UNIVERSITY OF CALIFORNIA SAN DIEGO

DynOMICS: a total microfluidic-AI system for genome-wide *E. coli* transcriptional dynamics and heavy metal biosensing

A dissertation submitted in partial satisfaction of the

requirements for the degree

Doctor of Philosophy

in

Bioengineering

by

Garrett Cook Graham

Committee in charge:

Professor Jeff Hasty, Chair
Professor Todd Coleman
Professor Prashant Mali
Professor Justin Meyer
Professor Joseph Pogliano

2019

The dissertation of Garrett Cook Graham is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California San Diego

2019

DEDICATION

To my parents, my brothers, and sister for all of their love. To Nicholas Csicsery, Elizabeth Stasiowski, and Gregoire Thouvenin for being the best project partners I could ever have. And to Dustin and Jenn Ragusa for their friendship and overwhelming generosity, especially as expressed via their kitchen.

EPIGRAPH

THE WINDHOVER

To Christ our Lord

I caught this morning morning's minion, king-
dom of daylight's dauphin, dapple-dawn-drawn Falcon, in his riding
Of the rolling level underneath him steady air, and striding
High there, how he rung upon the rein of a wimpling wing
In his ecstasy! then off, off forth on swing,

As a skate's heel sweeps smooth on a bow-bend: the hurl and gliding
Rebuffed the big wind. My heart in hiding
Stirred for a bird, – the achieve of, the mastery of the thing!

Brute beauty and valour and act, oh, air, pride, plume, here
Buckle! AND the fire that breaks from thee then, a billion
Times told lovelier, more dangerous, O my chevalier!

No wonder of it: shéer plód makes plough down sillion
Shine, and blue-bleak embers, ah my dear,
Fall, gall themselves, and gash gold-vermilion.

Gerard Manley Hopkins (1844-1889)

TABLE OF CONTENTS

Signature Page iii

Dedication iv

Epigraph v

Acknowledgements xiv

Vita xv

Abstract of the Dissertation xvi

Chapter 1 Introduction 1

1.1 Dynamics in genetic networks 1

1.2 Toxic heavy metals and their threat to American water supplies 2

1.3 Genome-scale microfluidic biosensors 5

1.4 Promise of machine learning in the quantitative life sciences 7

Chapter 2 Construction of a specific microfluidic biosensor for the detection of heavy metal contamination in drinking water 10

2.1 Development of a multi-strain biosensor microfluidic device and an accompanying loading protocol 10

2.1.1 Development of a commercially-viable lyophilization protocol for long-term storage of biosensor microfluidic devices 14

2.1.2 Multiclass, multilabel machine learning-classification of biosensor data . . 22

Chapter 3 Explainable multiclass machine learning on genomic time series with applications to the detection of heavy metal contamination 33

3.1 Introduction 33

3.2 Methodology 36

3.2.1 Experiments of interest 36

3.2.2 Data preprocessing and feature engineering 38

3.2.3 Additional data preprocessing 45

3.2.4 Implementation of Lubansky data differentiation 46

3.2.5 Initial supervised learning with random forests 47

3.2.6 Statistical distance distribution searches for the identification of arsenic- and mercury-sensitive promoters 49

3.2.7 Leave-N-out cross-validation 51

3.2.8 Classifier evaluation metrics for imbalanced datasets 55

3.2.9 Supervised learning with extreme gradient boosted trees (XGBoost) 59

3.2.10 Bayesian optimization of classifier hyperparameters 61

3.2.11 Supervised deep learning with long short-term memory recursive neural networks 65

3.2.12 Explainable artificial intelligence (XAI) with Shapley additive explanation values 70

3.3 Results and discussion 74

3.3.1 Flat-field correction 74

3.3.2 Data preprocessing and feature engineering 75

3.3.3 Implementation of Lubansky data differentiation 77

3.3.4 Initial supervised learning with tree ensemble learners 78

3.3.5 Statistical distance distribution searches for the identification of lead-, arsenic- and mercury-sensitive promoters 81

3.3.6 Supervised learning with extreme gradient boosted trees (XGBoost) 88

3.3.7 Supervised deep learning with long short-term memory recursive neural networks 90

3.3.8 Explainable artificial intelligence (XAI) with Shapley additive explanation values 95

3.4 Conclusion 105

3.5 Acknowledgements 106

Bibliography 107

LIST OF FIGURES

Figure 1.1 The Gold King Mine Disaster. 4
Figure 1.2 Sampling the toxic plume from the San Juan River. 5
Figure 1.3 The first neural network. 8
Figure 1.4 Machine learning and the black box problem. 9

Figure 2.1 The evolution of biosensor microfluidic design and implementation. 11
Figure 2.2 Finite element analysis (FEA) modeling of biosensor microfluidic designs. . 12
Figure 2.3 A lyophilizable, multistrain biosensor microfluidic device. 15
Figure 2.4 Multistrain, in-chip freeze-drying. 16
Figure 2.5 A 16-strain biosensor chip with cryoprotected, independently-loaded, lyophilized, and revived strains. 17
Figure 2.6 Batch cryoprotectant testing in Escherichia coli. 18
Figure 2.7 Intra-chip revival of lyophilized *Escherichia coli* biosensor strains. 19
Figure 2.8 Response of an engineered biosensor strain to single-toxin induction. 21
Figure 2.9 Response of an engineered biosensor strain to multi-toxin induction. 21
Figure 2.10 A graphical analogy for machine learning classification. 23
Figure 2.11 Five distinct features used in biosensor classifier construction. 24
Figure 2.12 Multiclass, multilabel classifier performance visualization on biosensor data. 29

Figure 3.1 An overview of the DynOMICS system. 34
Figure 3.2 The data extraction regions of a standard DynOMICS cell trap. 37
Figure 3.3 Response variation by cell trap region. 39

Figure 3.4 Optical vignetting in both transmitted and fluorescent channels of the DynOMICS optical system. 40
Figure 3.5 The DynOMICS automated flat-field correction process. 41
Figure 3.6 A visualization of the decicion boundaries of random forest classifiers classifying on specific sensor data. 48
Figure 3.7 The response and baseline derivative distributions of a cadmium-sensitive strain. 50
Figure 3.8 Cumulative distribution functions (CDFs) of induced and uninduced strain behaviors. 51
Figure 3.9 A depiction of four-fold cross-validation. 53
Figure 3.10 A depiction of leave-one-out cross-validation. 54
Figure 3.11 The imbalanced class composition of our data set. 56
Figure 3.12 An abstracted visualization of precision and recall as classifier evaluation metrics. 57
Figure 3.13 A conceptual depiction of an XGBoost learner. 60
Figure 3.14 Grid search versus random search strategies for classifier hyperparameter optimization. 63
Figure 3.15 The learning process of a Bayesian optimizer. 64
Figure 3.16 A generic recurrent neural network (RNN). 65
Figure 3.17 A long short-term memory recurrent neural network (LSTM-RNN) node. . 67
Figure 3.18 LSTM-RNN feature engineering and explainable AI (XAI) workflow. . . . 69
Figure 3.19 A conceptual depiction of the AI complexity-interpretability trade-off. . . . 71
Figure 3.20 Heatmaps of both raw and flat field corrected TL and FL backgrounds by device position. 76
Figure 3.21 A comparison of the raw and processed fluorescent signals. 77
Figure 3.22 First derivatives of Lubansky differentiated experimental data. 78
Figure 3.23 Resulting recall scores of optimized random forest classifiers. 79

Figure 3.24 The strain *ykgJ* responding to Pb(II). 80
Figure 3.25 The exploratory tree ensemble classifiers failed to detected lead in a binary unoptimized classification setting without feature selection. 82
Figure 3.26 The control strains' water- versus cadmium-induced first-derivative feature distributions. 83
Figure 3.27 A comparison of induced-uninduced energy distances between a metal-sensitive strain and the on-chip control strains. 85
Figure 3.28 Statistical distance distributions of an arsenic-insensitive strain. 86
Figure 3.29 Mercury response trajectories. 87
Figure 3.30 A summary of the performance of an XGBoost classifier using handpicked strains. 89
Figure 3.31 The classification results for an LSTM-RNN. 91
Figure 3.32 Deep learning network predictions on standardized experiment data as a function of time. 92
Figure 3.33 Deep learning network predictions on urban water samples as a function of time. 94
Figure 3.34 Deep learning network predictions on San Juan River samples containing contamination from the Gold King Mine Spill as a function of time. 96
Figure 3.35 A SHAP XAI's summary of the XGBoost feature importances by device position. 97
Figure 3.36 A SHAP XAI's summary of the XGBoost feature importances by strain. . . 99
Figure 3.37 A SHAP XAI's summary of the LSTM-RNN feature importances by strain. 100
Figure 3.38 A SHAP XAI's summary of the XGBoost feature importances for cadmium detection. 102
Figure 3.39 An LSTM-RNN SHAP XAI's attribution plots for cadmium predictions. . . 103

LIST OF TABLES

Table 2.1 F_1 scores for both random forest (RF) and support vector machine (SVM) classifiers trained on the full feature set and individual features, including and excluding the existence of a "water" class. 30

Table 3.1 A statistical summary of the effects of flat-field corrections. 75

ACKNOWLEDGEMENTS

I could not have completed this graduate program without Jeff, Lev, Ramon, Todd, Mike F., Scott C., Spencer S., Team DynOMICS, Omar, Arthur P., Thiago, Phillip, and Nolan. In addition, my community of friends, both here in San Diego and scattered about the world, especially the Ragusas, Casey O., Aaron P., Mike H., Adam T., Andrew T., Leo G., and Hoff.

In addition, Chapter 3, in part, is currently being prepared for submission of the material by with the following individuals: Graham, Garrett; Csicsery, Nicholas; Stasiowski, Elizabeth; Thouvenin, Gregoire; Hasty, Jeff. The dissertation author was the primary researcher and author of this material.

VITA

2010	B.S. in Mathematics and Physics *cum laude*, University of Richmond
2015	Masters of Science in Bioengineering, University of California, San Diego
2019	Doctor of Philosophy in Bioengineering, University of California, San Diego

PUBLICATIONS

Bleak, C., Bowman, H., Lynch, A. G., Graham, G., Hughes, J., Matucci, F., & Sapir, E. (2013). Centralizers in the R. thompson group Vn. Groups, Geometry, and Dynamics, 7(4), 821865. https://doi.org/10.4171/GGD/207

Cates, J., Graham, G. C., Omattage, N., Pavesich, E., Setliff, I., Shaw, J., Lipan, O. (2011). Sensing the Heat Stress by Mammalian Cells. BMC Biophysics, 4(1). https://doi.org/10.1186/2046-1682-4-16

Graham, G. C., Lipan, O., Graham, G. C., Lipan, O. (2011). The effect of coupled stochastic processes in a two-state biochemical switch. J Biol Phys, 37, 441462. https://doi.org/10.1007/s10867-011-9226-8

ABSTRACT OF THE DISSERTATION

DynOMICS: a total microfluidic-AI system for genome-wide *E. coli* transcriptional dynamics and heavy metal biosensing

by

Garrett Cook Graham

Doctor of Philosophy in Bioengineering

University of California San Diego, 2019

Professor Jeff Hasty, Chair

Recent developments in the field of quantitative biology have demonstrated that genetic networks rely upon information encoded in their temporal dynamics, rather than beginning and ending steady-states, to govern their behavior. However, until now, there has been no tool with which to continuously observe genome-wide transcriptional dynamics without terminating the subject population. In response to this need, we developed DynOMICS, a total microfluidic and machine learning system that can monitor the state of gene expression across the *E. coli* genome in real time.

We demonstrate its effectiveness as a field-deployable sensor, showing that it can learn the dynamic genomic signatures of heavy metal stress in both actual urban waters from several American cities and in samples from a toxic mining spill. By harnessing the microfluidics to a state-of-the-art deep neural network and an associated explanatory artificial intelligence (XAI) algorithm, we demonstrate its potential as a scientific instrument. We show that, in combination with DynOMICS, we can use deep learning networks to learn and understand bacterial transcriptional dynamics on a genome-scale. The combination of advanced microfluidics and AI-XAI is the first of its kind and is a powerful tool for quantitatively interrogating the E. coli genome.

Chapter 1

Introduction

1.1 Dynamics in genetic networks

The advent of synthetic and systems biology have revealed the extent to which life encodes and responds to information encoded in the dynamics of genetic networks and their stimuli, rather than relying solely upon steady-state signals [45]. Since synthetic biology's inception in 2000 with the publication of Michael Elowitz's represillator and Jim Collin's toggle switch, the field has demonstrated time and again that dynamic behavior is essential to understanding living systems [35, 31, 87, 50]. Systems biology, in contrast with synthetic biology, has the scale to capture global cellular parameters, which endows it with powerful predictive capabilities [62]. However, few omics-based modeling approaches are built on dynamics, usually due to a lack of data and a dearth of frameworks with which to build dynamic models [14] . Indeed, it has been shown that, even in systems where the contributions of dynamic behavior are negligible, the ability to study such systems dynamically greatly increases the information content of the experimental results [70].

Despite of the importance of capturing time series data, the current measurement technologies for the study of biological dynamics suffer from a variety of drawbacks [96]. RNAseq, qPCR, and flow cytometers only provide snapshots of fixed cells, requiring many separate measurements, usually from many separate experiments, to create a meaningful understanding of anything more than a

handful of genes. Fluorimetric plate reader and batch culture experiments, while able to produce time series data, are limited in the number of genes that can be simultaneously studied. In addition, these techniques usually rely on non-chemostatic environments, which renders them limited in their experimental capabilities. Finally, microfluidic technologies, especially combined with microscopy, offer a method to measure cells at high resolution within a chemostatic environment; however, these approaches are notoriously low-throughput, limited to very few genes-of-interest.

Thus, further advances in quantitatively understanding cellular function requires new experimental technologies that are not only high-throughput, but are capable of generating meaningfully long time series-data at appropriate resolutions [89, 45].

