel scale.

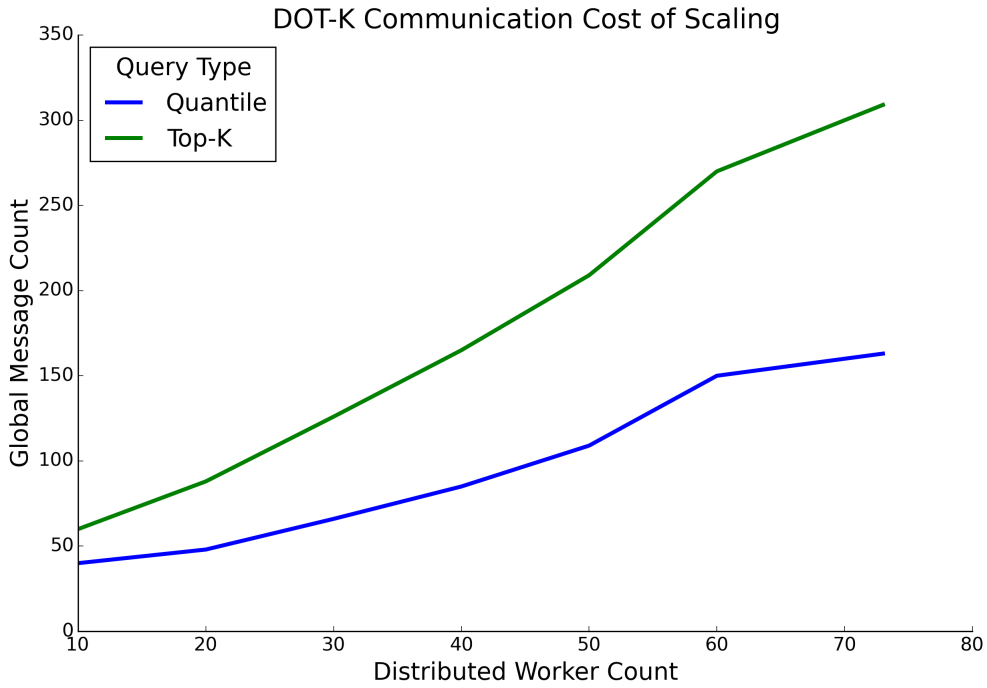

Figure 3.1: Machine-to-machine communications sent during DOT-K query executions

3.4.1.2 Message Count Discussion

Shown in Figure 3.1, it is clear that total message count per query grows linearly with parallelization, or partition count. In the previous section, we calculated that there would be one round of messages, or $2P$ necessary messages, in order to estimate high order quantiles and two rounds of communication, or $4P$ necessary messages, in order to obtain a top-k query result. However, the experiment shows a slightly higher message count than what was estimated; upon inspection of message contents, we found the extra few communications to be overhead incurred by Storm's fault tolerance and reliability features. Had we disabled the guaranteed message passing features

or not counted them in the total message count, the results would match our predictions more exactly. Regardless, we show DOT-K communication cost trends to be minimal for distributed quantile and top-k queries.

Note that the size of the contents of each message is less than a dozen real-valued numbers, with the exception of the final batch of messages during a top-k query in which each dataset partition sends the data elements greater than the kth order statistic back to the coordinator to form the top-k result; these messages contain, on average, k/P data elements.

3.4.2 Query Accuracy Evaluation

Our experimental goal is to discover any error trends and determine the practicality of scaling DOT-K on a variety of datasets. While the Pickands-Balkema-de Haan theorem states that the GPD is a good approximation of threshold exceedances, it is important to show the accuracy of this summarization method in practice. We implemented a pseudo-distributed version of DOT-K to evaluate the quality of top-k query results.

Our experiments show the relative error of DOT-K as we increase the number of dataset partitions. The number of elements of each partition is constant; as we increase partition count, the global dataset size grows as well. If we had kept global dataset size constant while increasing partition count, GPD approximation of partition tails would become worse as parti-

tion size decreased and we would be unable to show how partition count alone affects DOT-K accuracy.

We demonstrate DOT-K on datasets with several different partitioning strategies. While some databases are partitioned randomly, many applications partition datasets on some criterion and it is restricting to assume every dataset will have identically and independently distributed data across partitions.

The relative error metric we use is described in equation 3.6. $\hat{k}$ is the top-k query result from DOT-K, and is the set of dataset values that exceed the estimated kth order statistic.

$$\delta k = \left| \frac{k - \hat{k}}{k} \right| \tag{3.6}$$

3.4.2.1 Synthetic Datasets

We evaluate DOT-K on synthetic datasets consisting of real-valued random variables drawn from a randomly partitioned exponentially distributed synthetic dataset. DOT-K relative error results were obtained from the average of twenty executions on datasets generated with the same distribution parameters but different random seeds. For these experiments, k is 1,000, partition size is fixed to 300,000 data elements, and the partition count ranges from 1,000 to 10,000.

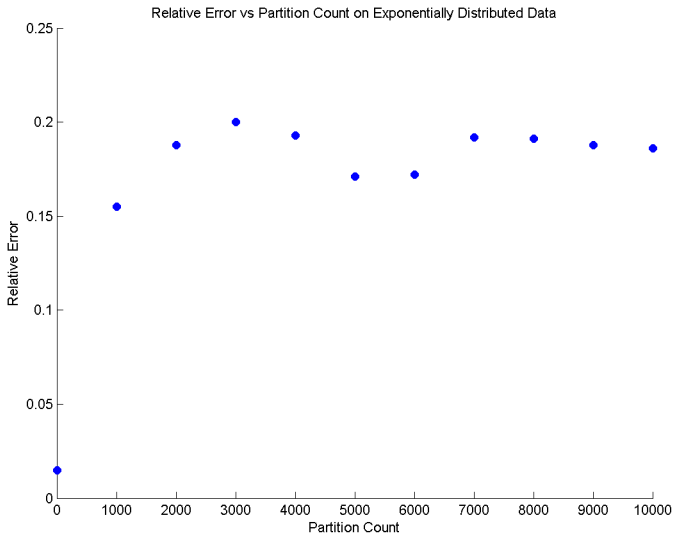

Figure 3.2: Relative error at large scale on a randomly partitioned exponentially distributed synthetic dataset.

Figure 3.2 shows DOT-K relative accuracy at high scale with partition counts from 1,000 to 10,000.

3.4.2.2 Berkeley PageRank Dataset

We evaluated DOT-K on the Berkeley Amplab Big Data Benchmark PageRank dataset [37] in order to test our method on a somewhat common application: finding the k highest ranked websites. The Berkeley PageRank dataset is based on the distributed system analysis benchmark work done by Pavlo et al [38] and consists of approximately 90 million URLs associated with their respective rank. For these experiments, k is set to 1,000, the amount of partitions ranges from 100 to 1,000, and the partition size is set to 90,000 URL-PageRank pairs. We run accuracy experiments on two

versions of this dataset: randomly partitioned and biased partitioned. For the randomly partitioned PageRank dataset we shuffled the data elements randomly over each partition. The biased partitioning scheme consists of simply ordering the data elements as we obtained them from the Berkeley Benchmark download; upon inspection, it is clear that chunking the original dataset, in order, produces partitions with very different tails. This undefined, but clearly biased, partitioning scheme is useful in simulating a real world practical query, as data is not always independently and identically distributed.

We ran experiments on two partitioning schemes of the PageRank dataset. Figure 3.3 shows average relative error of running DOT-K over twenty different random partitionings of the PageRank dataset. In this experiment, the data across partitions is independent and identically distributed.

Figure 3.4 shows the relative error of DOT-K executed on the PageRank dataset in the same state in which we obtained it; that is, partition data is differently distributed between partitions and there is a clear bias of extreme values concentrated in a few partitions. This experiment shows the result of one execution of DOT-K over a single PageRank dataset with a biased partitioning scheme.

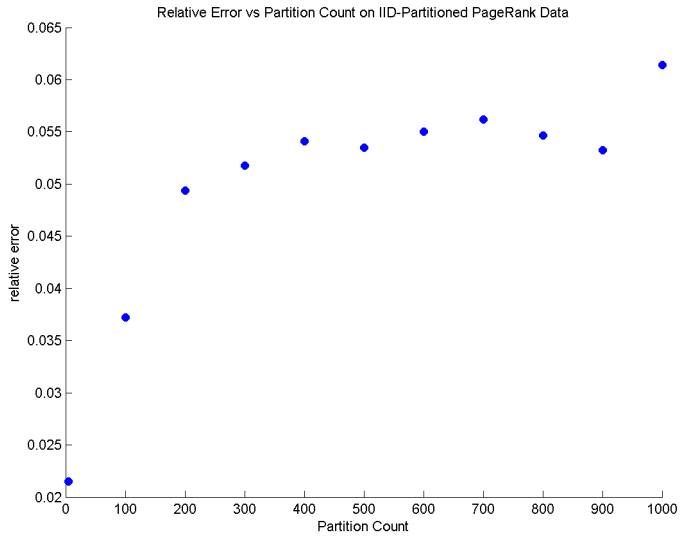

Figure 3.3: Relative DOT-K Error on a PageRank dataset with a random paritioning scheme.

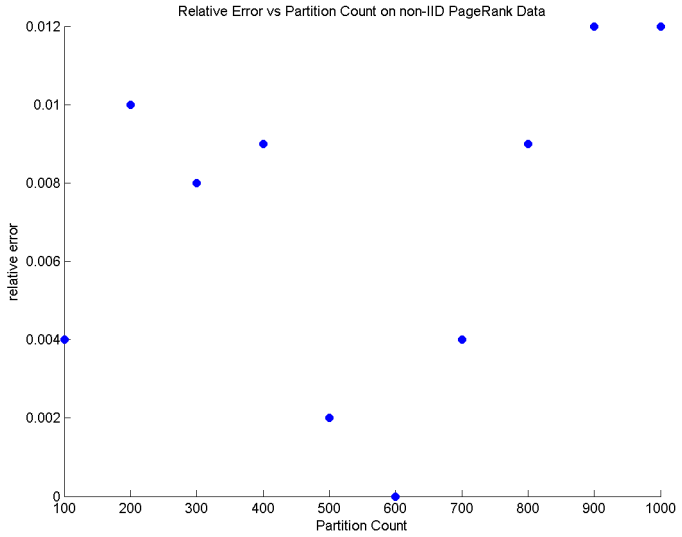

Figure 3.4: Relative DOT-K Error on a PageRank dataset in which extreme values are concentrated in few partitions.

3.4.2.3 Accuracy Discussion

DOT-K exhibits two trends between error and partition count. The error on the PageRank and exponentially distributed datasets either tails off to a limit or is too low to show any certain trends. However, error generally rises with partition count, likely due to the rising cumulative GPD fit uncertainty. Each GPD fit to local partition data has at least some small amount of error in its description of the real data; as partition count rises and more GPD parameter sets are involved in the DOT-K estimate of the global kth order statistic, there is both a greater cumulative amount of GPD fit error and a greater kth order statistic estimate error.

Surprisingly, DOT-K showed less overall error on the biased partitioned PageRank dataset than the randomly partitioned PageRank datasets. DOT-K executed on the biased partitioned PageRank data resulted in error too small to show any real trend. The nature of the biased partitioning may have aided DOT-K; in the unaltered PageRank dataset, URLs with outlier ranks were concentrated to few partitions. The DOT-K global threshold equation estimates that the many partitions with lesser tail distributions would not contribute any elements to the top-k query result. Partitions with maximum data values less than the largest threshold GPD parameter will not be represented in the global top-k query result, and may be pruned away. Therefore, DOT-K only needed to relate few partitions, meaning less

cumulative GPD fit error represented in the kth order statistic estimate. This property benefits applications in which the data is not randomly partitioned; DOT-K error scales with the amount of partitions that are estimated to contain elements belonging to the query result.

3.5 Conclusion

DOT-K shows promise as a communications efficient distributed top-k elements algorithm. By summarizing the tail distributions of dataset partitions, DOT-K pushes computation out to distributed nodes in order to conserve bandwidth usage; only the query, GPD parameters, kth order statistic estimate, and global top-k elements are communicated. Due to Pickands-Balkema-de Haan Theorem, the Generalized Pareto Extreme Value Distribution becomes more effective at describing dataset tails as the dataset size increases. DOT-K will become more useful as datasets grow larger and the desired k largest values are numerous enough to otherwise incur substantial bandwidth and sorting cost.

There are several improvements to be made to the DOT-K algorithm. Firstly, the accuracy of the entire method rests on the quality of the GPD parameter estimation at each of the dataset partitions. Numerous publications examine the quality of GPD parameter estimators, and several new

methods given by Zhang [31] and Husler, Li, and Raschke [32] combine the favorable qualities of different estimators. Performance would benefit if these methods provide a better or faster GPD fit. Secondly, we would like to experiment with different methods of reasoning about the relative tail distributions of the dataset partitions. Pruning away partitions estimated to not contribute to the top-k result is a start, but further inferences on the relationships between partition tail data may be possible. Finally, we would like to experiment with sublinear sampling techniques when fitting GPDs to partition tails. Currently DOT-K requires each partition to perform a linear scan to find the local top-k values; if a GPD could describe a sample of a partition tail without a significant loss of overall accuracy, the overall performance of DOT-K would benefit.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 4

Real-Time Floating-Zone Furnace User Guidance via Image Segmentation and Object Classification

The optical floating zone technique is widely used to prepare high purity materials in single-crystal form, helping enable fields ranging from quantum materials to modern electronics and optical devices. Here we demonstrate that machine learning methods, specifically deep neural networks, can be used to efficiently, and in real time, segment video frames during growth, identify the size and shape of the molten zone, and classify its

stability. Transfer learning enables rapid, effective model training to discriminate between three common zone stability modes with under 1000 manually labeled training artifacts. Sustained drops in model confidence effectively indicate transitions between modes and enable operator intervention to restabilize the growth. Further, we demonstrate that this model, trained on labeled images from one materials family, effectively identifies zone transitions during growths of a second, distinct class of materials, i.e. the ML/AI methods obviate the need for per-material training. Our results pave the way to effective acceleration and automation of the synthesis of new, functional materials by the widely used floating-zone method.

4.1 Introduction

Science and engineering research and applications depend on a wide range of data. Recent advances in machine learning (ML/AI) provide multiple methods to accelerate reduction and use of image information (e.g. [39] [40]). Fast ML/AI models hold particular promise for providing real-time feedback, optimization, and instrument control during complex experimental processes. Instance segmentation, the process of correctly detecting and precisely identifying different objects within an image, is a critical step in advanced analysis of image data, but is often complicated by the lack of

suitable quality and quantity of application-specific training data. In this report we provide both the trained ML model and open release of the associated dataset as components of materials data infrastructure required for data-driven materials science (*c.f.* [41] [42]).

The optical floating zone (FZ) technique is widely used in the preparation of functional materials. It is a specific instantiation of directional solidification, in which material crystallization occurs in one direction at a well-defined solid liquid interface. As a crucible-less technique, FZ methods are known for the ability to prepare materials of exceptional purity and uniformity that when combined with traveling-solvent methods are applicable to a wide range of inorganic solid state materials [43]. With this versatility, however, comes significant complexity; the attributes of the grown material depend not just on the input chemical stoichiometry and general thermodynamic conditions (e.g. temperature, pressure, gas fugacity), but on kinetic factors, including mass transport and fluid flows in the molten material, driven by a combination of instrument controls, pressure/temperature gradients, and physical properties (e.g. thermal conductivity) of the growing material itself. This complexity, combined with the opaque interdependence of control variables, has impeded both the broader use in growing new materials and automation of growth of known materials except in cases where the same material is being grown many times, at scale, when the benefit of

automation can finally outweigh the significant costs.

The FZ technique, therefore, provides an important application of recent advances in machine learning to accelerating scientific discovery. Here we report the development of a robust, instance-segmentation model for molten-zone geometry in a laser-diode, floating zone furnace. This model provides a route to real-time modeling of the synthesis of single crystals central to the development and production of novel electronic and optical materials. Modern instance segmentation, such as Faster R-CNN, relies on composite systems linking a region-position network with an object detection network [44]. In this study, we utilize the state-of-the-art Mask R-CNN system that adds predictive segmentation masking on each region-of-interest by implementing a fully convolutional network in parallel with Faster R-CNN's classification branch and bounding-box regression [45]. Automated and robust instance segmentation opens up not only the possibility of accelerated and partially automated FZ growth of new materials, but also provides a foundation on which to develop techniques to extract chemically meaningful information from optical imagery data.

4.2 Results

The top row of Figure 4.1 shows representative images of the three molten-zone classes identified for segmentation in this study. In each class, the molten-zone is the white and gray shaded region between the pink and orange. The molten zone is suspended between the varicolored feed rod above and the crystallized seed below. From left to right these molten zones are classed as a steady-state, stable growth (”stable melt”), a progressively unstable melt due to excess seeding rate from crystal rod extraction (”fast bottom”), and a progressively unstable melt due to excess feeding rate (”fast top”). The lower row of Figure 4.1 shows three similar molten zones with the region of interest outlined in a green dashed line and the mask in red as identified by our deployed ML model. We use the traditional model confidence ranking of He, et al. [45] which purports to be a measure of class match quality, but as such also incorporates aspects of mask fit quality. Note how closely masks fit the molten zone for these examples with model confidences ranging from 0.65 to 1, this despite the realization that in some applications the mask fit is poorly measured by these confidence rankings [46]. For comparison, Figure 4.2 shows an example of a labeled training image where the mask was created by manually marking coordinates along the outer edge of the molten zone. Masks, including those in Figure 4.1, are

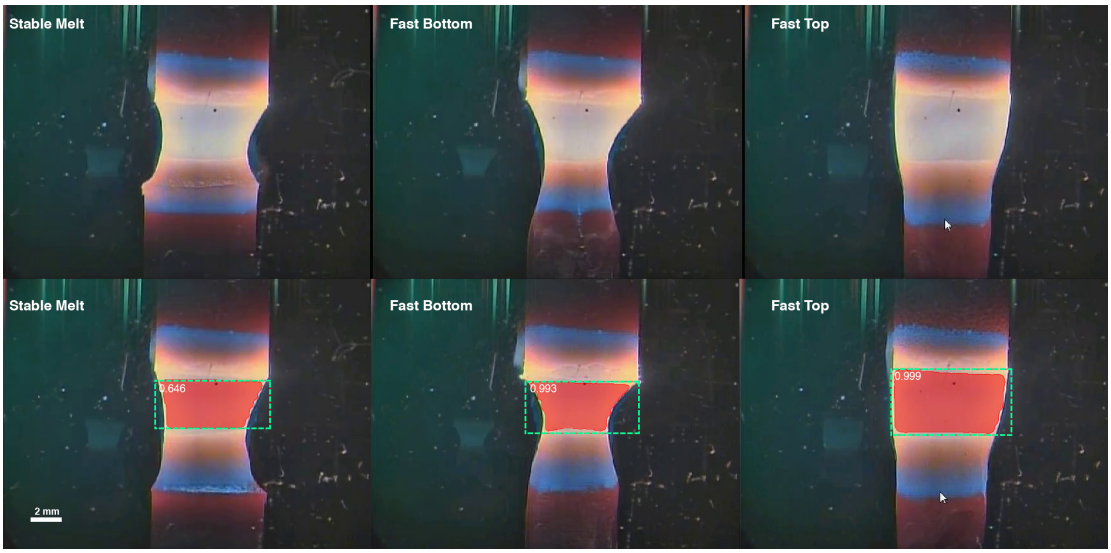

Figure 4.1: The top row are examples of the three boron carbide floating zone classes used to train the Mask R-CNN model. From left to right, ”stable melt,” ”fast bottom,” and ”fast top.” The bottom row shows regions of interest (dashed green rectangles) with model confidences noted and masks (red highlight) identified by the model for representatives of each class.

created by the trained model in under one second when deployed on CPU architecture and approximately 15 times faster with a single GPU.

4.2.1 Training Regimen Comparisons

Training configuration details have a direct effect on model performance. First-order hyper-parameter and model pre-training were tested to balance training efficiency with model accuracy and precision given the inherently small training set available.

In the Matterport Mask-RCNN implementation used in this study, training length comprises of the number of epochs multiplied by the number of

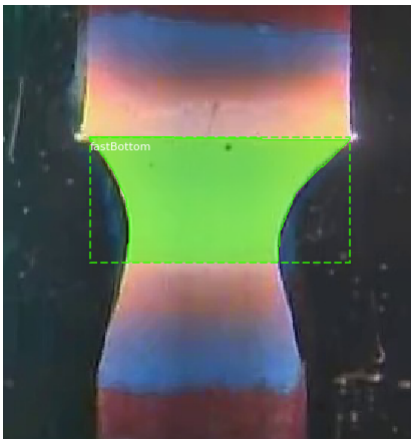

Figure 4.2: Typical training image with a manually outlined molten-zone mask and identified class for the ”fast bottom” class. 990 floating zone images were labeled for the training

steps per epoch [47]. Typically, one epoch is considered one full iteration over the entire training set, with steps per epoch being equal to the number of training set elements. However, to shorten training, one may set the steps per epoch to less than the size of the training set and randomly sample members of the set when training. Figure 4.3 shows the results of increasing training steps when using eight training epochs. Note that even for the short training regimens in this example we see the start of model convergence and diminishing returns on training beyond 200 steps. For this reason we limited further model development to 200 steps per epoch.

Figure 4.4 compares training of the entire network to only updating trained weights in the first layer (commonly referred to as training the network heads) all while training atop pre-trained networks. Figure 4.4 also shows the the sensitivity of observed model convergence to the number of

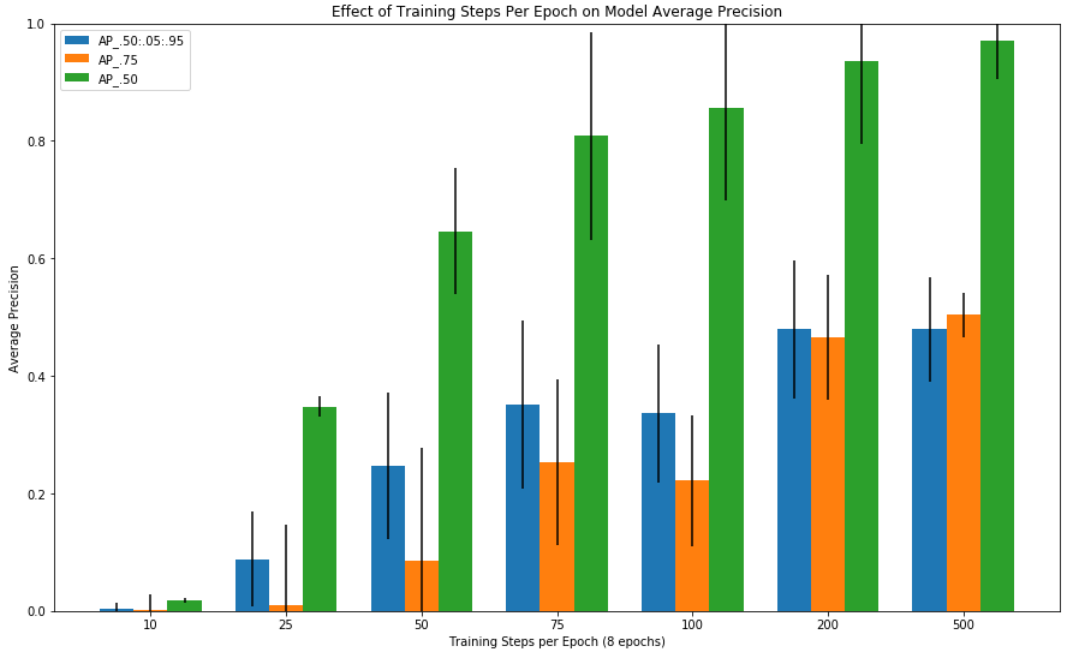

Figure 4.3: Training steps per epoch on model average precision. Each data point is the average performance of ten models, one model for each fold of the 10-fold cross validation split of the training data. The error bar represents one standard deviation in each direction of the ten models' Average Precision score when applied to the separate test set. AP_{50} is the average precision with a mask Intersection-over-Union threshold of .5, AP_{75} is the same with a IoU threshold of .75, and $AP_{50:.05:.95}$ is the average of the all AP scores taken with IoU thresholds from .5 to .95 in .05 increments. As we take more steps for training, our models not only score better but are more consistent between cross-validation folds. No evidence of over-fitting is shown yet at these training lengths, however AP performance does start to plateau starting at 200 steps per training epoch as the models start to converge.

training epochs. We note that a single 200-step training epoch of updating network heads took approximately 3 minutes compared to a 5 minute epoch when updating the entire network. More importantly, training the entire network leads to only marginally improved model convergence. Given the extra time cost and statistically insignificant benefit, it is clearly valuable to train only the network heads and employ transfer learning when using small, sub-thousand-element training sets, as commonly encountered in the materials sciences where the cost per labeled training artifact is non-negligible.