1.2 Toxic heavy metals and their threat to American water supplies

Concurrent with the rise of quantitative biology, heavy metal contamination has grown to be one of the United States' preeminent public health threats over the last decade. While more complicated and specific definitions exist, heavy metals are commonly defined as a large class of elemental metals that have a high atomic weight and are at least five times the density of water [107]. Heavy metals are found naturally in the earth's crust, usually in relatively low concentrations, but are concentrated to toxic levels via human activity. Mining, various industrial activities, and aging water supply infrastructures are the most common sources by which humans are exposed to these metals [90]. As a result of these pollutant sources, the World Health Organization (WHO) has identified toxic heavy metals as a significant international public health threat; no less than four heavy metals (arsenic, cadmium, lead, and mercury) made the WHO's list of the top ten chemicals most threatening to public health [118].

While heavy metal contamination may seem like a remote and distant threat to many Americans, and while it is true that the most frequent exposure to toxic heavy metals occurs in low-to-middle income countries, it constitutes a growing threat to the United States drinking and agricultural water

supplies [58]. Over the past five years alone, the contamination of US water supplies by heavy metals has lead to dozens of public health and environmental emergencies, profoundly affecting the health and livelihoods of hundreds of thousands of Americans [92]. In the most commonly cited case, that of Flint, Michigan, an estimated 140,000 people were exposed to dangerously high levels of lead and other metals, via their drinking water for over a year [92]. Even today, the city is still grappling with the health and societal consequences of the disaster [102]. Heavy metal contamination is not limited just to Flint: a 2016 investigation by Reuters found that there were close to 3,000 additional regions of the US with lead levels in their drinking water that were at least twice as high as Flint's [86]. This threat is likely to only grow more severe in the coming decades, as more and more heavy metals leach out of corroded pipes in aging water supply infrastructures [113].

Heavy metals are not just a threat to America's drinking water supply. Agricultural water sources and natural waterways are also threatened by the accumulation of decades and, in some areas, centuries of mining and agricultural activity. In 2019, the Associated Press used publicly-available records to determine that, at average daily flows, over 50 M gallons of toxic mine tailings still flow into and contaminate American waterways every day [23]. In worst-case scenarios, large-scale spills have occurred when dams and other barriers holding back reservoirs of mine tailings, coal ash, and contaminated waste water fail, injecting huge concentrations of toxic heavy metals into local rivers [55].

One recent such spill was the Gold King Mine Disaster, which occurred outside of Silverton, CO on August 5, 2015 [28]. The Gold King Mine, which was abandoned in 1923, had accumulated a large amount of contaminated groundwater inside of it, a consequence of acid mine drainage [34]. Acid mine drainage occurs when large-amounts of bedrock are exposed by the actions of mining. When metal sulfides in the rock are exposed to a combination of water and oxygen, the water can acidify, which in turn leaches other metal ions out of the surrounding rock [10]. These metals often include heavy metals, such as cadmium, arsenic, copper, zinc, iron, chromium, and lead [21]. The 2015 spill occurred when environmental cleanup crew, composed of Environmental Protection Agency (EPA) personnel and contractors, accidentally caused a dam plugging the mine's

Figure 1.1: The Gold King Mine Disaster. An EPA cleanup crew, tasked with decontaminating the mine after an accumulation of decades of acid mine runoff, accidentally caused the retaining dam to burst. Millions of gallons of water rushed into the Animas and then the San Juan River, turning both of them yellow and toxic [109, 110].

entrance to fail (Fig. 1.1). All of the mine's waste water poured into Cement Creek, and then into the Animas River, followed by the San Juan River, turning them bright yellow and contaminating their waters (Fig. 1.2).

While the long-term impacts of the disaster throughout all effected waterways are still being studied, the short-term impacts were devastating to regional peoples. The thousands of members of the Navajo Nation, through which the San Juan flows, had to shut off irrigation from the river to their crops and livestock in the days after the spill [3]. In the +40 C August temperatures, the damage to crops and livestock health was widespread and significant. While some farms received water that was delivered by the US EPA, many remote farms did not. As of the writing of this document, there are still ongoing lawsuits between the Navajo and the EPA over compensation [5].

Heavy metal toxins have long posed a threat to international and American water supplies and public health. Ironically, even as our nation's environmental protection and conservation laws have improved and will continue to do so, heavy metal contamination is guaranteed to occur more and more frequently in the coming years, as water supply infrastructure and abandoned mining and industrial sites age.

Figure 1.2: Sampling the toxic plume from the San Juan River. The plume made its way downstream to Mexican Hat, Utah approximately four days after the spill. The author drove overnight from San Diego to Mexican Hat in order to sample the plume for the purposes of this project.

1.3 Genome-scale microfluidic biosensors

Much of the agricultural damage incurred by the Gold King Mine Disaster could have been prevented by better metal sensing technologies. At the time of the spill, the only way to adequately determine the level of contamination in the river water was to send personnel from the states' environmental protection departments to the contaminated waterways and manually collect samples in specially pre-treated scientific containers. From there, the field specialists were required to transport the samples, on ice, to facilities with inductively coupled plasma mass spectrometers (ICP-MS) and other analytical chemistry equipment [98]. This process resulted in two-to-four day gaps between sampling and results. Since during that time it was unknown whether the water was safe for agricultural use, the farmers who depended on the San Juan River were forced to cut off their irrigation systems and pumps. The result was that these farmers had to watch as their crops withered and their livestock dehydrated and sickened, not knowing whether the water was still safe for use [3].

Much of this damage could have been avoided by continuous, on-site heavy metal detection technology. In fact, the scope of most of the heavy metal-related public health crises around the US in recent years would have been drastically reduced by an increase in sampling frequencies and

detection capabilities. Since real-time detection via automatable and remote chemical sensors is costly, the aforementioned analytical chemistry technology needed for detection remains confined to labs, often hundreds or even thousands of miles away from the sites where they are needed. The problem of heavy metal detection begs for a cheaper and faster solution.

Biosensors have long been hypothesized as a cheaper and more rapid sensing alternative to analytical chemistry methods for detecting contamination in water supplies, especially contamination by heavy metals [100]. While it is rare to find heavy-metal sensing genetic systems in multicellular organisms, many kinds of microbial life evolved genetic defenses to detect, capture, and export heavy metals from within their own cells [116, 51, 15]. Using these systems, multiple specific sensing strains can be engineered to detect a suite of toxins. In addition, once engineered, microbes have the advantage of being inexpensive to produce. Contamination can be detected in minutes-to-hours, rather than days [73].

Microfluidic devices offer one potential path towards building field-deployable, in-line heavy metal detection systems. They have numerous advantages over other biosensing device paradigms, notably a massive reduction in both size and cost. However, constructing a field-deployable biosensor capable of detecting multiple toxins would require either a single multitoxin-detecting strain or the isolated loading of unique sensing strains. In addition, it would require ensuring a level of robustness that most lab-based microfluidic platforms utterly lack. Until now, all microbial biosensors have either been single-toxin, single-strain, constrained to the lab, and/or batch culture-based [60, 26, 61].

Genome-scale microfluidic instruments, devices capable of tracking the dynamics of thousands of genes at a time, offer one way of building a versatile biosensor, as well as a way to study the dynamics of gene networks. These devices could leverage biology's natural ability to sense and respond to environmental stimuli. As non-specific sensors, they could potentially sense any substance or stimulus that perturbs the observed portion of an organism's genome, as long as the response pattern is learned from training data. Over the past five years, much progress has been made in constructing and using genome-scale devices in the lab. Notable devices include Sebastien Maerkl's yeast chemostat array and Savas Tay's mammalian neuronal stem cell chip [32, 121].

However, no one has come forth with a satisfactory bacterially-based device, wherein isogenic colonies are allowed to grow in a chemostat environment for days or weeks [115]. The possibilities that such a device would present, not only as a solution to the heavy metal-detection problem, but also to the gap in our abilities to study genome-scale dynamics, are tantalizing.

1.4 Promise of machine learning in the quantitative life sciences

Since its inception, the field of artificial intelligence has been inextricably linked with the quantitative life sciences. Artificial neural networks and their progeny, deep neural networks, were first described by neurophysiologist Warren McCullough and mathematician Walter Pitts in their seminal 1943 paper "A Logical Calculus of Ideas Immanent in Nervous Activity". The paper, which was published in the *Bulletin of Mathematical Biophysics*, attempted to demonstrate how networks of biological neurons operate by replicating them with electrical circuits that were capable of executing logic functions [81] (Fig. 1.3). Indeed, computer science and quantitative biology's kinship extends much further, with biological systems fascinating some of the most notable pioneering computer scientists (see Alan Turing's 1952 paper "The Chemical Basis of Morphogenesis" for the first quantitative attempt to explain pattern formation in biological systems).

Over the last two decades, computer science has experienced a massive increase in computing and algorithmic power, especially in the field of artificial intelligence (AI) [108]. Concurrently, the biological sciences have seen an increase in their abilities to gather magnitudes more experimental data in less time than ever before; the rise of -omics technologies embody this fact [105, 112, 44]. As data sets grow ever more complicated, the ability for machine learning algorithms to model arbitrarily complex functions has given researchers the power to discover patterns hidden deep within these data [24]. Most recently, in the past five years, deep neural networks have demonstrated astounding capabilities that were, until now, solely the realm of humans [69]. Notable examples include Google Deepmind's AlphaGo and AlphaStar projects [13].

Machine learning, however, has not yet been able to deliver fully on its potential to facilitate

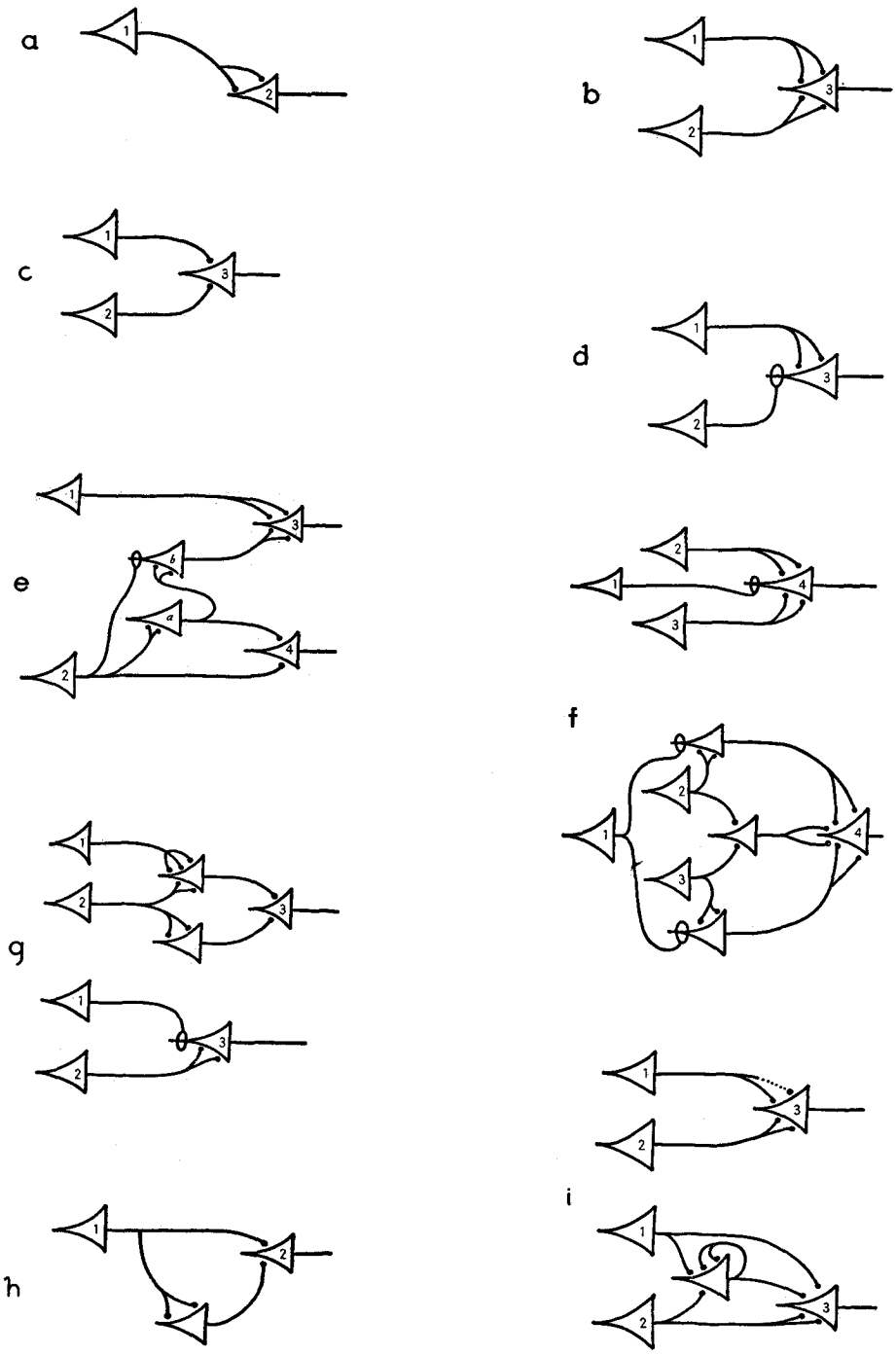


FIGURE 1

Figure 1.3: The first neural network. This figure is the original from McCullough and Pitts' 1943 paper that introduced the concept of neural networks and laid the foundation for the field of artificial intelligence. McCullough, as a neurophysiologist, endowed his artificial network some of the same shapes as actual biological neurons.