In Table 4.1 we report performance differences found between the two pre-trained network weights utilized in this study, the ImageNet Large Scale Hierarchical Image Database [48] and the COCO, Common Objects in Context large-scale, object detection, segmentation, and captioning dataset [49]. Both COCO and ImageNet are integrated within the Matterport Mask-RCNN implementation [47]. Note that each AP score reported in Table 4.1, along with every AP score reported in this study, consists of the average scores of 10 models each trained on a different cross-validation fold of our floating zone dataset. We also report one standard deviation of the AP performance of those 10 models. We found that with identical training configurations the COCO pre-trained network consistently scored a few points better than the ImageNet weights, although the performances are within a

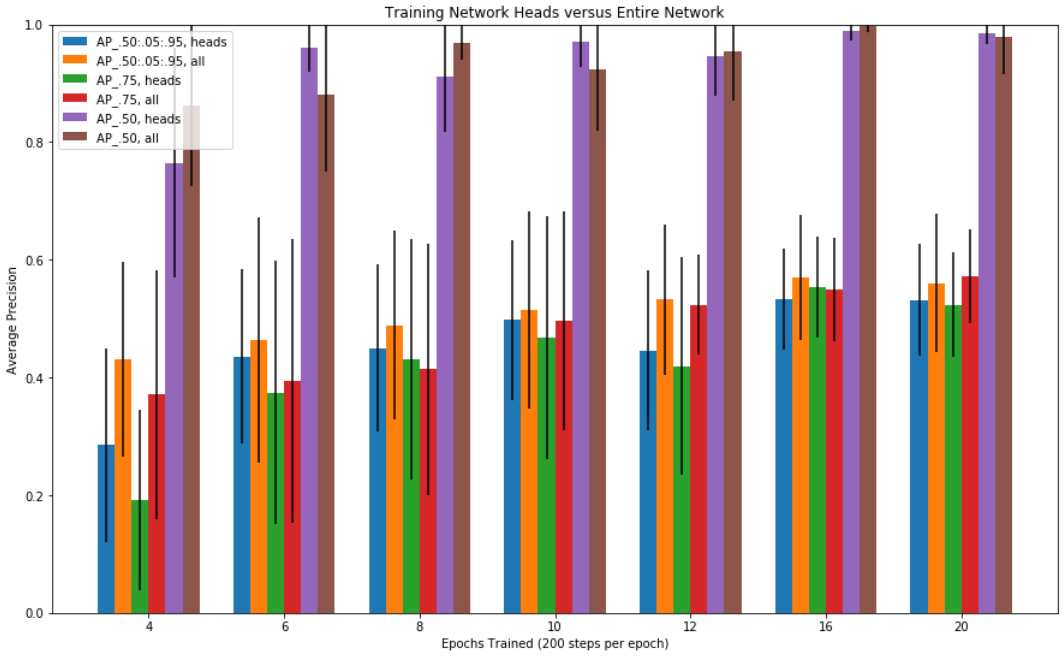

Figure 4.4: Comparison of model training of only first-layer weights (network heads) of the R-CNN or additional training to update weights of the entire network. In each result presented here transfer learning is employed and derived from pre-training. Not shown in this figure is the additional training time cost of allowing updates to the entire network; on average, training just the network heads took 58% as long as training the entire network. Similar to the results shown in Figure 4.3 there is a decrease in model variance and a slight increase in average precision as the number of training epochs increases. Full network training gives only marginal benefit in performance and variance compared to training only the network heads.

Table 4.1: ImageNet vs COCO Pre-Trained Networks Effect on Average Precision

Network/Epochs	$AP_{50:.05:.95}$	AP_{75}
ImageNet/12	**0.53** $\pm$0.13	**0.52** $\pm$0.09
COCO/12	**0.54** $\pm$0.07	**0.52** $\pm$0.08
ImageNet/16	**0.57** $\pm$0.11	**0.55** $\pm$0.09
COCO/16	**0.58** $\pm$0.09	**0.58** $\pm$0.07
ImageNet/20	**0.56** $\pm$0.12	**0.57** $\pm$0.08
COCO/20	**0.59** $\pm$0.08	**0.56** $\pm$0.05

standard deviation of one another. The variance of the COCO scores is also slightly lower, however, suggesting a faster model convergence.

All of our evaluations consistently show significantly higher AP_{50} scores relative to AP_{75} and $AP_{50:.05:.95}$ scores. The difference in scores is due to the model-produced mask quality relative to the ground truth annotations; strong AP_{50} scores indicate that the floating zone "stable melt," "fast top," and "fast bottom" classifications are accurate, but raising Intersection-over-Union thresholds from 0.5 to a more stringent 0.75 results in a portion of AP_{50} True Positives being counted as AP_{75} False Positives. Therefore in order to improve AP_{75} and $AP_{50:.05:.95}$ scores relative to AP_{50}, one must focus on actual improvements by tuning mask quality.

4.2.2 Model Usage in Practice

The speed of our model facilitates deployment in two, distinct applications: 1) classification of archived floating zone data, and 2) real-time clas-

sification of current floating zone melts in the PARADIM streaming-data platform [50].

Figure 4.5 shows application of the model to video data spanning 3 months of work and including over 825,000 frames on the laser-diode furnace. Figure 4.5 depicts molten-zone classifications, but also contains blank areas representing clusters of frames that the model was unable to classify; many of these blank sections are expected since the archived video includes warm-up and spin-down portions of experiments which either lack molten zones or include geometries outside our currently trained classes. Classifications with confidence scores below 0.3 are considered as non-classified or unrecognized frames and so dropped from the plot.

There is a clear relationship between time and molten zone class shown by grouping of colors, supporting the validity of our trained model. Within those groupings are multiple 'V'-like patterns associated with transitions between molten zone classes. These patterns are drops in model confidence during transitions caused by a lack of transitional melt-geometry classes. While improving recognition of geometries during transitions would be possible with appropriate training, the current model's sustained drop in confidence is itself an effective alert to developing changes in molten zone-geometry state.

Towards the end of each chunk of classified frames there are sudden

drops in confidence and consistent recognition of non-stable melt classes. These sections are the end of individual growth sessions where the growth becomes unstable and separates. Our model is able to recognize these shifts as the furnace is spooled down.

Further, all materials represented in these historical data are chemically distinct from that used in the model training. This demonstrates one of the most important benefits of ML/AI over other automation techniques: generality, without needing per-material training, and, indeed, without even knowledge of what material is being grown.

Additionally, application of our trained model creates value from archival data by producing insights that the data was not originally collected to provide. Specifically, figure 4.5 includes 240,000 'stable melt' classifications with an average confidence score of 0.65, 40,000 'fast bottom' classifications with average confidence of 0.6, and 12,000 'fast top' classifications with average confidence of 0.5. These data give a previously unknown overview showing that working in collaboration with FZ specialists is highly effective as $\sim$82% of growth time operates in a stable molten zone mode. It also shows that times of unstable melts are dominated by over-rotation of the growing crystal ($\sim$14% of all experiment time classed) and over-rotation of the feed rod is uncommon ($\sim$4% of experiment time).

In addition to retrospective insights, the model is also usable to guide

and accelerate identification of optimal growth conditions. Figure 4.7 is a screenshot of a user dashboard used during FZ sessions and an example of this second application. On the left side of the dashboard is a display of live video frames from the furnace with overlays of molten zone masks created by our deployed model. On the right side a scrolling plot of model-confidence versus time is colored by object class providing the experimentalist with up-to-the-second feedback on melt condition. These plots alert the experimentalist of a need to modify furnace parameters when model confidence in classification dips.

4.2.3 Labeled Training Data

An important outcome of this study is production and publication of a labeled training set for those seeking to reproduce or create their own models. To train our model we produced a dataset of 990 class-labeled, polygon-annotated images with 330 examples per class. In the dataset, each image is: 576x432 pixels, with a size of 729 Kb. Every training image has a paired .json file containing the pixel coordinates of the labeled molten zone polygon created. The dataset is published [51] and freely available with the public DOI: https://doi.org/10.34863/41cn-4361.

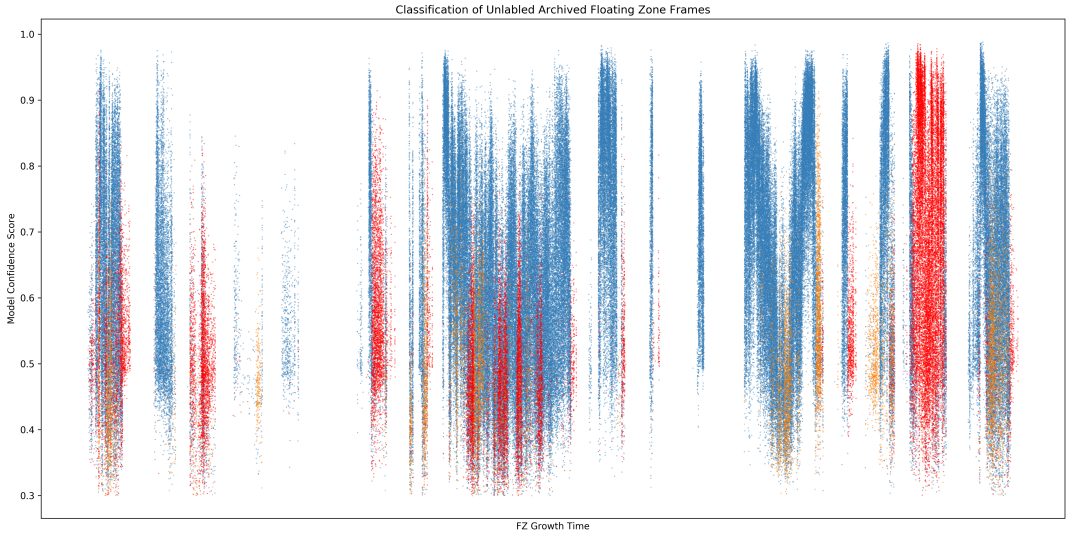

Figure 4.5: Classifications and confidence scores of 825,000 archived floating zone frames. Blue denotes frames classified as ”stable melts,” red are ”fast bottoms,” and orange are ”fast tops.” Blank areas represent model-unrecognizable frames. The frames are ordered sequentially by time stamp, however this large dataset encompasses numerous growth sessions stitched together. These data represent multiple growth attempts for three materials, all of which are distinct from the material utilized during model training.