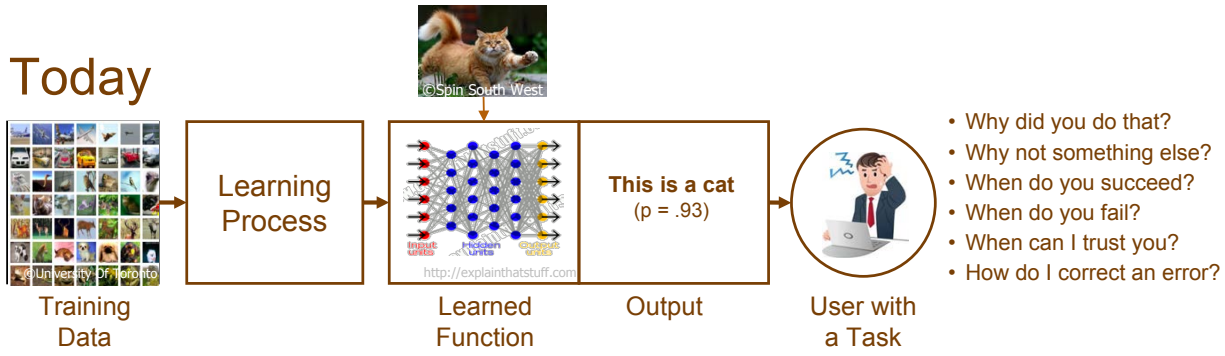


Figure 1.4: Machine learning and the black box problem. A massive challenge confronting the field of AI today is to understand how and why complex, high-performance AI algorithms make the decisions that they do. Until these models are interpretable, their use will be limited. Originally from [47]

scientific discovery. The primary reason for this limitation is because of what is known as the ”black box problem”. The black box problem refers to the fact that, in general, the more powerful an algorithm’s ability to model complex phenomenon, the more obscure and nonintuitive the algorithm’s inner-workings are to human operators 1.4. While artificial intelligence has shown itself to be extremely proficient at discovering the ”what” of scientific phenomena, we humans are still left struggling to understand its ”why”.

Hence, the rise of deep neural networks, AI’s ultimate black-box algorithms, has motivated a push to develop explainable artificial intelligence (XAI) techniques [4]. The quantitative life sciences have recently seen some excellent efforts to create XAI in service of research, but these techniques are either model-specific or limited in their applicability because of non-uniqueness or computational complexities [78, 119, 122].

Thus, the combination of a computationally-efficient, model-agnostic XAI method with a high-throughput, dynamic -omics experimental technology could represent a significant leap in our ability to quantitatively understand genomes. As such, we find our project in a unique position where, by judiciously combining high-throughput microfluidics with advanced data analysis, machine learning, and explainable artificial intelligence, we could not only create a new avenue for interrogating genomic dynamics, but could also build a field-deployable heavy metal toxin detector that is orders of magnitude faster than any currently available method.

Chapter 2

Construction of a specific microfluidic biosensor for the detection of heavy metal contamination in drinking water

2.1 Development of a multi-strain biosensor microfluidic device and an accompanying loading protocol

In order to realize a commercially viable, field-implementable microfluidic biosensor, it was first necessary to develop techniques to load multiple engineered sensor strains and lyophilize them *in situ* for facilitating long-term storage and transport. The original biosensor device design, as depicted in Figure 2.1, was not suitable to either advanced multistrain loading or lyophilization. In order to achieve these goals, a radical redesign of the basic biosensor device was necessary.

In order to achieve long-term storage capability via freeze-drying and multistrain loading, it was necessary to introduce reservoirs into the strain arrays. The reservoirs would act to increase the number of individual cells associated with each array and thereby increase the possibility that, after freeze-drying, at least one viable individual remains in the array upon rehydration. Additionally, it

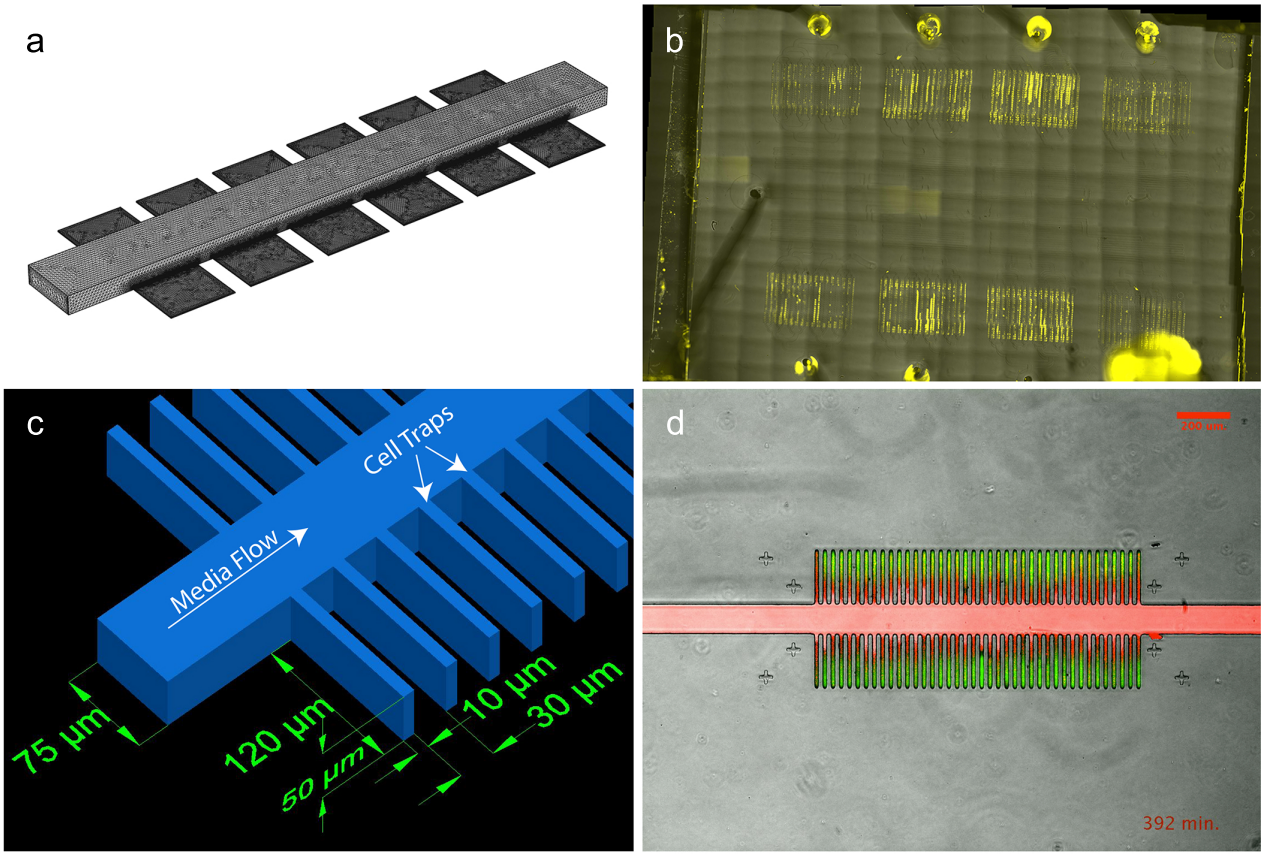


Figure 2.1: The evolution of biosensor microfluidic design and implementation. (a) A finite element analysis (FEA) mesh, built from a computer animated design (CAD) of a biosensor microfluidic device. This design, which is a powerful experimental tool, proved impractical for practical biosensing purposes due to its weakly visible FL requiring a research-grade inverted fluorescent microscope. (b) A multi-layer biosensor prototype microfluidic chip with strain FL visible as a consequence of induction with 1 uM sodium arsenite. (c) A three-dimensional model of the biosensor gill trap design. This design increases the colony density with respect to the device's z-axis, which effectively increases the FL intensity of an array of traps on the device. This intensity increase enables optical detection with lower-quality, less-expensive optics. (d) An array of arsenic-sensing E. coli fluorescing in the presence of 1 uM sodium arsenite. The green indicates strain FL and the red indicates the presence of sodium arsenite in the media.

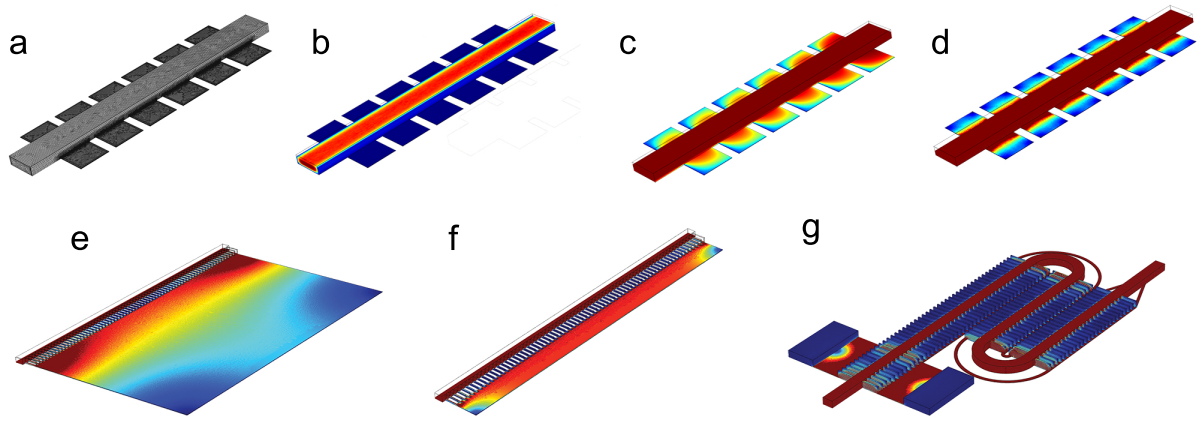


Figure 2.2: Finite element analysis (FEA) modeling of biosensor microfluidic designs. (a) A three-dimensional mesh of a standard main flow channel with side traps. (b) The velocity profile for laminar water flow in the main channel of the same design. (c) The flow velocity profile in the side traps of the same design. Note that the scale has been reduced to highlight the differences in low-velocity flow regimes within unobstructed traps. (d) The concentration profile for a rate-limiting nutrient (here assumed to be an essential amino acid for E. coli) based on a reaction-diffusion model within the same device, roughly simulating the consumption of such a nutrient by a bacterial colony within the trap. This kind of FEA model allows an experimentalist to approximately determine whether a design is worth building and testing by filtering out designs with obvious flaws. (e) A gill trap-based biosensor device design featuring a large reservoir behind the traps for freeze dry-revival. (f) A gill trap-based biosensor device design featuring a much smaller reservoir behind the traps for freeze dry-revival. (f) The final evolution of a multistrain, lyophilizable biosensor chip design that was subsequently successfully built and tested. This design featured the novel bottle-necked reservoirs and feeder channels, which have served as the foundations for all subsequent biosensor designs.

was thought that these reservoirs could act as a loading region, either fluidically via port-loading or mechanically via hand- or robot-guided strain spotting.

Initially, the existing biosensor gill chip design was taken and modified by placing a large rectangular reservoirs behind the existing trap areas, in order to act as a deposition area, as depicted in Fig. 2.2e and 2.2f. The gill traps' dimensions were left unchanged, while the reservoirs varied in depth from 200-1600 μm. These devices were built and tested experimentally with arsenic-sensing strain pLB-As3 for effects on growth rate and toxin-sensing ability. Due to the tendency of laminar flow to spread flow velocities evenly in a manner proportional to the cross-sectional area through which the fluid is flowing, larger reservoirs performed poorly with regards to populating gill traps, while smaller reservoirs performed better.

Since the process of designing, experimentally testing, and redesigning microfluidic devices

is extremely time-consuming, a computer-based design and modeling pipeline was created in order to efficiently design and rapidly vet more efficacious devices. To verify that such a pipeline could indeed work, three-dimensional models of the aforementioned reservoir devices were first analyzed using finite-element analysis (FEA) software (COMSOL Multiphysics v4.3a). When the models correctly predicted that shallower reservoir depth would allow for better trap-seeding via lower laminar through-flow, three-dimensional models of candidate reservoir devices were then built and analyzed via COMSOL, as seen in Fig. 2.2. By narrowing the reservoir width from 1600 μm to 200 μm, I found that the flow velocities at the reservoir corners near the gill trapping regions were greatly reduced (see analogous regions in Fig. 2.2e and f). I concluded that minimizing the width of the cell deposition reservoir would minimize flushing of colonies growing in nearby trapping regions. I subsequently modified the gill chip to allow seeding of the cell traps by narrow upstream strain reservoirs (see lower left of Fig. 2.2g).

Prior to the purchase and incorporation of the spotting robot discussed in Chapter 3, which would allow for high throughput, single-step loading of strains via physical deposition of bacterial microcolonies (”spotting”), a fluidic-based multistrain loading protocol was designed and tested. The novel strain reservoirs can be filled with cryopreserved cells via injection through the loading ports/channels that feed into the corners of the reservoirs. After *in situ* lyophilization of the chips, the loading ports are then sealed with a quick-curing PDMS variant (Dow Corning Sylgard 170). Upon device use, the cells revive by media flow from the main channel into the narrow cell reservoirs. Cells washed from the reservoirs seed the gill traps by flowing from small “feeder” channels connected to the reservoirs at the rear of the traps. These feeder channels, another novel microfluidic design, allow the backs of the traps to be seeded with revived cells from the upstream reservoir while maintaining low laminar flow through the trapping channels. This low laminar flow prevents cell washout from both the feeder channels themselves and from the attached gill traps. Reservoir and feeder channel dimensions can be modified to account for various cell-spotting techniques.

In order to successfully culture multiple independent biosensor strains, a 16-strain and an 18-strain biosensor chip were designed, built, and successfully tested for multistrain loading

and freeze-drying (see Fig. 2.3). The chips' dimensions conformed both to the constraints of the biosensor's optical detection systems and to the constraints of its strain requirements. The previously described multistrain port-loading technique was successfully tested with the device (see Fig. 2.4 and Fig. 2.5). An additional air-drying and chemical bonding-based method was developed and tested before being abandoned in favor of robot-based spotting.

2.1.1 Development of a commercially-viable lyophilization protocol for long-term storage of biosensor microfluidic devices

In parallel to the developments described in the preceding subsection, a lyophilization technique was designed and tested to enable easy, non-refrigerated long-range transport of biosensor chips. Such a capability would enable field-implementation of the biosensor, as opposed to if the device required laboratory-based strain-loading, culture, and climate-controlled transport to the testing site. In order to ensure maximum chance of revival after lyophilization, a range of cryoprotectants suitable for engineered biosensor strains and for microfluidic geometries was formulated from a combination of literature-based protocols, current industrial practices, and experimentation.

Via literature searches, candidate cryoprotectants known to reliably preserve engineered *E. coli* strains during and after freeze-drying and during rehydration were identified.