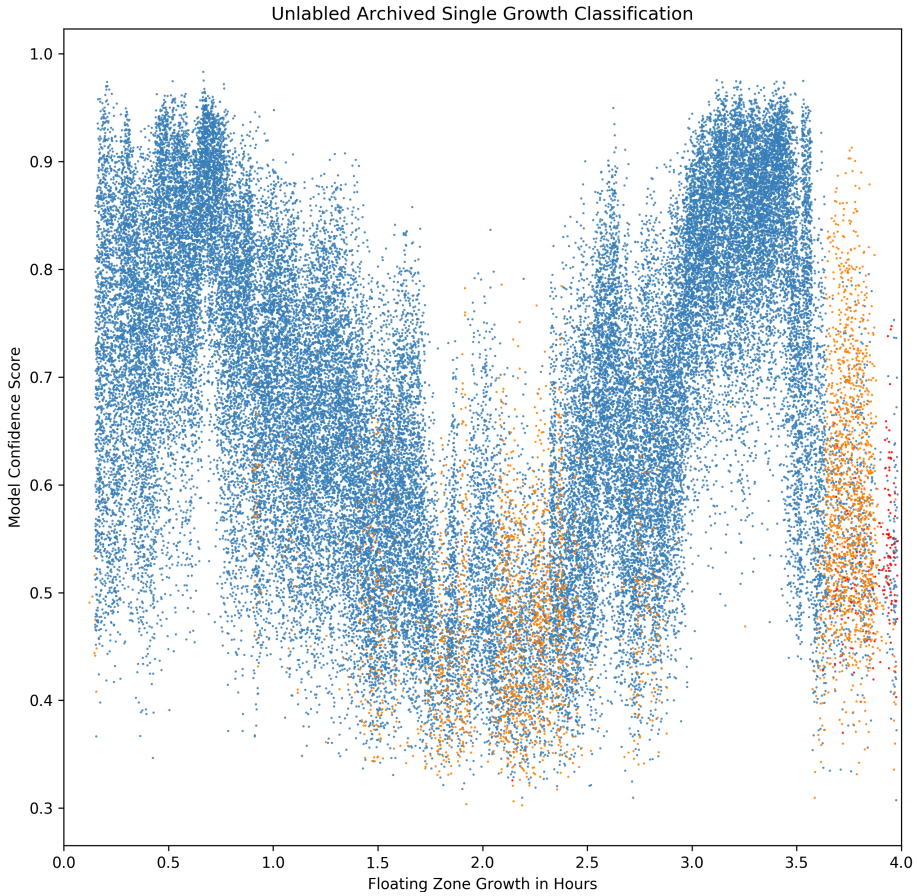

Figure 4.6: A detailed view of one of the growth sessions depicted in Figure 4.5. Note the dip in model confidence as the growth transitions between states. At the end of the session the melt becomes unstable and disconnects; the model recognizes the start of this event as unstable states.

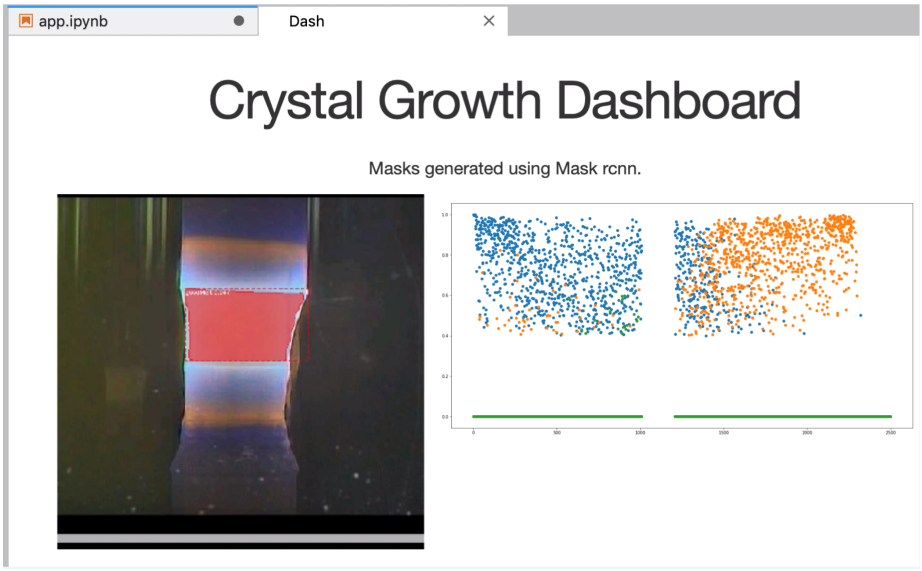

Figure 4.7: Screenshot of the live online dashboard prototype. Updating once per second, the dashboard displays the current growth state with a classifier-produced mask on top. Also displayed is a graph showing the recent (1 hour) history of the current session, with the y axis being classifier confidence and the color being the classifier-detected class of the growth. Depicted on the right is the history of a user manipulating the growth into a different state, with the classifier recognizing this transition between classes over time as the color (class) of the live updates change from blue to orange (”stable melt” to ”fast top”)

4.2.4 Conclusions and Future Work

Development and training of a deep neural net in instance segmentation of FZ molten zone geometry provides unprecedented speed and accuracy to evaluate and classify experiment state. Application of the model provides immediate feedback to experimentalists charged with producing high-quality single crystals during time-limited usage of PARADIM facilities. The model is an effective means to derive insight from the complexly encoded Big Data comprising years of archived video of FZ synthesis. The model is a foundation for future work to deconvolve the high-latency relationships between furnace parameters and molten zone geometry as a basis for developing automated synthesis and perhaps optimization of crystal quality. Even though we demonstrate that an effective model may be constructed with small amounts of training data, we do not discount the value of expanding the training dataset. Driving down the cost of creating more training data is essential, and making use of technology such as Google's Fluid Annotation [52] could dramatically reduce such labor costs. We are currently expanding our training set, applying the technique to the full wealth of historical data from expert and non-expert growths, and exploring hyper-spectral imaging to improve model performance and capability.

4.3 Methods

The model was trained on data collected from boron carbide single crystal growths as described by Straker et al. [53]. The historical data of figure 4.5 corresponds to data collected during growths between November 2019 and February 2020. All image data were collected from PARADIM's tilting laser diode floating zone (TiltLDFZ) furnace (Crystal Systems Inc FD-FZ-5-200-VPO-PC) with 5 200 W GaAs lasers (976 nm), utilizing the real-time data pipeline built on custom-built, open sourced tools [54]. A notch filter to suppress laser contamination of image data was used. Frames were cropped before labeling or classification. Instance segmentation utilized the Mask R-CNN framework [45] trained using the manually labeled dataset described below. Mask R-CNN is a deep neural net that extends Faster R-CNN by incorporating a branch for predicting segmentation masks on each region of interest (RoI) in parallel with the existing branch for classification and bounding-box regression. The mask branch is a small, fully convolutional network which when applied to each RoI predicts pixel-by-pixel masks. This mask branch adds a small overhead, but Mask R-CNN remains similar in implementation and training to the widely used Faster R-CNN framework [45]. For this study, we utilized the freely available Matterport implementation of Mask R-CNN [47]

4.3.1 Dataset

With the goal of recognizing awry floating zone states, we created a training dataset for the Mask R-CNN learner. Building a Mask R-CNN model to identify new objects, or classes of objects, requires training and tuning with annotated image examples of each new object. For the reported model, we identify three basic classes of molten zone: ”stable melt,” ”fast bottom,” and ”fast top”. The ”stable melt” class refers to a stable molten zone geometry capable of sustaining growth without modification. Classes labeled ”fast bottom” and ”fast top” embody unstable molten-zone shapes that occur when the top or bottom shafts of the furnace translate too quickly. When recognized, these unstable states can be converted to stable geometries by small corrections of shaft speeds. Our study focused on training using data from the synthesis of large, single-crystal boron carbide [53] in a laser-diode furnace where only these three classes dominated states.

To create an effective training set, we recorded live footage of the molten zone while manipulating the furnace to evoke the three class states. In the floating zone training experiment, we collected 600 frames (10 minutes) of each class as an expert operator alternated from stable to unstable states. Video clips were sliced into JPEG frames at 2 second intervals and sorted into classes by PARADIM experts. Images were ”landmarked” with pixel coordinates outlining the relevant molten-zone using LabelMe [55, 56].

4.3.2 Training Methods

Prototype models showed some success at classifying molten zone states, but suggested training regimen and model evaluation experiments to relate performance data to model accuracy. Here we report method details and explain concepts fundamental in training and measuring performance.

4.3.2.0.1 Hyper-parameter Tuning

The Mask R-CNN framework provides a number of options that condition the effectiveness of the trained model. Variation of these hyper-parameters, within the Config class of the Matterport implementation [47], can lead to significantly different model performance despite training on the same dataset. Many of these hyper-parameters are straighforward options determined by the properties of the training data, but some training parameters benefit from tuning in an experimental fashion. We evaluated hyper-parameter tuning of the number of training epochs, number of training steps taken per epoch, training network heads versus training the entire network, and making use of pre-training with either the ImageNet Large Scale Hierarchical Image Database [48, 57] or the COCO, Common Objects in Context large-scale, object detection, segmentation, and captioning dataset [49].

4.3.2.0.2 Transfer Learning

Our molten zone learner utilized pre-trained network weights from the COCO and ImageNet datasets provided in the Matterport implementation [47]. This transfer learning pre-encoded general object recognition into the network to compensate the small size of our training dataset and reduce the number of required training epochs. This provided a substantial speed gain compared to training a bare model. He et al. [45], for example, needed over 44 hours of training on eight GPUs for a dataset of 120,000 images. In contrast, our training regimen is two orders of magnitude less time intensive while run on a single GPU. Recently, He et al. have evaluated the value of pre-training in instance segmentation and recommend it when working with a dataset less than 10,000 elements. [58]

4.3.2.0.3 k-Fold Cross-Validation

Cross validation is a method of resampling training data to reduce bias. The method also maximizes the size of the training while still reserving a validation set from the complete training set. We specifically apply k-fold cross-validation to more accurately show the effect of the Mask R-CNN hyper-parameters on the trained model fitness. [59]

In k-fold cross-validation, one partitions the labeled training data into k subsets of equal size. For each subset, a model is trained with that subset

serving as the validation set and the rest of the labeled data as the training set. Each of the k models are evaluated for fitness and their resulting scores are averaged together. The advantage of this method of cross validation is each piece of labeled data equally serves as validation data exactly once and as training data $k - 1$ times.

For our evaluations we use 10-fold cross-validation; each of the 10 folds of our dataset reserves 810 labeled images as training data and 90 as validation. Our test set is separate from the cross-validation training/validation sets and consists of 90 images.

4.3.3 Evaluation Method

Given a hyper-parameter selection, we train a separate model for each of the 10 folds of our 10-fold cross validation split of our dataset. Following training, each of the 10 models are tasked with classifying the 90 images of the test set. We then report the average and standard deviation of the Average Precision measurements of the ten models relative to different hyper-parameter selections.

Following the PASCAL Visual Object Classes (VOC) 2007 challenge [60], the ”average precision” (AP) [61] metric became the standard for image object segmentation evaluation. The COCO image segmentation challenge expanded upon the VOC AP metric to include Average Precision at several

mask Intersection-over-Union (IoU) thresholds. [49] The standard COCO metrics that we use to evaluate our model include AP_{50}, AP_{75}, and $AP_{50:.05:.95}$. AP_{50} is the average precision measured with an IoU threshold of .5, AP_{75} is the same but with an IoU threshold of .75, and$AP_{50:.05:.95}$ is the average of AP measurements taken at all IoU thresholds between .5 and .95 in .05 increments.

Average precision summarises the shape of the precision-recall curve of a model's performance when executed on a given test set. After the model performs classification, each element in the test set is either a True Positive, False Positive, or False Negative. There are no True Negatives since each test set element has an object to classify within the image. A False Negative is when the model detects nothing. A False Positive denotes an incorrect classification; for example, in our case a False Positive would be when the model identifies a molten zone as the wrong class. Finally, a True Positive is a correct classification. Precision and recall are derived from the number of True Positives, False Positives and False Negatives:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

4.3.3.0.1 INTERSECTION-OVER-UNION

(IoU) threshold is an additional criteria for a classification to be considered a True Positive. Meant to quantify the quality of a model-produced mask, IoU is the intersection of the model mask pixels with the ground-truth label pixels divided by the union of the mask and ground truth pixels. It is a measure of how similar the mask is to the ground truth annotation of the molten zone. Given an IoU threshold of .5, a classification is considered a True Positive only if the associated mask has an IoU over .5; otherwise, it is considered a False Positive.

$$IoU = \frac{Mask \cap GroundTruth}{Mask \cup GroundTruth}$$

4.3.3.0.2 PRECISION-RECALL CURVE

With precision and recall defined, the next step towards calculating AP is constructing a precision-recall curve. For each model classification of the test set sorted by confidence score in descending order, points on the precision-recall curve are found by determining whether the current classification is a True Positive or False Positive or False Negative, while plotting and updating the current precision and recall at each step. Note that the denominator in each recall calculation is a fixed value representing all relevant documents; in our case, it is the size of the test set.