The cryoprotectant-growth medias include:

1. 2.5% Luria-Bertrani Broth (LB) (w/v) + spectinomycin
2. 2.5% LB + 0.4 %glucose (w/v) + spectinomycin;
3. 2.5% LB + 0.4 %sucrose + spectinomycin;
4. 2.5% LB + 0.4 %trehalose + spectinomycin;
5. M9 + 0.4 %glucose + spectinomycin;

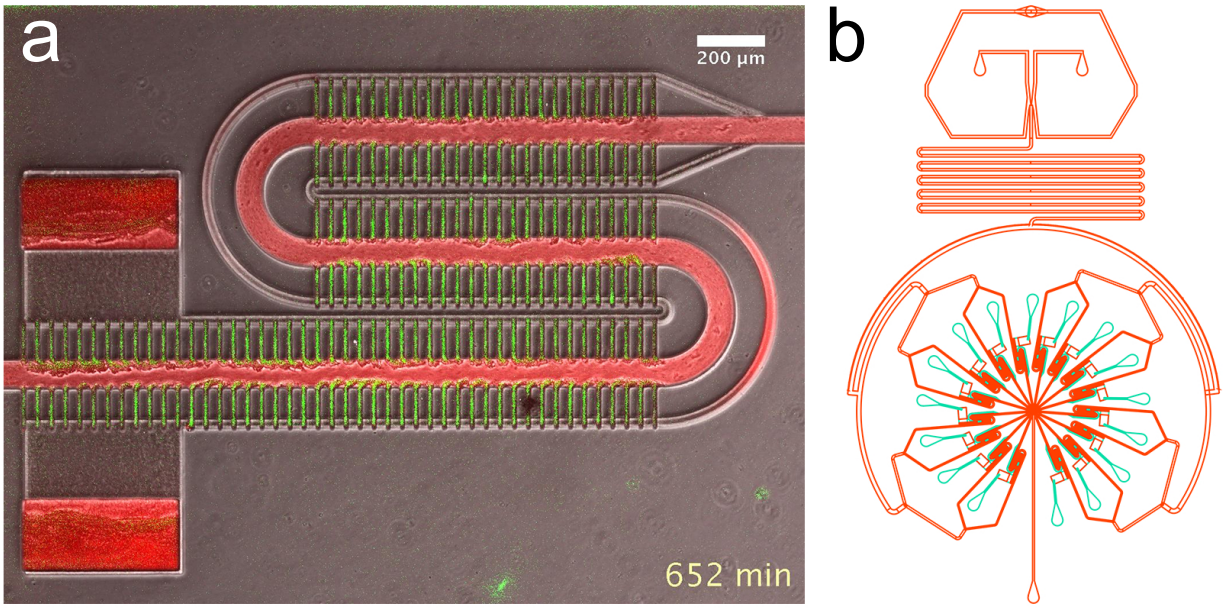


Figure 2.3: A lyophilizable, multistrain biosensor microfluidic device. (a) An instance of the biosensor design initially derived through iterative FEA analysis, successfully growing a loaded with a biosensor strain that is fluorescing in the present of 1 uM sodium arsenite. (b) A 16-strain full-biosensor chip design with loading ports (cyan) and lyophilization reservoirs visible (rectangles). The arrays in this device were based on the successful design in (a).

To test these cryoprotectants, biosensor strains were grown overnight to stationary phase. The strains were then double-washed in cryoprotectant and concentrated to 50x their batch culture concentration. After lyophilization in a commercial freeze-dryer for 12 hr, the strains were stored in anaerobic, nitrogen-flushed, desiccated, and light-free packaging at room temperature to protect from viability-reducing oxidative and photo-oxidative reactions.

Relative cryoprotectant efficacy was determined via plate reader revival experiments performed 24hr, 1 week, 2 weeks, 4 weeks, and 8 weeks after lyophilization. Cells were revived via rehydration and resuspension in 200μl of revival medium within their respective microplate wells. The plates were then immediately placed into a Tecan Infinite M200 Pro plate reader, where growth rates were monitored over the following 48 hours.

Revival media included:

1. M9+0.4% glucose+spectinomycin

2. Trace Select M9+0.4% glucose+spectinomycin

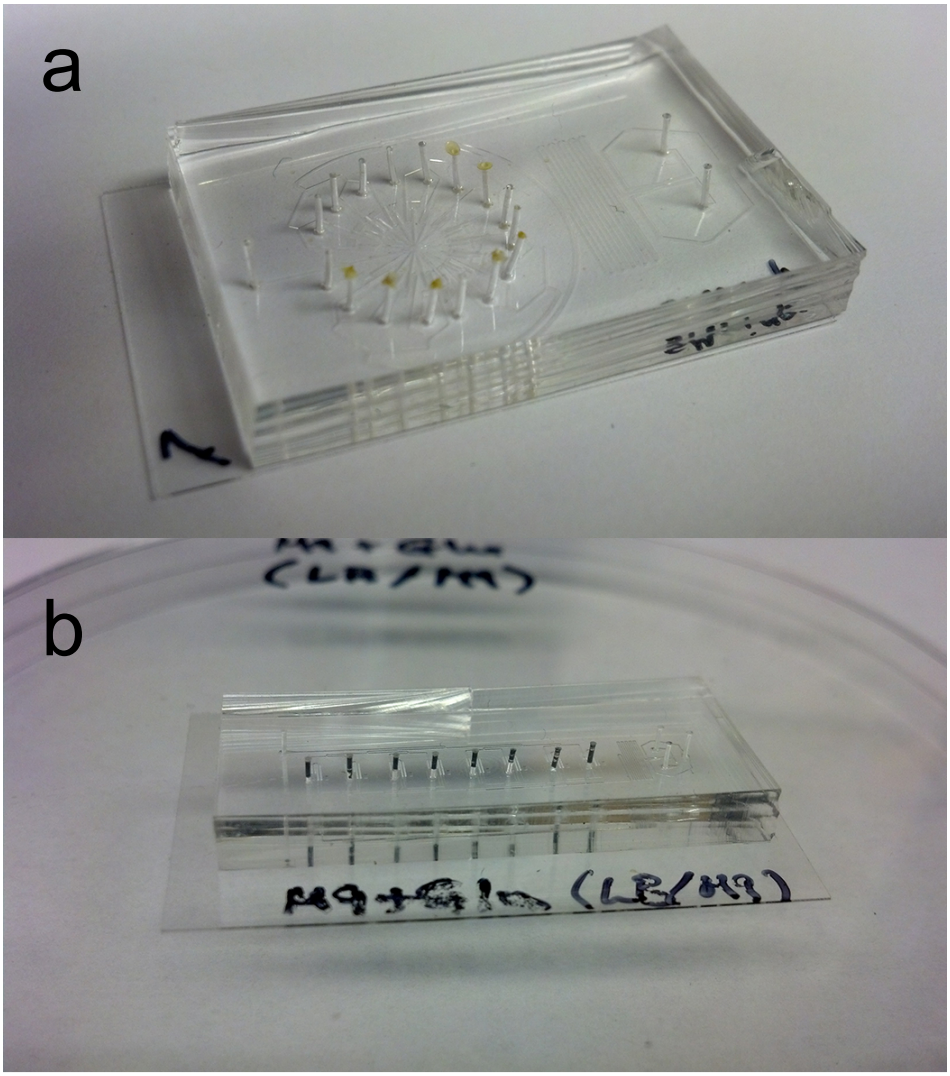


Figure 2.4: Multistrain, in-chip freeze-drying. (a) A 16-strain biosensor chip with cryoprotected, independently-loaded, and lyophilized strains. The loading ports have not yet been sealed with silicone elastomer. (b) A different lyophilizable biosensor device with loading ports sealed with quick-curing PDMS, which is visible as black material blocking the loading port columns.

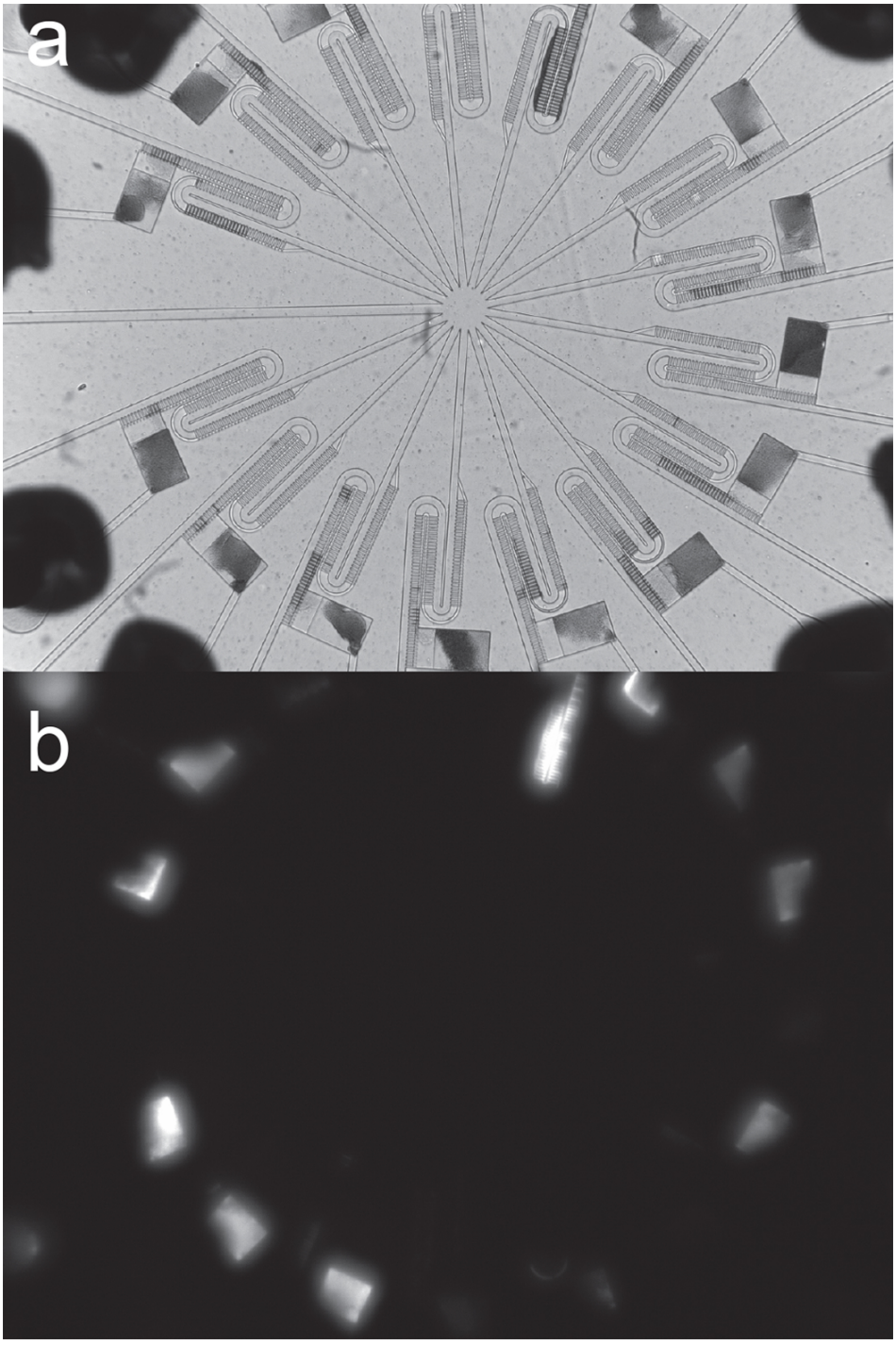


Figure 2.5: A 16-strain biosensor chip with cryoprotected, independently-loaded, lyophilized, and revived strains. (a) A transmitted light view of the strain arrays and sealed loading ports. (b) Reservoir FL that occurrs upon revival.

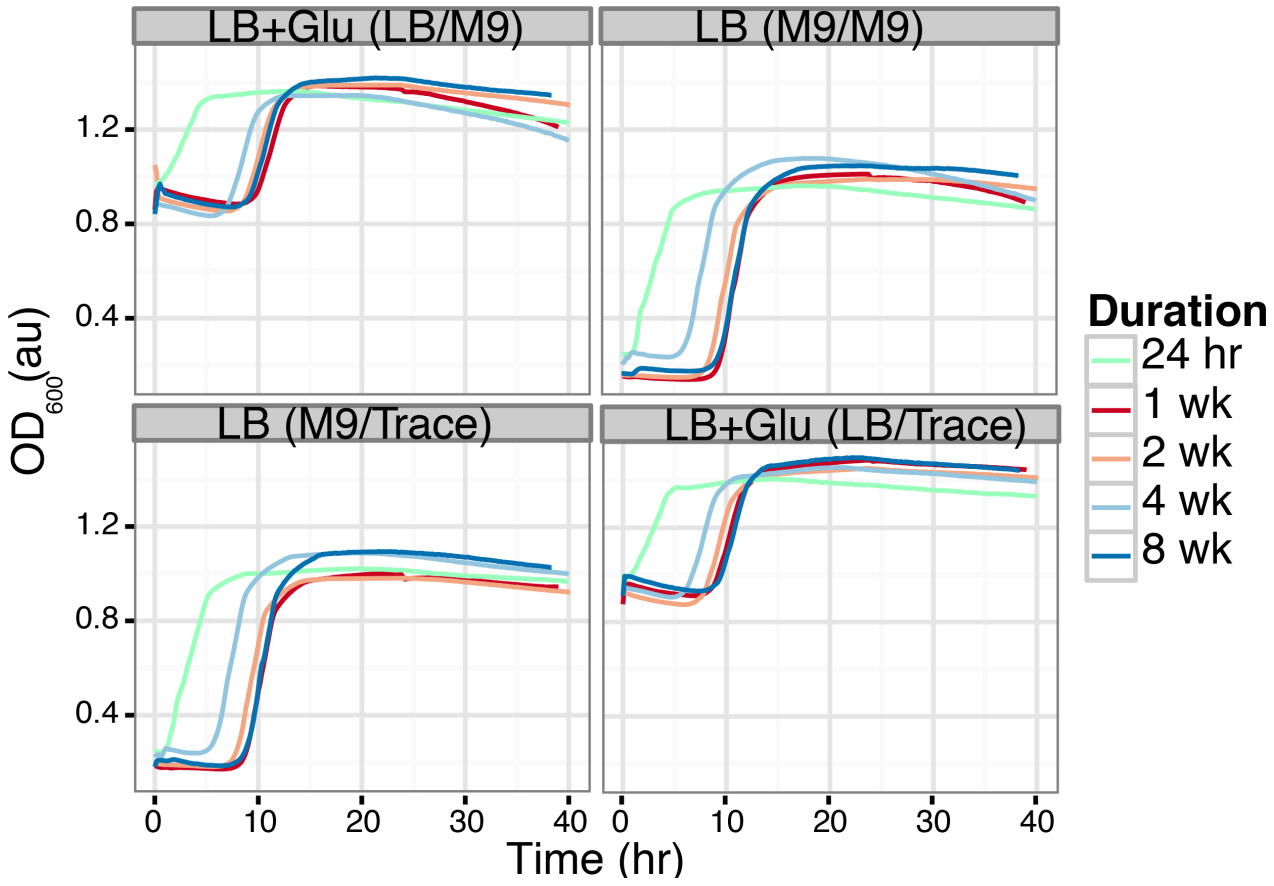


Figure 2.6: Batch cryoprotectant testing in Escherichia coli. Over 32 combinations of different growth, cryoprotectant, and revival medias were tested for their freeze-dry revival efficacy for durations ranging from 1 day to 8 weeks. The plots in this figure depict four of the best combinations. The first media combination listed is the cryoprotectant media; the two in parentheses are the growth and revival medias, respectively. Thus, LB+Glu (LB/M9) indicates that the cells were freeze-dried in Luria-Bertrani Broth plus 0.4% glucose after being first grown to confluence in LB; they were revived using M9 minimal media.