4.3.3.0.3 Average Precision

Following the PASCAL VOC 2010 challenge, the average precision measure was changed to sample the precision-recall curve at all recall values, rather than sampling precision at 11 fixed recall levels. The PASCAL VOC 2012 Development Kit [62] defines the Average Precision calculation as follows:

1. Compute a version of the measured precision/recall curve with precision monotonically decreasing, by setting the precision for recall r to the maximum precision obtained for any recall $r' \geq r$

2. Compute the AP as the area under this curve by numerical integration. No approximation is involved since the curve is piecewise constant.

4.3.4 Training Environment

The model was prototyped and trained using the SciServer, NSF DIBB, cloud platform accessible in a browser [2]. We developed and executed the Mask R-CNN training code on SciServer Compute, a Docker container host with a Python Jupyter notebooks interface and bulk file storage. Database storage for the dataset and training experiment results utilized SciServer CasJobs. The GPU jobs queue within SciServer Compute provided access to

Nvidia CUDA and cuDNN libraries to GPU-accelerate training. We scripted experiment coordinators in SciServer Compute Docker containers to organize training datasets, schedule training jobs in the GPU jobs queue, and store results in the CasJobs database for further analysis.

4.3.5 Hardware Performance Notes

We ran the Mask R-CNN classifier on a variety of hardware setups and have collected some general observations. Matterport's Mask R-CNN implementation is built using Tensorflow and Keras with Nvidia's CUDA libraries allowing execution on either a CPU or a GPU [47]. Despite our relatively light regimen of typically a dozen training epochs with 200 steps per epoch over a 900 image training set, training any of our models using a CPU took over an hour per epoch. Training on a single Nvidia Tesla K40 GPU averaged three minutes per training epoch for the network heads and five minutes per epoch when updating the weights of the entire network.

Using a CPU to run an existing Mask R-CNN model in inference mode, however, was more reasonable. Our initial floating zone live feed classification prototype ran on a Tensorflow CPU-build and averaged one-image-per-second classifications. To compare this to GPU-executed classifier performance, we developed a benchmark test that records the time it takes the model to classify 900 images. We ran this benchmark with Tensorflow

running on two, single-GPU platforms. The first utilized an Nvidia Tesla K40 GPU with 12 GB GDDR5 memory, 2880 CUDA cores and Kepler architecture. The second platform had a Nvidia Quadro 8000 GPU with 48 GB GDDR6 memory and 4608 CUDA cores with Turing architecture. The K40 was able to classify the benchmark set in 230 seconds, or 3.9 images per second. The Quadro GPU could classify the set in 61 seconds, or 14.75 images per second or 3.78 times as fast as the K40. We found the performance of our usage of the Matterport implementation in line with the reported performance of Mask R-CNN on the COCO challenge, which was about 200ms per frame on a single GPU. [45]

Chapter 5

A Brain Computer Interface for Human-Machine Co-Learning

Perception and understanding of large scale data (PULSD) in an exponentially accelerating world is a difficult but important and interesting challenge. To tackle it, we will have to advance our capabilities on all fronts from machine learning via data rendering to applied brain sciences. Here we describe our work revolving around a novel synthesis of the aforementioned key concepts into a unique framework for cooperative learning: a visual dataset navigation system where both a search and classification of the data is directed by an analyst's real-time interest and subconscious intuition. Both the volume and the dimensionality of todays data are enormous; hence the efficiency of the interactions becomes a critical issue. We demon-

strate the power of our initial prototype, a proof-of-concept on a dataset search using an EEG Brain-Computer Interface and an iterative Rapid Serial Visual Presentation (RSVP) technique, and describe the challenges and results so far.

5.1 Introduction

Our understanding of our dynamically changing surroundings is increasingly limited by our ability to perceive the available data. When we are able to identify relevant features and quickly interpret them in their observed contexts, we can achieve the high level of awareness that is required to take action in case of new events. The same way we navigate when walking on the street or driving a car, we now need to learn to navigate in digital data. As we collect exponentially more samples from the physical and cyber worlds, our chances of keeping up with the data rate are slimming rapidly. Today, data analysts almost exclusively rely on machine learning techniques. These applications transform the raw data into feature spaces that capture different aspects of the input stream. While often existing methods cannot scale to the current large volumes of data, recent theoretical advancements of randomized algorithms are enabling approximate variants to be directly applicable to large-scale data. This, however, is

not enough! Here we argue that this incremental improvement in machine learning is only a small step toward the goal of rapid knowledge extraction.

The number of methods available for knowledge extraction has reached a point where making the correct choice of method by itself becomes a major issue. Some perform better on certain types of complex problems, while others are more robust to omnipresent artifacts. The purpose of these different low-dimensional embeddings of the raw data is to aid the learning experience and to speed up discoveries, but the aggregate output of all these methodologies is again too much for anyone to fully comprehend. Computers can perform any one of an infinite number of possible projections or non-linear mappings, but unsupervised programs do not know where to look in the data, and cannot tell obvious patterns from emerging discoveries. Yet, analysts must constantly assess the validity, correctness and relevance of these results, while they attempt to fully understand the situation reflected in the data. Our goal is to assist such analysts.

We work on a future integrated approach for human-machine co-learning, where immediate feedback from the operator can drive data rendering by steering through the parameter space of possible embeddings and features. Monitoring the analysts behavior and responses will provide invaluable information about ongoing thought processes. Using eye-tracking and EEG technology, we capture signals and correlate them with the rendering. Al-

gorithms can automatically render feature vectors into a familiar environment, where the analysts will subconsciously make the right choices without effort and without blurring the results with false rationales that may be based on selective or insufficient facts. In such interactive brain-computer systems the conversion of data to knowledge will be a natural and optimal process.

5.2 Background

The field of brain-computer interfaces (BCI) has exploded in the past decade. The appeal is clear: the process of ingesting, interpreting and reporting on a piece of data is limited in many cases by the physical activities involved, for example, reading words on a computer screen or typing on a keyboard. In addition, the classic methodology includes several layers of abstraction and categorizations from thoughts to words to written language and reverse, all of which add placeholders and thereby reduce the accuracy of the information. If it were possible to replace the act of reading and typing with a direct link between ones brain and a computer, an incredible speedup could be realized.

In fact, we apply practical technologies today to achieve this through non-invasive neural interfaces such as electroencephalography (EEG). The

operational principle is simple: conductive electrodes make contact with the scalp, and pick-up microvolt-scale electrical potentials created by many simultaneously active neurons in the cortex. The electrical nature of the EEG signal allows for high temporal measurement precision. Clever designs for user interfaces recently transformed EEG to an interactive technology.

A particularly striking example is the paradigm called Rapid Serial Visual Presentation (RSVP) [63]. A user is presented with a series of images in rapid succession while their neural activity is monitored. If the user looks for a particular type of visual content in the image stream, the brain signals following the matching content will differ from the baseline behavior. This difference, called P300 due to its positive polarity and 300-millisecond latency, is reliably detected by digital signal processing techniques, and can be traced back with high precision to the triggering image. Most importantly for the purposes of BCI, because the P300 is a product of the subconscious processing of visual content, it is manifested in the absence of any physical response and can be elicited at image presentation rates far exceeding those to which a human could normally respond. At a presentation and processing rate of up to ten images per second, this is approximately ten times faster than what could be achieved without BCI. Indeed, this capability has not been ignored within the community; outside of medical fields, RSVP has largely been employed for image categorization and surveillance

purposes [64].

It is known that sensory stimuli outside of normal patterns further elicits autonomous nervous system responses, e.g., the pupil reflex, the eye lid reflex and scanning eye movements. These reflexes include electromyogram (EMG) and electrooculogram (EOG) signals in addition to action potentials of the peripheral portions of the cranial nerves outside the skull, and these are more easily accessible through skin recordings of bio-potentials. It may be possible to identify specific composite bio-potential signatures based on the temporal sequence of discrete responses at the different frequency bands that are analyzed.

Advanced BCI technology has great significance for interactive data exploration for several reasons. One is obviously the speedup to be gained by monitoring event-related evoked potentials, but another is potentially just as important. We have an opportunity to map independent wave bands in the alpha and beta ranges to any kind of user action. In other words, we can practically grow several brainfingers and learn to use them to steer the visualizations toward interesting features. A subconscious random walk in several dimensions could map out more details in data than thousands of mouse clicks. This is high-dimensional navigation without the complication of the missing human intuition in high dimensions.

5.3 EEG Signal Processing

As EEG signal responses are collected, they must be related to the image viewed at the time of signal collection and then classified as containing a P300 wave or not containing a P300. Continuous EEG signal is sliced into epochs roughly 600ms long and synchronized with the RSVP frequency. Our system uses LabStreamingLayer (LSL) [65] to coordinate image presentation with collected EEG signal.

The presence of a P300 wave within an epoch of EEG signal is not easily discerned. EEG signal is inherently noisy due to the sensitivity of the electrodes. Previous classifiers needed the average of several responses to the same image, or stimulus, to cancel out the noise clouding the P300 signal. However, recent advances in convolutional neural network classifiers have both increased the accuracy of P300 detection and lowered the amount of training needed for employing the signal classifier. Our system uses the EEGNet [66] classifier to detect P300 waves, and in practice we have found it far superior to the traditional Linear Discriminant Analysis (LDA) classifiers.

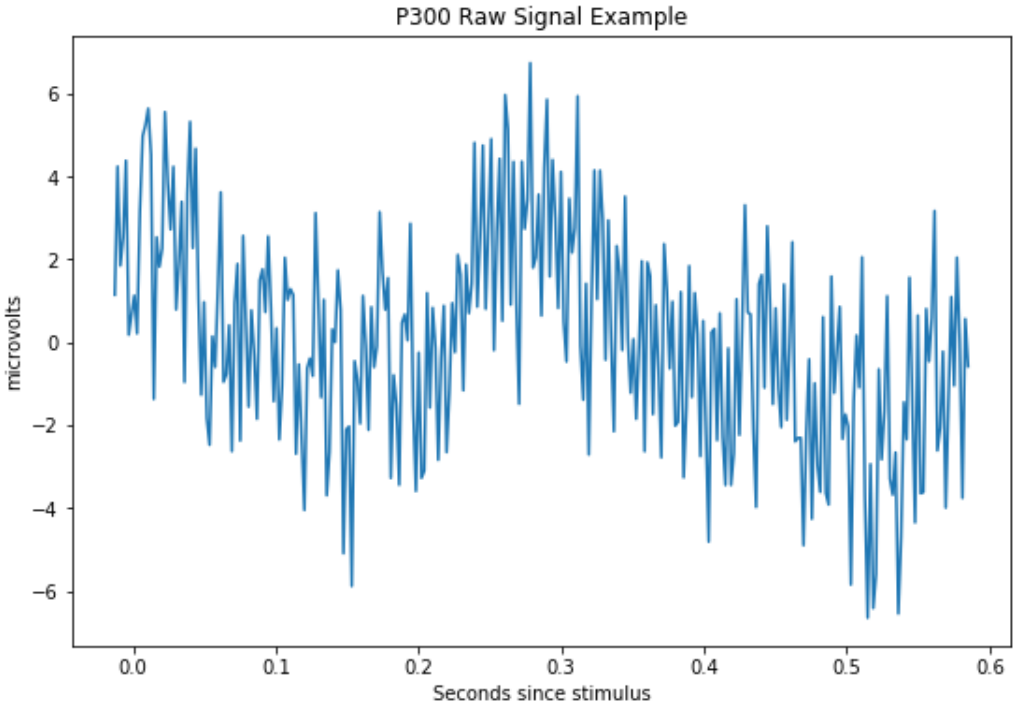

Figure 5.1: An Example of a P300 reaction to a 'target' stimulus presented at time 0. Pictured is the raw signal from 16 electrodes averaged together. Note the positive deflection at 300ms after the stimulus presentation

5.4 Data Exploration using PULSD Prototype

In this section we introduce how our system will be used by an analyst to explore a data set. We present the set of problem space assumptions that indicate if a data set is a good candidate for being explored via the PULSD prototype, followed by how iterative RSVP can provide a dataset search solution.

5.4.1 Problem Space Assumptions

Suppose an analyst has an arbitrary dataset. Also, suppose that it is possible to render images of this data set such that human insight may be gained by quickly viewing them. In the rendering process, lets assume that an image I, is rendered using some parameters θ (Figure 5.2). For example, if we have high-dimensional data, we could generate 2D visualizations of the data by selecting various rotation parameters. In this example, θ would contain those rotation parameters. In another example using netflow data, θ could contain binary values indicating if certain types of traffic should be visible or not. The assumption we are making is that an analyst does not know the correct set of parameters to investigate the dataset. There may

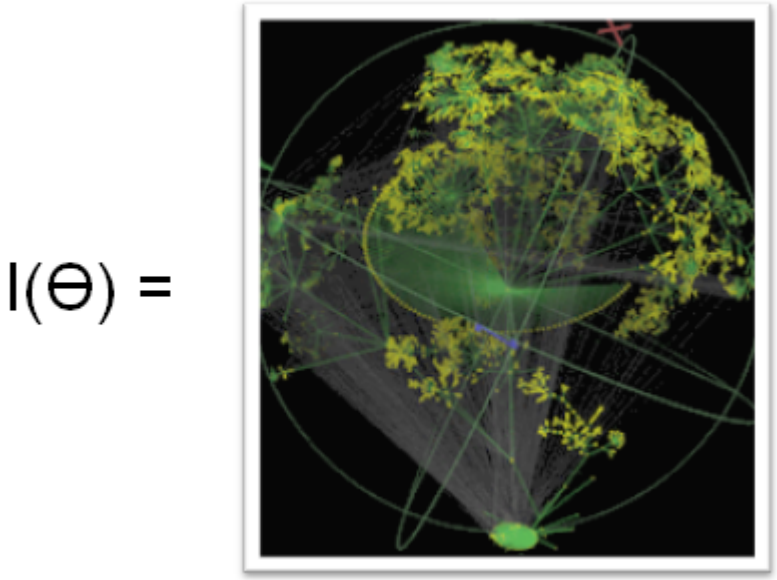

Figure 5.2: An Image I is generated using parameters θ

be some anomalous data points or places in the data where some structure may indicate an event. We also assume that finding an event in the visualizations is non-trivial. Furthermore, we assume the parameter space is too large to investigate all possible combinations of parameters.

Let us also assume that an algorithm cannot find images that contain something interesting; however, an experienced analyst can. This requires a human in the loop for the data exploration.

5.4.2 BCI for Data Exploration

The current PULSD system operates in two phases: training and testing. In the training phase, the system learns the subjects P300 response

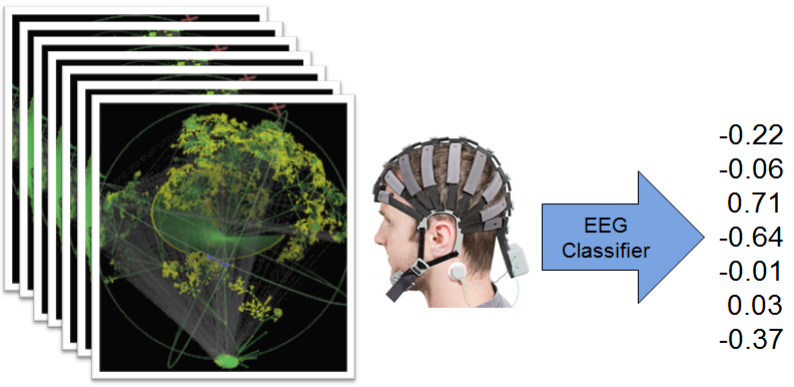

Figure 5.3: The PULSD system converts images into scores for ranking and processing

by recording the EEG signals of both visual 'targets' and 'distractors'. The target images are hand selected to be 'interesting' images, or images that should generate a P300 response. The distractor images are images that should not generate a P300 response for the subject. By recording and aggregating the EEG signals from these two classes, PULSD learns the subjects specific P300 response.

Then, during the testing phase, we present images for which we do not know if the image is a 'target' or 'distractor.' PULSD then compares the subjects EEG response to the learned target and distractor signals. The system generates a score based on this comparison indicating if the subjects response to this image is more like a target or a distractor response. In this way, PULSD can take an image and generate a score, where greater values indicate that the image is a target and lower values indicate that the image is a distractor (Figure 5.3).

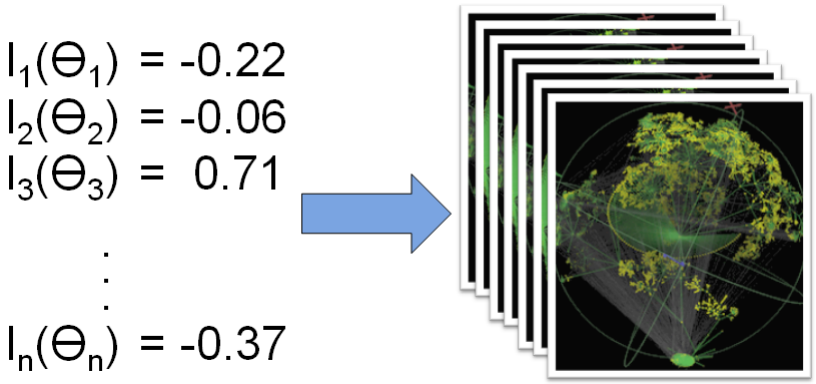

Figure 5.4: PULSD ranks the images by the subject's interest, and generates more images like the highest ranked images using the θ parameter.

Using this technique, we can present a random set of images to a subject and rank the images by how interested the subject is in them. PULSD can then select the top N ranked images and use those to develop a new set of images for presentation. Because each image in the original set was generated by a set of parameters θ, we can use an algorithm to generate similar images, based on θ for each highly ranked image (Figure 5.4).

The new set of images can be mixed in with more randomly generated images, so as to not limit the data exploration to the initial random set, and presented to the subject in the system again. This process is free to repeat as many times as desired or until a stopping criterion is met.

5.5 Iterative Data Exploration Experiments on Hidden Cube Dataset

The Hidden Cube Dataset is a simulated dataset generated by performing a high-dimensional rotation on the wireframe of a three-dimensional cube mixed with noise in the remaining few dimensions. The low-dimensional cube signal is hidden such that scatter plots of the resulting rotation simply show a noisy blob. By randomly rotating the simulated dataset, it is possible to happen upon some of the hidden cubes structure; however, most random rotations show only noise. The goal of exploring this dataset is to find a rotation of the simulated dataset that most clearly reveals the hidden cube structure. For experimental evaluation purposes, we calculate each random rotations principal angle with respect to the transpose of the rotation matrix initially used to create the simulated dataset. The principal angle is the minimal angle between two subspaces: the signal and the 2D view. In a practical sense, the principal angle tells us how close we are to finding the rotation matrix that reveals the complete hidden cube structure; a rotation with a low principal angle generally creates an image that shows more of the hidden cube structure while a high principal angle rotation shows mostly noise, as an example shows in Figure 5.5. We use this principal angle metric to quantify our experiment progress on finding the hidden cube

structure from iteration to iteration.

We use the results of an RSVP iteration to determine what subsections of the dataset to explore next, as explained in the previous sections. For the Hidden Cube Dataset, we randomly perturb the rotation matrices associated with the triggered images in order to create similar rotations to show in the next iteration. Some of the randomly similar rotations will show more structure while some will show less. Our experiment methodology is as follows:

1. Generate a starting set of 400 rotations with a fixed amount of targets and distractors. Our control variable is the ratio of targets to distractors in the starting set of random rotations.

2. Record the average Principal Angle of the current set of rotation images for this iteration. This is the dependent variable of our experiment.

3. Run an RSVP session and extract the top 20 scored images, where score denotes the match to the ideal P300 wave identified during training.

4. Create a new set of 240 images by generating 12 randomly similar rotations for each of the top 20 scored images. Some of these randomly similar rotations will show more cube structure and some will show

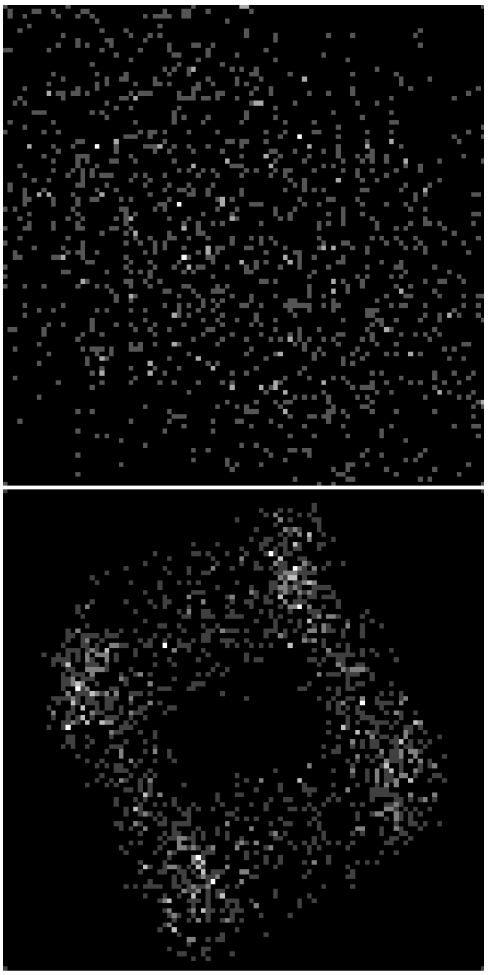

Figure 5.5: The top rotation visualization has a principle angle of 0.166 radians with respect to the rotation matrix used to hide the cube, while the bottom rotation, showing some cubic structure, has an angle of 1.349 radians

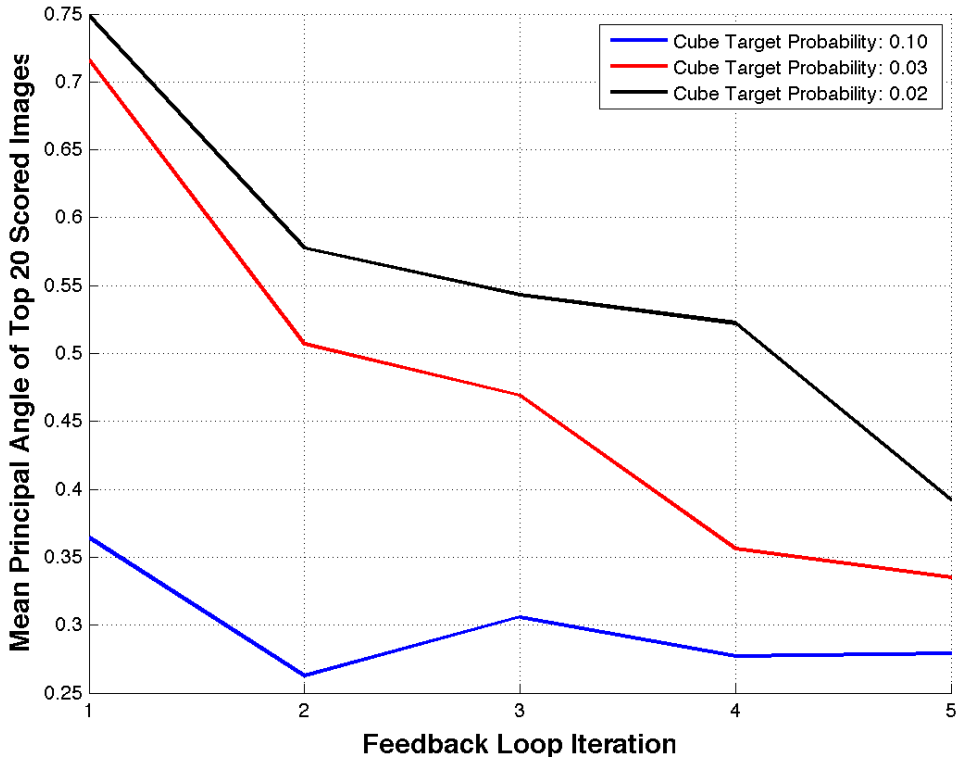

Figure 5.6: The average principal angle in radians of the top 20 scored images of each search iteration. These top 20 images were used as the basis for the next iterations set of images; for each top image, twelve similar images were shown next iteration. Cube Target Probability is the percentage of images showing some structure in the starting image set.

less.