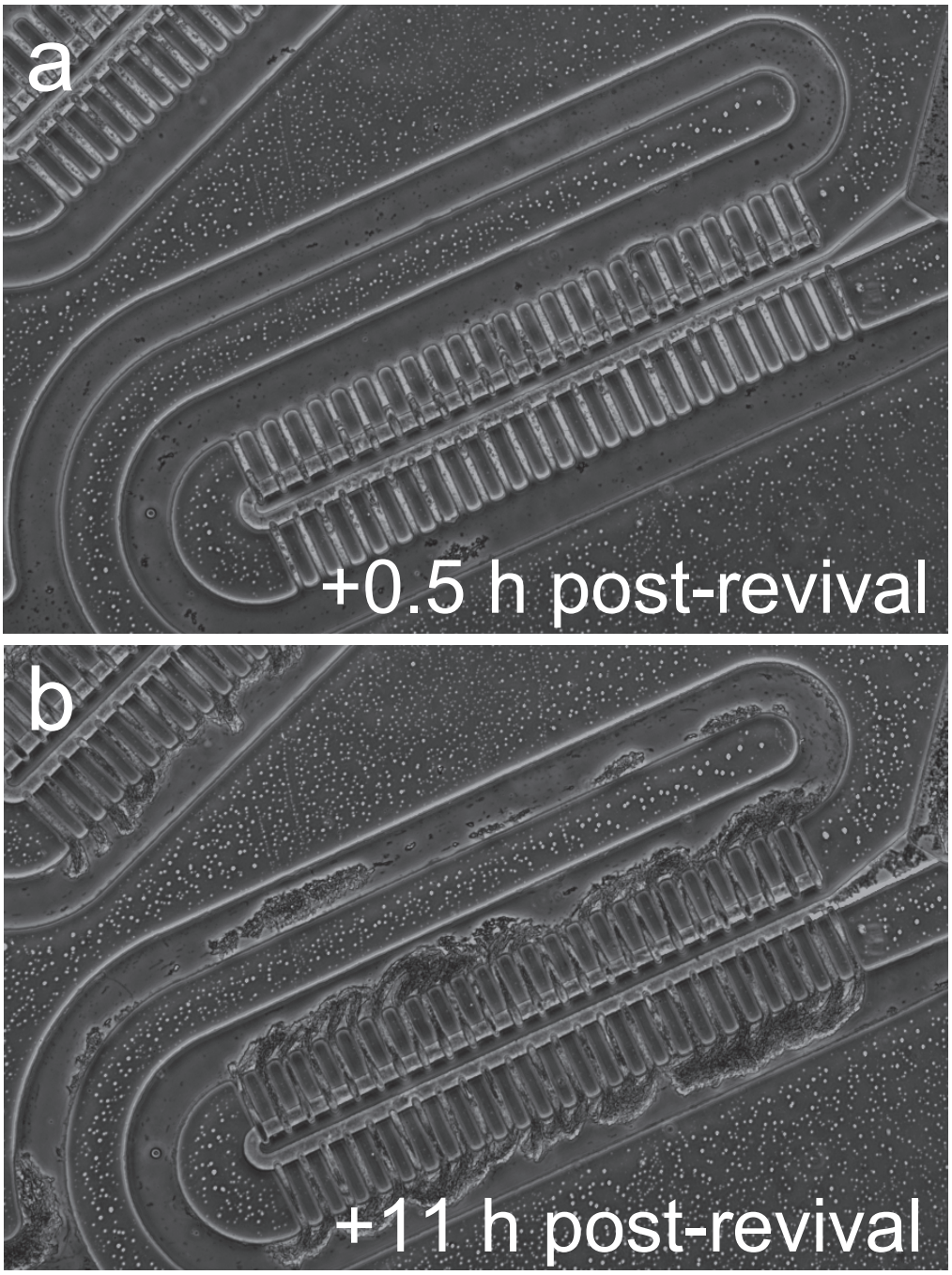


Figure 2.7: Intra-chip revival of lyophilized *Escherichia coli* biosensor strains. (a) A strain array half an hour after rehydration with growth media. (b) Robust growth in the same array eleven hours after rehydration.

3. Trace Select M9+0.4% glucose

4. HM9 (nitrate) + 0.4

and were selected to be representative of the growth media used in the final device. Strains protected with optimal cryoprotectants showed little difference in viability between cryoprotectants after two months of preservation.

The best cryoprotectants, including LB + 0.4% glucose and LB + 0.4% sucrose (see Fig. 2.6), were used to perform on-chip lyophilization with successful shelf-lives of at least eight weeks and probably greater.

A biosensor chip loaded with biosensor strain pLB-Amm3 (LABBS31), lyophilized with optimal cryoprotectants, and protectively packaged according to the protocol above was even transported to the Mojave Desert and exposed to rough conditions for 48 hours, followed which it was stored indoors at room temperature for an additional 72 hours. Temperatures to which the chip was exposed ranged from near-freezing up to +35C. All reservoirs of lyophilized ammonia-sensing strains revived following rehydration with media.

All on-chip revival occurred via rehydration in Trace Select M9+0.4% glucose+spectinomycin, HM9 (ammonia) + 0.4% glucose, and HM9 (nitrate) + 0.4% glucose via de-gas driven chip wetting and subsequent gravity- or pump-driven flow. Initial signs of revival occurred on time scales equivalent to those from the plate reader experiments.

The described protocols constituted novel methods to load multiple, specifically-engineered biosensor strains and lyophilize them on-chip, neither of which, to the author's knowledge, had been done previously. These new protocols render the biosensor device potentially field-implementable and, thus, commercially viable.

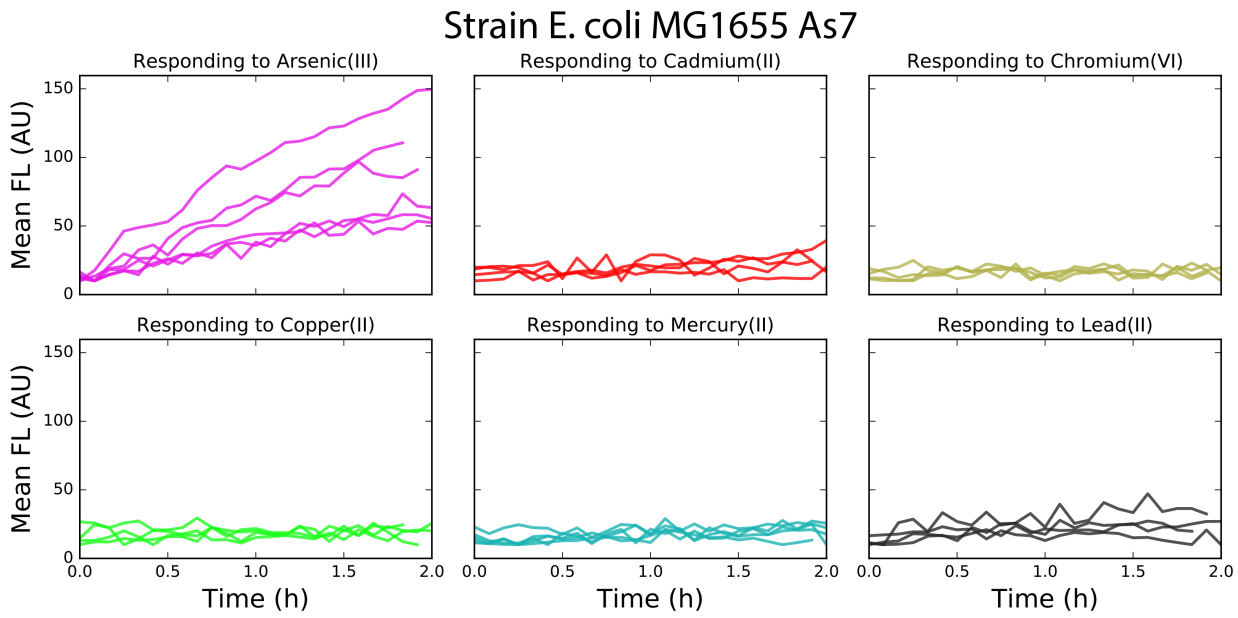


Figure 2.8: Response of an engineered biosensor strain to single-toxin induction. A single biosensor strain, engineered to sense arsenic, reliably fluoresces in response to induction by arsenic and does not fluoresce when induced with any of five other single metal toxins. The responses in this figure appear orderly and trivial to interpret.

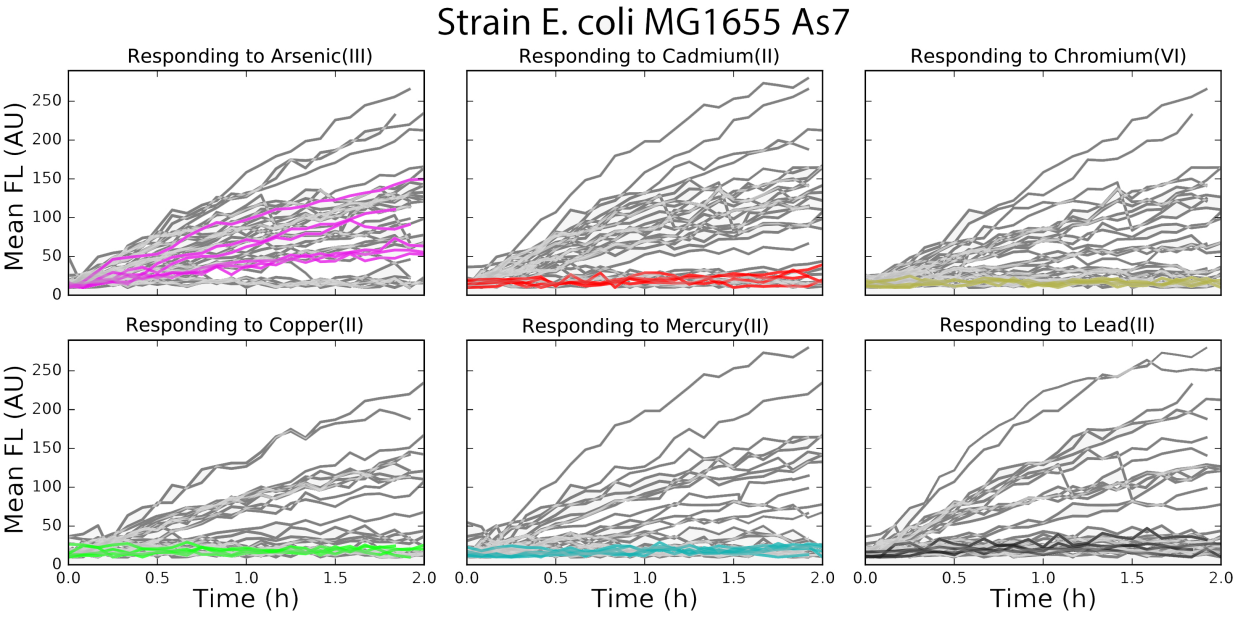


Figure 2.9: Response of an engineered biosensor strain to multi-toxin induction. The previous figure depicted the same strain responding to single-toxin inductions. When induced with multiple toxins simultaneously, the strain's responses become more difficult to interpret. In order to use all information made available by the biosensor strains to determine which metals were present, we employed machine learning classifiers.

2.1.2 Multiclass, multilabel machine learning-classification of biosensor data

Motivation and Multiclass, multilabel machine learning

While the response of a single strain to induction by single toxins appears trivial to interpret, as depicted in Fig. 2.8, responses of sensor strains to multi-toxin inductions can be non-intuitive, as seen in Fig. 2.9

The ability to discriminate between contamination with single or multiple toxins is a desirable quality for any biosensor, since heavy metal contamination usually involves multiple metals presented simultaneously. This phenomenon occurs as a result of the contamination's causal mechanism, which is usually a strongly non-neutral pH that leaches metal ions out of the water's surroundings, and was seen prominently in recent water contamination events in Flint, Michigan and the Gold King Mine Spill outside of Durango, Colorado in August 2015.

Since discrimination and classification constitute supervised learning problems, a biosensor with multiple specifically engineered strains is an ideal application for the implementation of machine learning classifiers. Creating a classifier involves training computer models to correctly assign instances, such as a heavy metal induction, to a correct class or set of classes (such as "contains arsenic" with a single toxin induction or "contains mercury, lead, and copper" with a multi-toxin induction), as seen in Fig. 2.10.

In the following section I describe how I implemented a process for engineering and extracting features from biosensor fluorescent time-series data; trained, tested, and validated the performance of a variety of machine learning algorithms, including a novel accuracy metric for the multiclass, multilabel setting; created a method for selecting an optimized suite of sensor strains for a variety of applications; and validated the long-term performance of the sensor via the quantification of sensor drift.

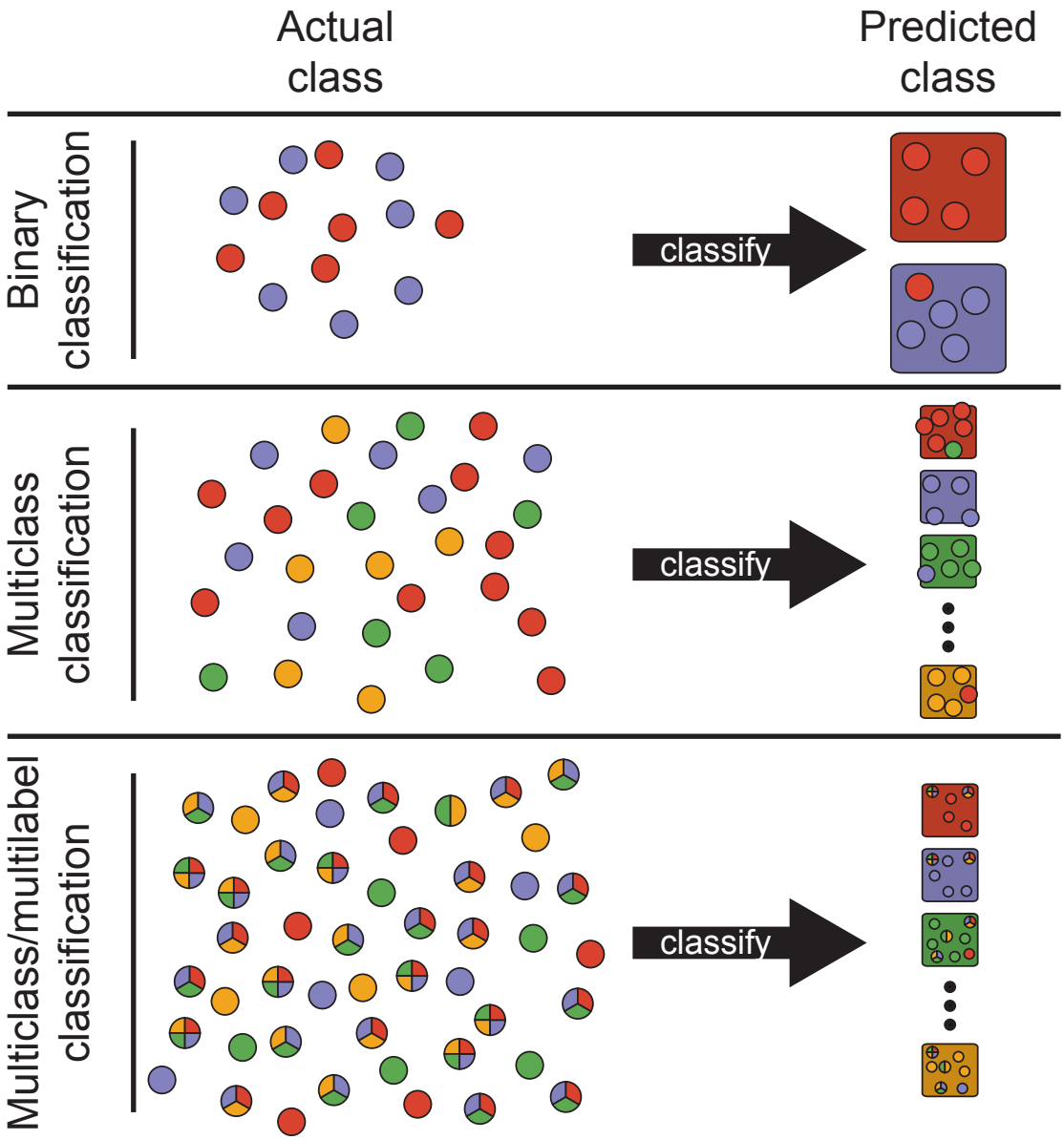


Figure 2.10: A graphical analogy for machine learning classification. In any classification problem, the goal is to train a classifier to correctly assign the maximum number of labels to any instance. Here, the instances are represented by circles. Their ground truth label (or label set, in the multiclass case) is represented by their color or colors. Correctly classifying an instance is represented by the assignment of a circle into the square of the same color. An incorrect classification is represented by an assignment of a circle to a square of an incorrect color. Binary classification involves only two classes, represented here by either red or blue, and an instance can only ever belong to one class. Multiclass classification involves instances pertaining to three or more possible classes; an instance can still only belong to one class. Multiclass/multilabel classification involves instances pertaining to three or more classes, in which an instance can belong to one or more of the classes simultaneously. In the multiclass/multilabel, even simply determining a classifier's performance can be nontrivial.

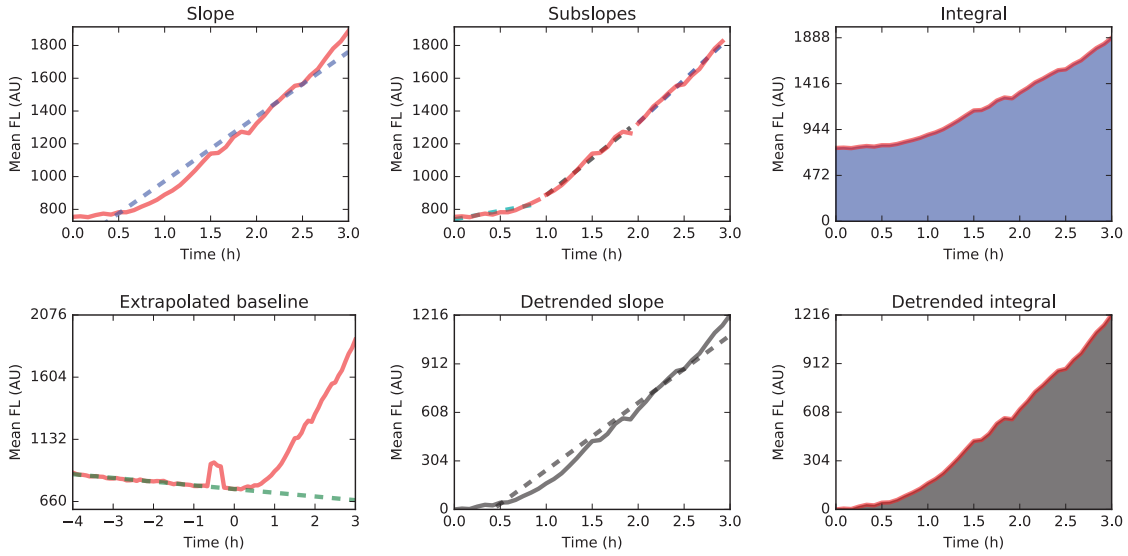


Figure 2.11: Five distinct features used in biosensor classifier construction. The features, slope, subslopes, integral, and detrended slope and integral, were used in training and testing machine learning classifiers to discriminate between multitoxin inductions. Slope uses the linear regressively-fit slope, here fit over a three hour window. Subslopes uses the same linear regressively-fit slope, but here fit on sub-intervals of the same time window, which gives a more complete picture of the dynamics during this interval. The integral feature calculates the numerical integral over the same time window. The detrended slope and integral features first detrends the mean fluorescent intensity using the extrapolated baseline before calculating slope and integral; the extrapolated baseline is calculated by linear regressively-fitting a line to the baseline trace over a certain pre-induction time window and then extrapolating that line over the full induction window.

Feature Engineering

A critical step in supervised learning is the identification of a feature set that preserves signal information while reducing noise and, if possible, reduces the dimensionality of the data without excessive information-loss. To achieve this goal, I engineered and extracted a set of features, as seen in Figure 2.11, from the biosensor strains' fluorescent responses for use in classifier training and testing. To do so, I first calculated the "slope" feature of the strain response over the full exposure window by performing a linear regressive fit to the FL time-series values during that time (Fig. 2.11a). Then I divided the full exposure window into three one-hour subwindows and calculate a "subslope" feature for each time window in the same manner (Fig. 2.11b). Third, I calculated the "integral" feature as the numerical integral under the FL response curve (Fig. 2.11c). Next, I generated two additional "detrended" features by calculating a linear regressive fit to a "baseline" 3-h window of the response trace prior to the tracer dye pulse and extrapolating it forward through the induction window (Fig. 2.11d). I calculated the "detrended slope" feature by subtracting this baseline from the FL response within the induction window and then performing a linear regressive fit (Fig. 2.11e). Finally, I calculated the "detrended integral" feature as the numerical integral under the detrended GFP response curve (Fig. 2.11f).

Classifier training

Random forests (RFs), support vector machines (SVMs), and neural networks (NNs) are common supervised machine learning algorithms used in multiclass/multilabel classification. Each algorithm has unique advantages and disadvantages for both implementation and classification, depending largely on the data set of interest. I implemented and cross-validated all three varieties of classifiers on both small-device and mid-scale device data, with anywhere from 14-32 strains. All three outperformed a random classifier on the dataset by a statistically significant margin. RFs and NNs showed the most promise at classifying the biosensor data, as shown by the classifier metric comparisons in Table **??**.

Random forest classifiers are supervised machine learning algorithms built on aggregate collections of decision tree classifiers, developed in 2001 by Leo Breiman and Adele Cutler. RFs are especially suited to small data sets, which can be loosely defined as data sets where the dimensionality of the feature vector is roughly the same order of magnitude as the number of examples. A decision tree, the canonical example of a "weak learner" classifier, is fit to a dataset by passing each element of an instance's feature vector through a sequence of if-then rules. The optimal rule is decided upon by which decision rule yields the largest information gain at that node, which is usually measured by the accuracy the tree's classification ability up until that node. Decision trees can be grown to their maximum depth to perfectly fit all data in a training set, but allowing unconstrained growth can lead to overfitting.

Thus, it is usually necessary to constrain maximum tree depth and the maximum number of leaf nodes. However, adding these constraints usually severely restricts a decision tree's classification ability, reducing it to barely outperforming a random classifier; this type of classifier is known as a "weak learner". A random forest is a collection of many "weak learner" decision trees. Each decision tree in a random forest is trained by randomly subsetting, with replacement, examples from the training set and training the tree on that subset, within the algorithm depth and node constraints. Combining many "weak learner" trees creates a "strong learner," which avoids overfitting the training set while simultaneously performing well when classifying irregular data. Additionally, random forests do not require that data is standardized to mean 0 and variance 1 prior to training and testing, which results in a significant computation time-gain over SVMs and NNs. The number of trees in the forest is usually defined by the user.

My decision trees were implemented from the open-source library scikit-learn's Ensemble module's RandomForestClassifier() class (scikit-learn v0.17.1). Classifiers were 4-fold cross-validated over the two-dimensional hyperparameter surface corresponding to number of leaf nodes and tree depth, where maximum leaf node number was tested across the range $[2^1, 2^{10}]$ and maximum tree depth was allowed to vary across $[2^2, 2^{10}]$. Maximum multiclass/multilabel F_β score was used to evaluate forest performance. The hyperparameters associated with the forest possessing the

maximum average F_1 score across its folds were reported. Generally, random forest classifiers performed the best on the feature sets. This result is expected, given the relatively small number of data points as compared with feature vector dimensionality.

Support vector machines classify data by separating classes via an optimally-separating maximum-margin hyperplane in the feature space or a higher- or lower-dimensional derivative thereof. The algorithm was originally proposed in 1963 by Alexey Chervonenkis and Vladimir Vapnik and received more renewed attention in 1992 when application of the kernel trick allowed SVMs to classify using kernel functions, which are a type of positive-definite functions that can be designed to map a point into a higher-dimensional space. Depending on whether a data set is seperable or not, this mapping usually allows the algorithm to compute a hyperplane that separates the data classes in the higher-dimensional space. The data and the hyperplane, once fit, can be transformed back to the original feature space. The algorithm utilizes a small number of vectors close to the decision boundary to compute the optimal hyperplane parameters. Additionally, this classification method incorporates advantages of both regression and k-nearest neighbors classification, while dealing with non-linearly separable data in a computationally efficient manner. However, it usually performs poorly with small data sets.

I utilized scikit-learn's SVM module's C-Support Vector Classification class, implemented in my parallelized cross-validation module to optimize for the classifier's hyperparameters. Using a radial basis function-kernel, with hyperparameters C and γ optimized by searching between $[2^0, 2^{20}]$ and $[2^{-15}, 2^1]$, respectively. Again, maximum average multiclass/multilabel F_1 score was used to discriminate the optimal parameters. As expected, while SVMs fit and trained on the data significantly better than a random classifier implemented with the class prior probabilities, it was outperformed by both random forests and neural networks.

Artificial neural networks, referred to alternatively as neural networks or multilayer perceptrons, were developed in 1950s and 1960s and became popular in the 1980s with technological advances and with the discovery of the backpropagation weight-optimization algorithm. Originally proposed as an attempt to mimic the brain's neural circuitry, a neural network is formed of the

following: an input layer of nodes, where each node corresponds to an element of a feature vector; one or more hidden layers of nodes, which may have more, less, or an equal number of nodes as the input layer; and an output layer, corresponding to the predicted label. When an instance requiring classification is passed from the input layer to the first hidden layer, its elements are multiplied by a unique set of weights depending on which node it is being passed to. The weight elements are then summed and passed through an activation function to produce a scalar output for that specific node in the hidden layer. That hidden layer then becomes the input layer for the next hidden layer and the process is repeated. Finally, depending on the classification context, the combined sums are passed from the final hidden layer to the output layer, where a post-processing function is applied. In order to train a NN, stochastic gradient descent or a comparable minimization method is used to optimize the node weights based on a selected cost function, which can include a weight regularization function. Neural networks are able to adeptly learn nonlinear class structures and to update themselves in real-time as additional data points become available, making them an excellent choice for the biosensor.

A neural network was implemented using the beta version of scikit-learn v.0.18.0's neural network module's MultiLayer Perceptron (MLP) Classifier. Each class label was limited to using a 3-layer multilayer perceptron utilizing the rectified linear activation function, L2-weight regularization, the limited-memory Broyden-Fletcher-Goldfarm-Shanno weight optimization method, and inversely varied the learning rate proportional to the square root of the time step. All data was scaled to have mean 0 and variance 1 prior to training the classifier or predicting the point's class. Surprisingly, the neural network appeared to compare favorably with or outperform random forests across all feature sets. This result was unexpected, since random forests generally perform better when classifying data with small training sets.

Classifier performance

The performance of each version of the trained classifiers is best represented by the F_1 score, which is a measure of the test's accuracy computed as a weighted average of its precision and recall.