5. Start next iteration by repeating from Step 2.

Figure 5.6 shows the results of three Hidden Cube search experiments, each consisting of five iterations of RSVP sessions. Each experiment was run with a different initial target probability; that is, the Cube Target Probability represents the ratio of rotations showing some cubic structure to rotations showing only noise in the initial set of images displayed in the first

RSVP iteration.

5.6 Discussion

The challenge of this experiment is that the subject must raise his or her P300 trigger threshold between iterations as the experiment progressively narrows down its search for the cube structure. The properties of the shown images change between iterations; as the experiment progresses, what may have been a target in the first iteration could be a distractor in subsequent iterations. After a couple iterations, it is no longer good enough to simply distinguish noise from structure. In order to further narrow down on the cube, the subject must discern between images that show a large amount of cube structure versus images that only show some basic structure as shown in Figure 5.7.

Figure 5.6 shows the average principal angle of the top twenty scored images, where score represents the match between the exhibited P300 wave during image presentation and the ideal P300 wave. The top twenty images of one iteration are used to seed the randomly similar images of the next iteration. This graph suggests that the human subject was able to adjust his or her structure threshold between iterations; as the experiment progressed the subject was able to discriminate between images that showed more

structure versus images that showed less structure. The human subject in the loop has had a positive impact on guiding the search for the hidden cube, even after the initial iteration. However, the experiment with an initial set target probability of 0.10 appears to have reached a plateau after the second iteration. This may be a limit imposed by the nature of the Hidden Cube Dataset; it becomes increasingly hard for the Hidden Cube Dataset similarity function to produce randomly similar rotations that show even more structure than an already good rotation. Or, the plateau could be a result of a subjects inability to discern between more or less cube structure at a point when all the images in the set show strong structure. Experiments with more iterations will be needed to investigate just how far we can narrow down the cube structure.

5.7 Conclusion

With this study we demonstrate that the RSVP paradigm, when employed iteratively, serves as an effective method for human-directed search over a visual dataset. While RSVP is typically used for image classification tasks, it can be extended into a steering mechanism when paired with a 'similarity function' over a particular dataset. Our experiment shows that a user may dynamically modify his or her search target as the data presented

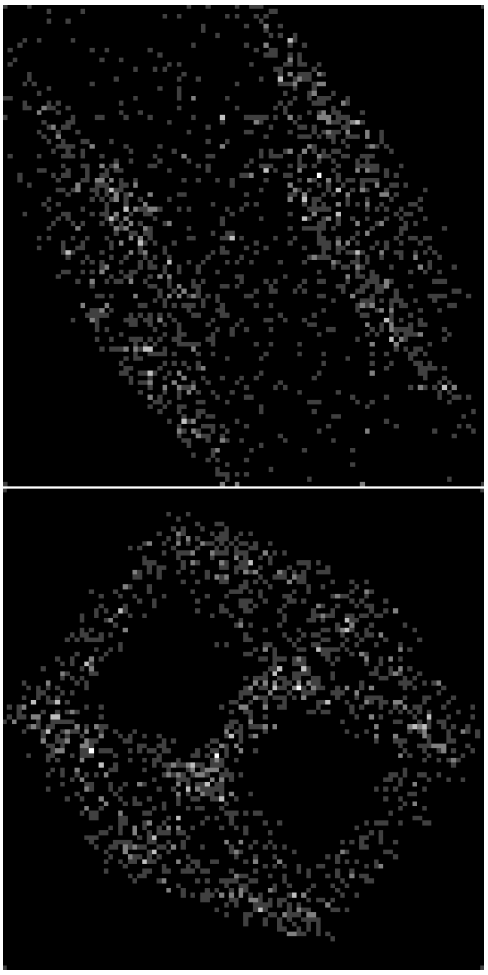

Figure 5.7: The top image does show structure and would certainly be a target in the first search iteration. However, in later iterations, the bottom image would be a target and the top a distractor. We test whether the subject can adapt to the change in prevalence of structure as the search progresses

to the user changes, thereby narrowing down on a search goal.

In this study we use the HiddenCube dataset with an easily defined 'similarity function;' in order to generate similar data, all that needed to be done was a slight, random change to the parameters used to generate the original triggered-upon image. When generalizing this system to other datasets, the similarity function will not be so cut-and-dry. However, recent advances in object recognition and image segmentation algorithms, such as the Mask R-CNN [45] algorithm, open up wide possibilities for generalizing the similarity function to other datasets. For example, when employing RSVP on natural imagery, an pre-trained ImageNet [57] or COCO [49] object recognition model could detect and recognize the subject of an image that a user found interesting. Subsequently, the next batch of images shown in iterative RSVP would feature other examples of the object previously identified. Indeed, the goal of this work is to bring the user closer to data that he or she finds interesting; while an iterative RSVP search is an essential component of this goal, the next steps involve incorporation of generalized, online machine-learning techniques over user-identified, 'interesting' data.

Chapter 6

Conclusion

As datasets continue to grow, so must our capability to extract intelligence from them. Each of the chapters in this dissertation show examples of applied data navigation. While some examples, such as in the distributed top-k elements algorithm outlined in Chapter 3, depict tools for answering fundamental data queries over a data-center scale storage system, other chapters explore techniques for bringing the human user closer to the data at hand via machine guidance. Data relevance is the goal; in the Chapter 2 Molecular Dynamics Database a reinforcement learner guides the MD simulation to relevant interactions, while in Chapter 5 we explore the human definition of relevant data by measuring user reactions to data through a Brain-Machine Interface (BMI). Each of these projects works towards bridging the gap between the data, the user, and the machine learning techniques

in between.

However, there are still significant connections to be made between the projects outlined in this thesis; while the BMI in Chapter 5 proved to be successful in directing a search over an experimental dataset, the prototype remains to be generalized into concrete application. One such application could be training set refinement of the floating-zone (FZ) furnace crystal growth classification model detailed in Chapter 4. The next step in improving that model is selecting more training data to include in the next iteration. The Mask-RCNN model I trained in Chapter 4 was applied to roughly 800,000 unlabeled historical FZ images. Ideally, one would seek out images in the historical set that were classified incorrectly but with high model confidence; labeling and including these images in the next training set would correct the model where it most needs correction. Yet, at this scale, finding these ideal corrective images by sorting through them manually is akin to looking for a needle in a haystack. The BMI for the perception and understanding of large scale data, shown in Chapter 5, would be a perfect tool for finding these corrective images and refining the FZ training set. An expert FZ furnace operator would be shown, via RSVP, a set of the model-classified historical FZ images identified as belonging to the same melt class. While the model likely classified the majority correctly, any mis-classifications would stand out to the user and evoke a P300. The

BMI would then tag that image as an incorrect classification and endeavor to include FZ images with similar classification and pixel properties in the next batch of RSVP, potentially uncovering error trends in the FZ model.

By speeding up the process through which we correct prototype trained ML-models, iterative RSVP becomes a valuable tool for human-machine co-learning. While in Chapter 4 I found that labeling training data to be the slowest roadblock in applying Mask R-CNN to new purpose, there is already a focus on developing less labor-intensive methods for annotating machine-readable training data. The Fluid Annotation [52] interface is one such project that shares similar human-machine collaboration goals. In Fluid Annotation, a neural network model makes a first pass at annotating an image while a human user corrects any mistakes. However, as models learn and make fewer mistakes, our process for finding these mistakes must keep up. Iterative RSVP and BMIs can continue to accelerate the process for which we correct and train new models as we close the gap between us and our machine learners.

Bibliography

[1] (2019) Harnessing the data revolution (hdr) at nsf. [Online]. Available: https://www.nsf.gov/cise/harnessingdata/

[2] M. Taghizadeh-Popp, J. W. Kim, G. Lemson, D. Medvedev, M. J. Raddick, A. S. Szalay, A. R. Thakar, J. Booker, C. Chhetri, L. Dobos, and M. Rippin, "Sciserver: a science platform for astronomy and beyond," 2020.

[3] (2016) Data infrastructure building blocks (dibbs). [Online]. Available: https://www.nsf.gov/pubs/2017/nsf17500/nsf17500.htm

[4] (2019) The mede data science cloud at sciserver. [Online]. Available: http://mededsc.sciserver.org/

[5] J. de Pablo, N. Jackson, M. Webb, L.-Q. Chen, J. Moore, D. Morgan, R. Jacobs, T. Pollock, D. Schlom, E. Toberer, J. Analytis, I. Dabo, D. DeLongchamp, G. Fiete, G. Grason, G. Hautier, Y. Mo, K. Rajan, E. Reed,

and J.-C. Zhao, “New frontiers for the materials genome initiative,” *npj Computational Materials*, vol. 5, p. 41, 04 2019.

[6] N. Carey, T. Budavari, N. Daphalapurkar, and K. Ramesh, “Data integration for materials research,” *Integrating Materials and Manufacturing Innovation*, vol. 5, 12 2016.

[7] D. C. Rapaport, *The Art of Molecular Dynamics Simulation*, 2nd ed. USA: Cambridge University Press, 2004.

[8] S. Chaudhuri and U. Dayal, “An overview of data warehousing and olap technology,” *SIGMOD Rec.*, vol. 26, no. 1, p. 6574, Mar. 1997. [Online]. Available: https://doi.org/10.1145/248603.248616

[9] Y. Zhuge, H. García-Molina, J. Hammer, and J. Widom, “View maintenance in a warehousing environment,” in *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD 95. New York, NY, USA: Association for Computing Machinery, 1995, p. 316327. [Online]. Available: https://doi.org/10.1145/223784.223848

[10] R. Kohavi, N. J. Rothleder, and E. Simoudis, “Emerging trends in business analytics,” *Commun. ACM*, vol. 45, no. 8, p. 4548, Aug. 2002. [Online]. Available: https://doi.org/10.1145/545151.545177

[11] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, "Mdanalysis: A toolkit for the analysis of molecular dynamics simulations," *Journal of Computational Chemistry*, vol. 32, no. 10, pp. 2319–2327, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21787

[12] Y. Ahmad, O. Kennedy, C. Koch, and M. Nikolic, "Dbtoaster: Higher-order delta processing for dynamic, frequently fresh views," *Proc. VLDB Endow.*, vol. 5, no. 10, p. 968979, Jun. 2012. [Online]. Available: https://doi.org/10.14778/2336664.2336670

[13] M. Wick, A. McCallum, and G. Miklau, "Scalable probabilistic databases with factor graphs and mcmc," *Proc. VLDB Endow.*, vol. 3, no. 12, p. 794804, Sep. 2010. [Online]. Available: https://doi.org/10.14778/1920841.1920942

[14] X. Feng, A. Kumar, B. Recht, and C. Ré, "Towards a unified architecture for in-rdbms analytics," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD 12. New York, NY, USA: Association for Computing Machinery, 2012, p. 325336. [Online]. Available: https://doi.org/10.1145/2213836.2213874

[15] A. Thakar, A. Szalay, K. Church, and A. Terzis, "Large science

databases - are cloud services ready for them?" *Sci. Program.*, vol. 19, no. 23, p. 147159, Apr. 2011. [Online]. Available: https://doi.org/10.1155/2011/591536