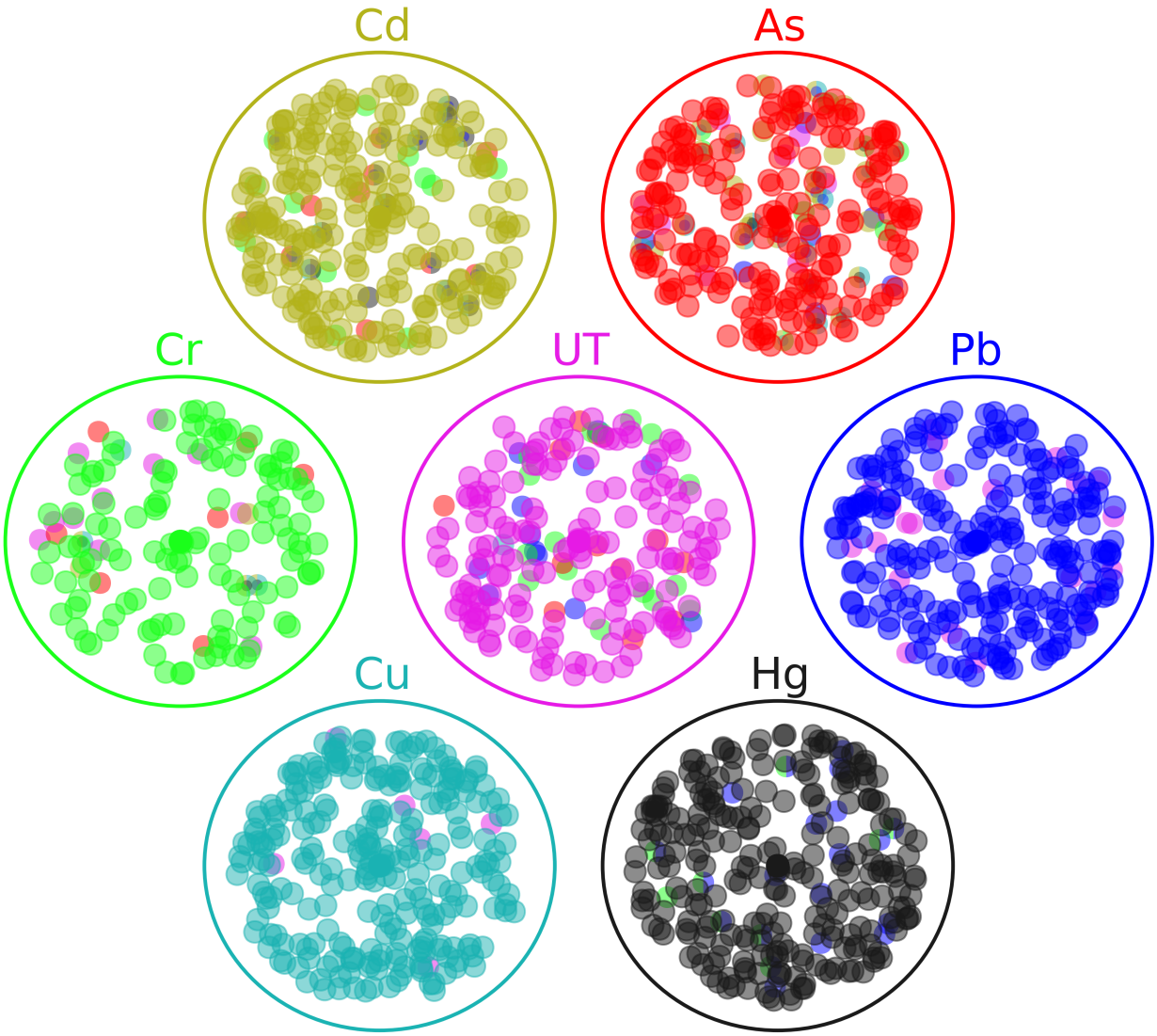


Figure 2.12: Multiclass, multilabel classifier performance visualization on biosensor data. This figure represents a random forest classifier classifying multitoxin inductions. Correct assignment of a label to an induction is represented by a shaded circle being placed into larger circle of the same color. For instance, a triple toxin induction with arsenic, lead, and mercury would result in each of a small red circle, blue circle, and black circle being placed in the larger red arsenic, black lead, and red mercury circles, respectively. A misclassification event, where an induction is incorrectly assigned to the wrong class, is depicted by a smaller multicolored circle, representing the true classes for that induction, placed in the larger circle representing the class to which the induction has been incorrectly assigned.

Table 2.1: F_1 scores for both random forest (RF) and support vector machine (SVM) classifiers trained on the full feature set and individual features, including and excluding the existence of a "water" class. The highest F_1 scores for RF and SVM classifiers are boldfaced. Rows are sorted by highest-to-lowest RF F_1 score to identify the most valuable features to successful classification.

Feature Set	Water Class Included?	RF F_1 score	SVM F_1 score
Full	No	**0.789**	0.646
Slope	No	**0.789**	**0.759**
Detrended slope	No	0.760	0.755
Subslopes	No	0.759	0.645
Slope	Yes	0.745	0.695
Full	Yes	0.744	0.545
Detrended slope	Yes	0.724	0.725
Detrended integral	No	0.706	0.695
Subslopes	Yes	0.668	0.633
Detrended integral	Yes	0.587	0.619
Integral	No	0.447	0.513
Integral	Yes	0.262	0.378

Table 2.1 displays F_1 scores for both RF and SVM classifiers trained on the full feature set and individual features, including and excluding the existence of a "water" class in addition to six classes for the primary toxins and one class for the secondary toxins. Generally, classification is improved by including the "slope" feature in the feature set and minimizing the number of sorting bins by excluding the "water" class. The highest-performing classifier, with an F_1 score of 0.789 (boldfaced), uses RFs, excludes the water class, and uses either the "full" set of features or only the "slope" feature. The highest-performing SVM classifier, with a slightly lower F_1 score of 0.759 (boldfaced), also excludes the "water" class and uses only the "slope" feature. In scenarios where it is important not only to discriminate between individual toxins present but also to discriminate between individual toxins and pure water, the highest-performing RF and SVM classifiers had slightly lower F_1 scores of 0.745 and 0.725, respectively.

In addition to determining which feature set maximizes classifier performance, the F_1 scores in Table 2.1 allow us to rank the contribution of each individual feature to the overall performance of the classifier. Rows in Table 2.1 are sorted by highest-to-lowest RF F_1 score, and this ordering reveals a crude ordering of decreasing feature value as "full"/"slope," "detrended slope," "subslopes," "detrended integral," and "integral."

Generally, RF classifiers outperformed SVM classifiers on the data set, which is unsurprising

given its relatively sparse nature. SVM classifiers, however, generally outperform RFs on larger datasets. Therefore, we expect the performance of SVM classifiers to gain on RF classifiers as more training examples are generated. Overall, given the extremely challenging nature of multi-label, multi-class classification problem, the classifiers have performed exceptionally well.

As a consequence of operating on, by machine learning standards, a relatively small data set, it was necessary to establish a baseline for classifier performance in order to truly discern whether classification was indeed occurring, or whether classifier performance was a spurious artifact of this particular data set. To establish such a baseline, we reasoned that a user could expect a machine learning algorithm that correctly and generalizably classifies data to significantly outperform a random classifier classifying the same data set. We therefore constructed a random classification algorithm that first calculates the class prior probabilities for each label and then randomly assigns or does not assign these labels to each test instance, according to the labels' calculated priors. The random classifiers acting on both Y01 and synthetic water data both had non-zero F_1 scores. Thus, the metrics from the random classifiers serve to establish the necessary baselines.

Additionally, another problem emerged while classifying unstandardized data using the MLP classifier. The machine learning community widely recognizes MLPs as being unable to classify unstandardized data in a meaningful way; however, when classifying unstandardized data, we saw that the MLPs' F_1 scores increased slightly above the F_1 scores for their associated random classifiers. This result prompted us to derive a new multilabel, multiclass metric, which we have named *multilabel accuracy* (MLA). For N classified data points, c labels, $\vec{T}$ and $\vec{P}$ representing the total set of ground truth label vectors and predicted label vectors, respectively, and $\vec{t}$ and $\vec{p}$ representing the same for a given data point, respectively, we define the MLA as:

$$A\left[\vec{T},\vec{P}\right] = 1 + \frac{\sum_{i}^{N} 2(\vec{t}\cdot\vec{p}) - \vec{1}\cdot(\vec{t}+\vec{p})}{N\cdot c}$$

Whereas the canonical F_1 score only takes into account true positives, the MLA metric rewards both true positives and true negatives. Thus, a successful machine learning algorithm will

see a positive, significant increase in both metrics over the metrics produced by a random classifier.

Chapter 3

Explainable multiclass machine learning on genomic time series with applications to the detection of heavy metal contamination

3.1 Introduction

While Chapter 2 demonstrates the effectiveness of engineering a specific heavy metal biosensor, the goal of the DynOMICS system was to build a general system that could learn a variety of dynamic *E. coli* transcriptional response patterns in order to detect substances of interest (Fig. 3.1). To accomplish this, we used the well-known *E. coli* fluorescent transcriptional library that was built by the Uri Alon lab in 2006 [120]. A Singer ROTOR cell-handling robot was used to load the library from agar plates into the microfluidic chips, where they were kept fluidically isolated in uniquely-indexed cell traps.

Once a microfluidic chip had been loaded with the library, media and waste lines were plugged into its inlet and outlet ports (see Fig. 3.1a). Fresh media, driven by gravity, would flow into the media inlet, which is denoted by the star at the top of the device. Having entered the chip, the media would then flow into a manifold channel structure, passing the cell spotting and trapping

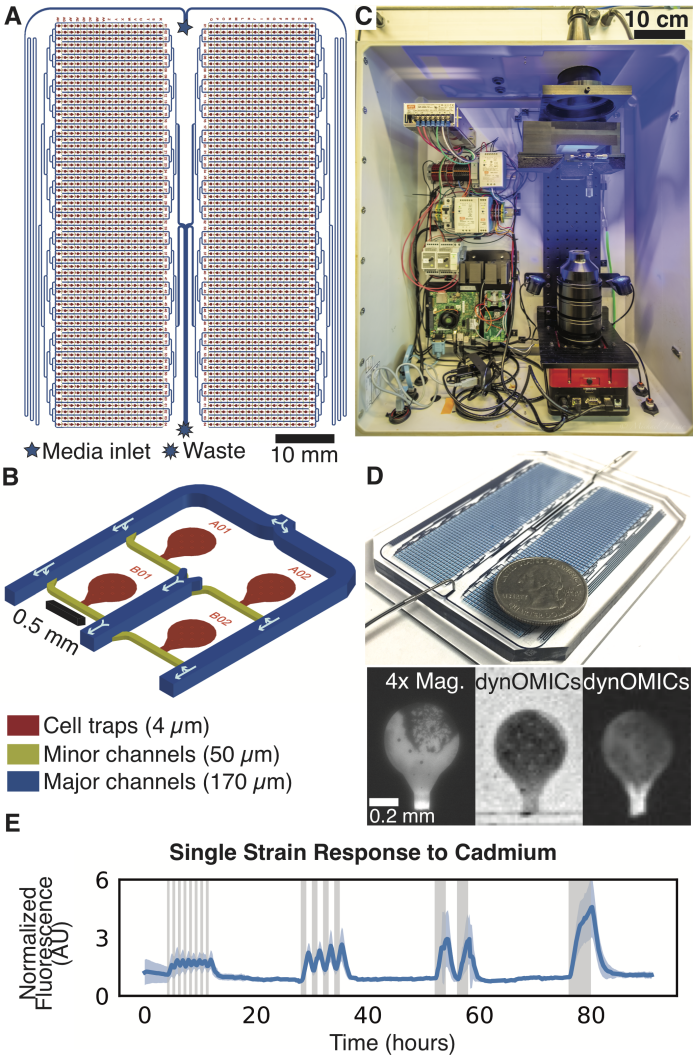


Figure 3.1: An overview of the DynOMICS system. (a) A schematic of a 2K-strain DynOMICS microfluidic chip: This CAD design schematic of a DynOMICS 2K-strain microfluidic chip shows the general channel and cell region organization, as viewed from above the device. (b) A view of a DynOMICS' chip's flow channels and colony regions: This CAD-generated three-dimensional image is an accurate representation of the structure and flow patterns of a DynOMICS microfluidic chip. At each channel branching, the media flow patterns are explicitly depicted with white arrows. (c) The interior of a DynoMICs device: This photograph is of one of our three custom-built DynOMICS devices, showing the optical, electrical, and computer systems housed inside. (d) A dye-loaded PDMS DynOMICS chip and TL/FL images from DynOMICS optics: At the top of this image is a photo of one of our fabricated PDMS chips, with a US quarter placed on its surface for scale. Media inlet and outlet lines are connected to both of the chip's ports. The chip itself has been vacuum loaded with blue dye. At the bottom is a comparison of a fluorescence image taken at 4x magnification on a lab-grade Nikon microscope and the same image taken using our DynOMICS device. (e) An example of a DynOMICs time series: DynOMICS is capable of producing thousands of similar time series in a single experiment. Here, cadmium inductions are denoted by the grey regions of the plot. The averaged response of several redundant positions for the cadmium-sensitive strain *zntA* is depicted by a blue line.

regions along the way. In flowing past the cell traps, the colonies were thereby supplied with a fresh supply of nutrients, while waste products and excess cells were transported away. The media, excess cells, and waste products exited the chip through a series of combining outlet channels, eventually arriving at the waste port, which is denoted by a many-pointed star at the base of the schematic.

A more detailed view at the structure of channels and cell trap regions can be seen in Fig. 3.1b. Media enters the major manifold channels (blue) from a series of branching channels that originate from the media inlet. As the media flows down the major channels, it filters through minor manifold channels (yellow), passing the cell trapping regions (red). The cells, which are spotted by the cell-handling robot into the large, red, circular regions, exchange fresh nutrients for waste products at the interface of the cell trap and minor channel.