[16] D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, K. J. Bowers, J. C. Chao, M. P. Eastwood, J. Gagliardo, J. P. Grossman, C. R. Ho, D. J. Ierardi, I. Kolossváry, J. L. Klepeis, T. Layman, C. McLeavey, M. A. Moraes, R. Mueller, E. C. Priest, Y. Shan, J. Spengler, M. Theobald, B. Towles, and S. C. Wang, "Anton, a special-purpose machine for molecular dynamics simulation," *Commun. ACM*, vol. 51, no. 7, p. 9197, Jul. 2008. [Online]. Available: https://doi.org/10.1145/1364782.1364802

[17] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *ACM Comput. Surv.*, vol. 40, no. 4, pp. 11:1–11:58, Oct. 2008. [Online]. Available: http://doi.acm.org/10.1145/1391729.1391730

[18] S. Nepal and M. V. Ramakrishna, "Query processing issues in image(multimedia) databases," in *Proceedings of the 15th International Conference on Data Engineering*, ser. ICDE '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 22–29. [Online]. Available: http://dl.acm.org/citation.cfm?id=846218.847271

[19] U. Guntzer, W.-T. Balke, and W. Kiessling, "Optimizing multi-feature queries for image databases," in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 419–428. [Online]. Available: http://dl.acm.org/citation.cfm?id=645926.671875

[20] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '01. New York, NY, USA: ACM, 2001, pp. 102–113. [Online]. Available: http://doi.acm.org/10.1145/375551.375567

[21] S. Michel, P. Triantafillou, and G. Weikum, "Klee: A framework for distributed top-k query algorithms," in *Proceedings of the 31st International Conference on Very Large Data Bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 637–648. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083592.1083667

[22] P. Cao and Z. Wang, "Efficient top-k query calculation in distributed networks," in *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '04. New York, NY, USA: ACM, 2004, pp. 206–215. [Online]. Available: http://doi.acm.org/10.1145/1011767.1011798

[23] M. Greenwald and S. Khanna, "Space-efficient online computation of quantile summaries," *SIGMOD Rec.*, vol. 30, no. 2, pp. 58–66, May 2001. [Online]. Available: http://doi.acm.org/10.1145/376284.375670

[24] Q. Zhang and W. Wang, "A fast algorithm for approximate quantiles in high speed data streams," in *Scientific and Statistical Database Management, 2007. SSBDM '07. 19th International Conference on*, July 2007, pp. 29–29.

[25] S. Coles, *An Introduction to Statistical Modeling of Extreme Values*, 1st ed., ser. Springer Series in Statistics. Springer-Verlag London, 2001.

[26] J. P. III, "Statistical inference using extreme order statistics," *Ann. Statist.*, vol. 3, no. 1, pp. 119–131, 01 1975. [Online]. Available: http://dx.doi.org/10.1214/aos/1176343003

[27] A. A. Balkema and L. de Haan, "Residual life time at great age," *Ann. Probab.*, vol. 2, no. 5, pp. 792–804, 10 1974. [Online]. Available: http://dx.doi.org/10.1214/aop/1176996548

[28] C. Scarrott and A. MacDonald, "A review of extreme value threshold estimation and uncertainty quantification," *REVSTAT - Statistical Journal*, vol. 10, no. 1, pp. 33–60, 2012. [Online]. Available: https://www.ine.pt/revstat/pdf/rs120102.pdf

[29] J. R. M. Hosking and J. F. Wallis, "Parameter and quantile estimation for the generalized pareto distribution," *Technometrics*, vol. 29, no. 3, pp. 339–349, Sep. 1987. [Online]. Available: http://dx.doi.org/10.2307/1269343

[30] E. Castillo and A. S. Hadi, "Fitting the generalized pareto distribution to data," *Journal of the American Statistical Association*, vol. 92, no. 440, pp. 1609–1620, 1997.

[31] J. Zhang, "Likelihood moment estimation for the generalized pareto distribution," *Australian and New Zealand Journal of Statistics*, vol. 49, no. 1, pp. 69–77, 2007. [Online]. Available: http://dx.doi.org/10.1111/j.1467-842X.2006.00464.x

[32] J. Husler, D. Li, and M. Raschke, "Estimation for the generalized pareto distribution using maximum likelihood and goodness of fit," *Communications in Statistics - Theory and Methods*, vol. 40, no. 14, pp. 2500–2510, 2011.

[33] P. de Zea Bermudez and S. Kotz, "Parameter estimation of the generalized pareto distribution," *Journal of Statistical Planning and Inference*, vol. 140, no. 6, pp. 1353 – 1373, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378375809002766

[34] M. Wu and C. Jermaine, "Guessing the extreme values in a

data set: a bayesian method and its applications," *The VLDB Journal*, vol. 18, no. 2, pp. 571–597, 2009. [Online]. Available: http://dx.doi.org/10.1007/s00778-009-0133-6

[35] N. Marz, *Storm: Distributed and Fault-Tolerant Realtime Computation*. storm.apache.org, 2014.

[36] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy, "Storm@twitter," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 147–156. [Online]. Available: http://doi.acm.org/10.1145/2588555.2595641

[37] P. Wendell, *Berkeley AmpLab Big Data Benchmark*. https://amplab.cs.berkeley.edu/benchmark/, 2014.

[38] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 165–178. [Online]. Available: http://doi.acm.org/10.1145/1559845.1559865

[39] K. Alberi, M. B. Nardelli, A. Zakutayev, L. Mitas, S. Curtarolo, A. Jain,

M. Fornari, N. Marzari, I. Takeuchi, M. L. Green, M. Kanatzidis, M. F. Toney, S. Butenko, B. Meredig, S. Lany, U. Kattner, A. Davydov, E. S. Toberer, V. Stevanovic, A. Walsh, N.-G. Park, A. Aspuru-Guzik, D. P. Tabor, J. Nelson, J. Murphy, A. Setlur, J. Gregoire, H. Li, R. Xiao, A. Ludwig, L. W. Martin, A. M. Rappe, S.-H. Wei, and J. Perkins, "The 2019 materials by design roadmap," *Journal of Physics D: Applied Physics*, vol. 52, no. 1, p. 013001, 2018.

[40] J. E. Saal, A. O. Oliynyk, and B. Meredig, "Machine learning in materials discovery: Confirmed predictions and their underlying approaches," *Annual Review of Materials Research*, vol. 50, no. 1, p. null, 2020. [Online]. Available: https://doi.org/10.1146/annurev-matsci-090319-010954

[41] The Minerals Metals & Materials Society (TMS), *Building a Materials Data Infrastructure: Opening New Pathways to Discovery and Innovation in Science and Engineering*. Pittsburgh, PA: TMS, 2017. [Online]. Available: dx.doi.org/10.7449/mdistudy_1

[42] L. Himanen, A. Geurts, A. S. Foster, and P. Rinke, "Data-driven materials science: Status, challenges, and perspectives," *Advanced Science*, vol. 6, no. 21, p. 1900808, 2019.

[43] "8 - floating zone growth of oxides and metallic alloys," in *Handbook*

of Crystal Growth (Second Edition), second edition ed., ser. Handbook of Crystal Growth, P. Rudolph, Ed. Boston: Elsevier, 2015, pp. 281 – 329. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780444633033000080

[44] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[45] K. He, G. Gkioxari, P. Dollr, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[46] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring R-CNN," *CoRR*, vol. abs/1903.00241, 2019. [Online]. Available: http://arxiv.org/abs/1903.00241

[47] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Interna-*

tional Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

[49] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollr, "Microsoft coco: Common objects in context," 2014, cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. [Online]. Available: http://arxiv.org/abs/1405.0312

[50] D. Elbert, N. Carey, and T. McQueen, "The PARADIM streaming data platform," in preparation.

[51] N. Carey, W. Phelan, A. Rachidi, C. Krill, P. Appel, J. Zahn, M. Hudes, B. Schuster, T. McQueen, and D. Elbert, "Data sets: Deep neural net instance segmentation for optimization of optical floating zone melt geometry," 2020. [Online]. Available: https://doi.org/10.34863/41cn-4361

[52] M. Andriluka, J. R. R. Uijlings, and V. Ferrari, "Fluid annotation: a human-machine collaboration interface for full image annotation," *CoRR*, vol. abs/1806.07527, 2018. [Online]. Available: http://arxiv.org/abs/1806.07527

[53] M. Straker, A. Chauhan, M. Sinha, W. A. Phelan, M. Chandrashekhar,

K. J. Hemker, C. Marvel, and M. Spencer, "Growth of high purity zone-refined boron carbide single crystals by laser diode floating zone method," *Journal of Crystal Growth*, vol. 543, p. 125700, Aug. 2020. [Online]. Available: https://doi.org/10.1016/j.jcrysgro.2020.125700

[54] T. McQueen, "Message layer encryption for kafka," https://github.com/tmcqueen-materials/kafkacrypto, 2019.

[55] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008. [Online]. Available: http://dx.doi.org/10.1007/s11263-007-0090-8

[56] K. Wada, "labelme: Image Polygonal Annotation with Python," https://github.com/wkentaro/labelme, 2016.

[57] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[58] K. He, R. B. Girshick, and P. Dollár, "Rethinking imagenet pre-training," *CoRR*, vol. abs/1811.08883, 2018. [Online]. Available: http://arxiv.org/abs/1811.08883

[59] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of*

Statistical Learning: Data Mining, Inference, and Prediction, ser. Springer series in statistics. Springer, 2009. [Online]. Available: https://books.google.com/books?id=eBSgoAEACAAJ

[60] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[61] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. USA: McGraw-Hill, Inc., 1986.

[62] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[63] J. J. Vidal, "Toward direct brain-computer communication," *Annual Review of Biophysics and Bioengineering*, vol. 2, no. 1, pp. 157–180, 1973, pMID: 4583653. [Online]. Available: https://doi.org/10.1146/annurev.bb.02.060173.001105

[64] S. Lees, N. Dayan, H. Cecotti, P. McCullagh, L. Maguire, F. Lotte, and D. Coyle, "A review of rapid serial visual presentation-based brain–computer interfaces," *Journal of Neural Engineering*,

vol. 15, no. 2, p. 021001, jan 2018. [Online]. Available: https://doi.org/10.1088%2F1741-2552%2Faa9817

[65] C. Kothe. (2014) Lab streaming layer. [Online]. Available: https://github.com/sccn/labstreaminglayer

[66] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "Eegnet: A compact convolutional network for eeg-based brain-computer interfaces," *CoRR*, vol. abs/1611.08024, 2016. [Online]. Available: http://arxiv.org/abs/1611.08024

Vita

Nick Carey is a Computer Science Ph.D. candidate advised by Tamas Budavari. Since earning his B.S. in C.S. from University of Maryland, he's been a part of JHU's Institute for Data Intensive Engineering and Science. His research lies in Brain-Computer Interfaces, Data Navigation, Neural Network classifiers, and applying the above to the Material Sciences in HEMI/MEDE.

6.1 Publications

S. Nutanong, N. Carey, Y. Ahmad, A. Szalay, T. Woolf, "Adaptive exploration for large-scale protein analysis in the molecular dynamics database," *In Proceedings of the 25th International Conference on Scientific and Statistical Database Management (SSDBM)*. Association for Computing Machinery, New York, 2013, Article 45. doi: 10.1145/2484838.2484872.

N. Carey, T. Budavri, Y. Ahmad and A. Szalay, "DOT-K: A distributed on-

line top-K elements algorithm using extreme value statistics," *2016 IEEE 12th International Conference on e-Science* (e-Science), Baltimore, MD, 2016, pp. 130-136. doi: 10.1109/eScience.2016.7870893.

N. Carey, T. Budavari, N. Daphalapurkar, K.T. Ramesh, "Data integration for materials research," *Integrating Materials and Manufacturing Innovation*, 2016, Volume 5. doi: 10.1186/s40192-016-0049-0.

N. Carey, W. Phelan, A. Rachidi, C. Krill, P. Appel, J. Zahn, M. Hudes, B. Schuster, T. McQueen, D. Elbert, "Optimization of Optical Floating Zone Melt Geometry via Deep Neural Net Instance Segmentation," In Preparation, 2020. 10.34863/41cn-4361