During the course of an experiment, each microfluidic chip was housed inside one of our DynOMICs "boxes", as can be seen in Fig. 3.1c. On the right of the picture, at the bottom of the box, is the camera, which is identifiable as the red and black box. Atop the camera is its lens assembly, a black cylinder. The camera and the lens assembly point upwards, towards the stage mount and transmitted-light diffuser housing at the top-right of the box. This optical assembly has results in a microscope with 0.5x magnification and a numerical aperture of 0.048, which is approximately twice that of a normal 0.5x microscope. The wide field of view and high numerical aperture allow the camera to take high-resolution photographs, as can be seen in Fig. 3.1d. On the left side of the box is the power supplies, electrical hardware, and computer board that coordinate the device's imaging, data processing, and biological-support systems.

The combination of the 2K microfluidic chip and custom experiment platform gave us the ability to image the on-chip cells with high temporal and spatial resolution for several days to several weeks throughout the course of an experiment (Fig. 3.1d). While our standard imaging period was once every ten minutes, the device was capable of taking an image every four minutes. This temporal resolution is a significant improvement in the ability to study *E. coli* transcriptional dynamics.

After building and optimizing this much larger and more complex microfluidic device, we utilized it to produce tens of thousands of time series data. An example of one of these series, an

averaged response trace of the strain *zntA* to cadmium, is shown in Fig. 3.1e. The huge volume of dynamic, noisy transcriptional expression data necessitated that we develop an analysis strategy that combined techniques from both standard data science and from new methods in machine learning and artificial intelligence. In this chapter, we discuss the methods that we developed and employed in order to explore, classify, and understand the dynamic transcriptional data produced by DynOMICs.

3.2 Methodology

3.2.1 Experiments of interest

All initial supervised learning tasks were performed on features subsetted from the standardized set of experiments with the following experiment indices: 1711, 1726, 1742, 1778, 1784, 1794, 1798, 1808, 1822, 1823, 1824, 1835, 1836, 1838, 1844, 1848, 1924, 1952.

The experiments with the following indices were included inductions with with San Juan River water samples: 1838, 1844, 1848. Those inductions and their subsequent recovery periods on lab-grade MilliQ-filtered DI-H_2O were dropped from the training data.

Similarly, the following urban water experiments were withheld entirely from the training data set:

1. 1864: Chicago
2. 1867: San Diego
3. 1872: New York
4. 1874: Seattle
5. 1883: Miami

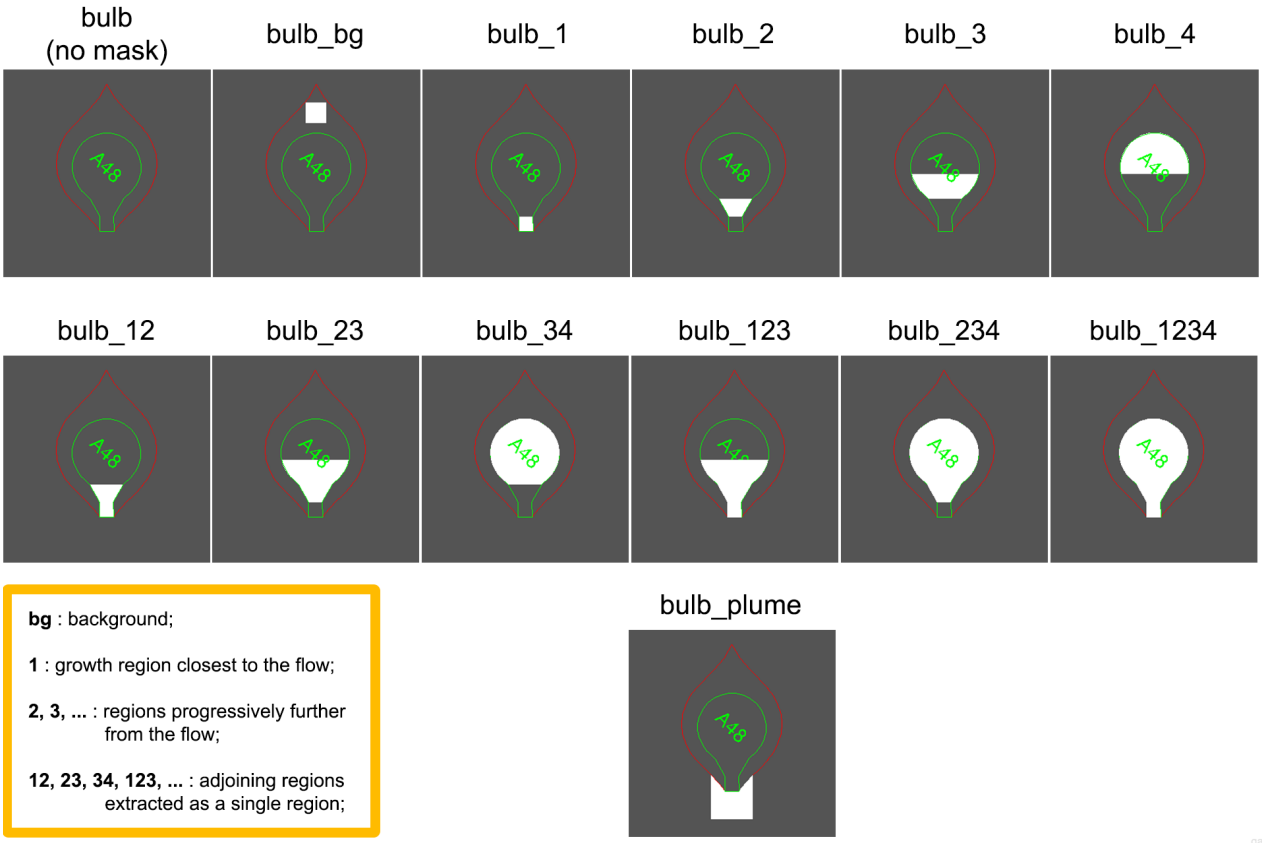


Figure 3.2: The data extraction regions of a standard DynOMICS cell trap. In these subfigures, the green light-bulb shaped region is the trap cavity, where a colony would be spotted and subsequently grown. The red teardrop outline represents the external walls of this particular trap design, which was originally a member of the *dynomics_cones_v_2_0* device. The white regions in each subplot is the named data extraction region. Note that the *bulb_plume* region is actually entirely external to the microfluidic trap. Instead, it resides within the channel where it is used to monitor biofilm formation.

3.2.2 Data preprocessing and feature engineering

Once the variability problems had been stabilized for the microfluidic designs and biofilm mitigation, all data for analysis and supervised learning was selected to come from the *bulb_1* region of our device's microfluidic traps. Figure 3.2 displays the thirteen trap regions from which signal data was extracted post-registration and flat field correction. The inset legend explains the logic behind the enumeration of each region: regions of the trap closer to the channel (which, in this case, are located at the bottom of each subfigure) have lower numbers, while regions of the trap further from active laminar flow have higher numbers. The reasons for this choice was that the *bulb_1* region displayed responses that were of greater relative magnitude and of shorter relaxation times than did the $bulb_2$ region of the traps (Fig. 3.3b).

Flat-field correction

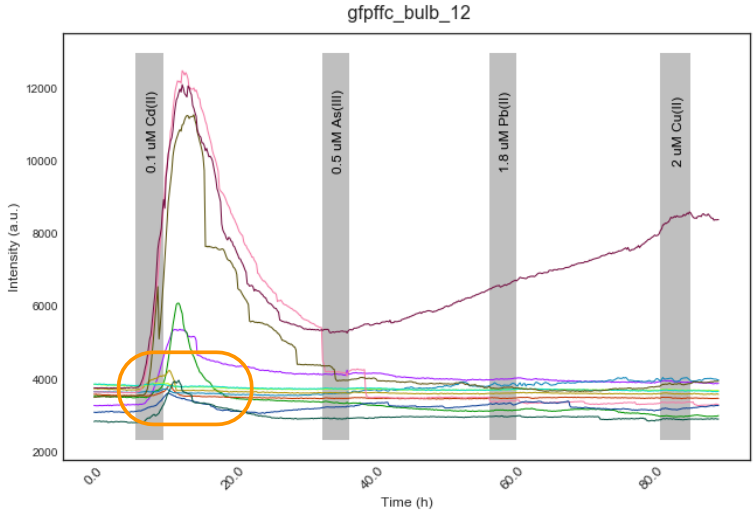
In order to correct for the uneven intensities that are the result of optical vignetting, all images were first flat-field corrected according to the automated pipeline discussed in the previous chapter (Fig. 3.4a). This occurred prior to data being extracted from any image.

Flat field correction was performed by converting the images into 16-bit grey scale *numpy.ndarrays*. Then, the following element-wise operations were performed:

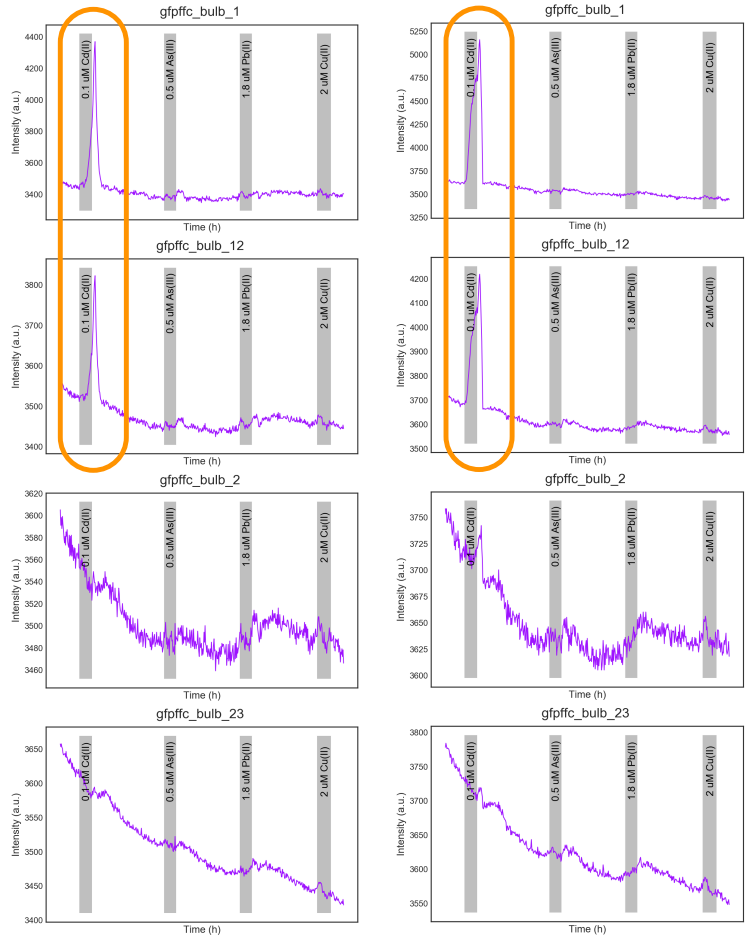
$$C = m * \frac{R - D}{F' - D'} \tag{3.1}$$

where R is the raw image to be flat-field corrected (Fig. 3.5a), D is the dark-current image for that device, taken at the same exposure settings as R (Fig. 3.5b), F' is the flat-field image for that device, not necessarily taken at the same exposure setting as R (Fig. 3.5c), and D' is the dark-current image for the same device, taken at same exposure as F'. Finally, m is the mean value for all values in the array $(F' - D')$.

The reasoning behind why this equation works is as follows. By subtracting the dark-current from the raw image, you remove the effects of any hot pixels or regions of the cameras charge-

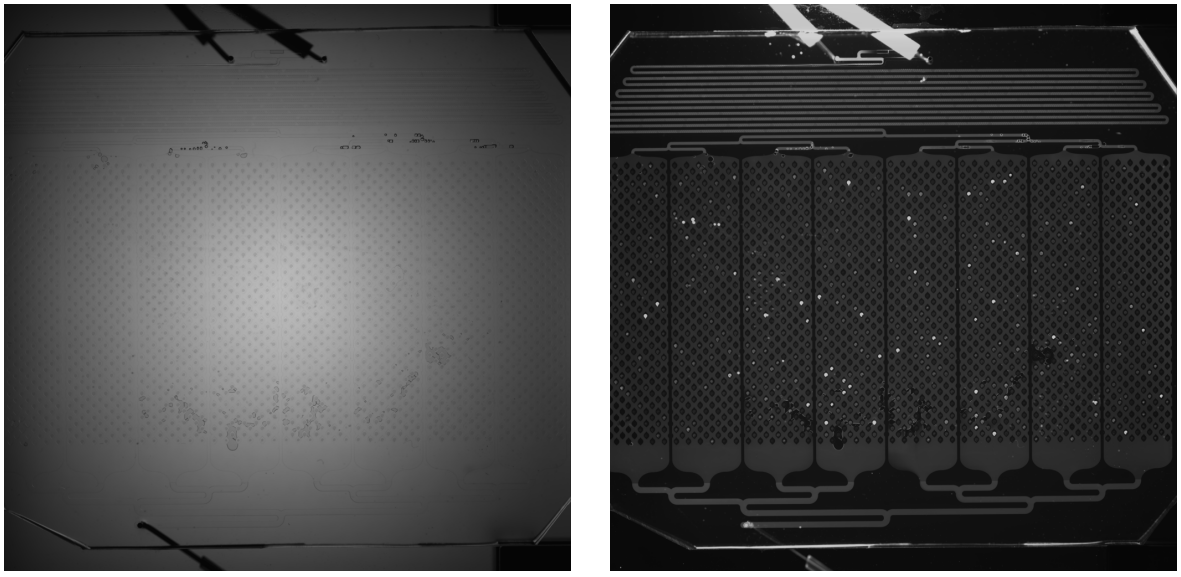


(a)



(b)

Figure 3.3: Response variation by cell trap region. Exp. 1249 demonstrates how the regions of the microfluidic traps closer to the open channel display much more pronounced responses, most likely due to that region's cells being in a more active growth state. The highlighted trajectories in (a) are the same as depicted in (b). The obvious difference in amplitudes guided us towards a derivative-based feature from the front of the cell traps.



(a) (b)

Figure 3.4: Optical vignetting in both transmitted and fluorescent channels of the DynOMICS optical system. (a) Severe vignetting of the transmitted light images. (b) The vignetting of the fluorescent channel's images was far less severe, but still enough to introduce bias to strains' signals, depending on their position on the device.

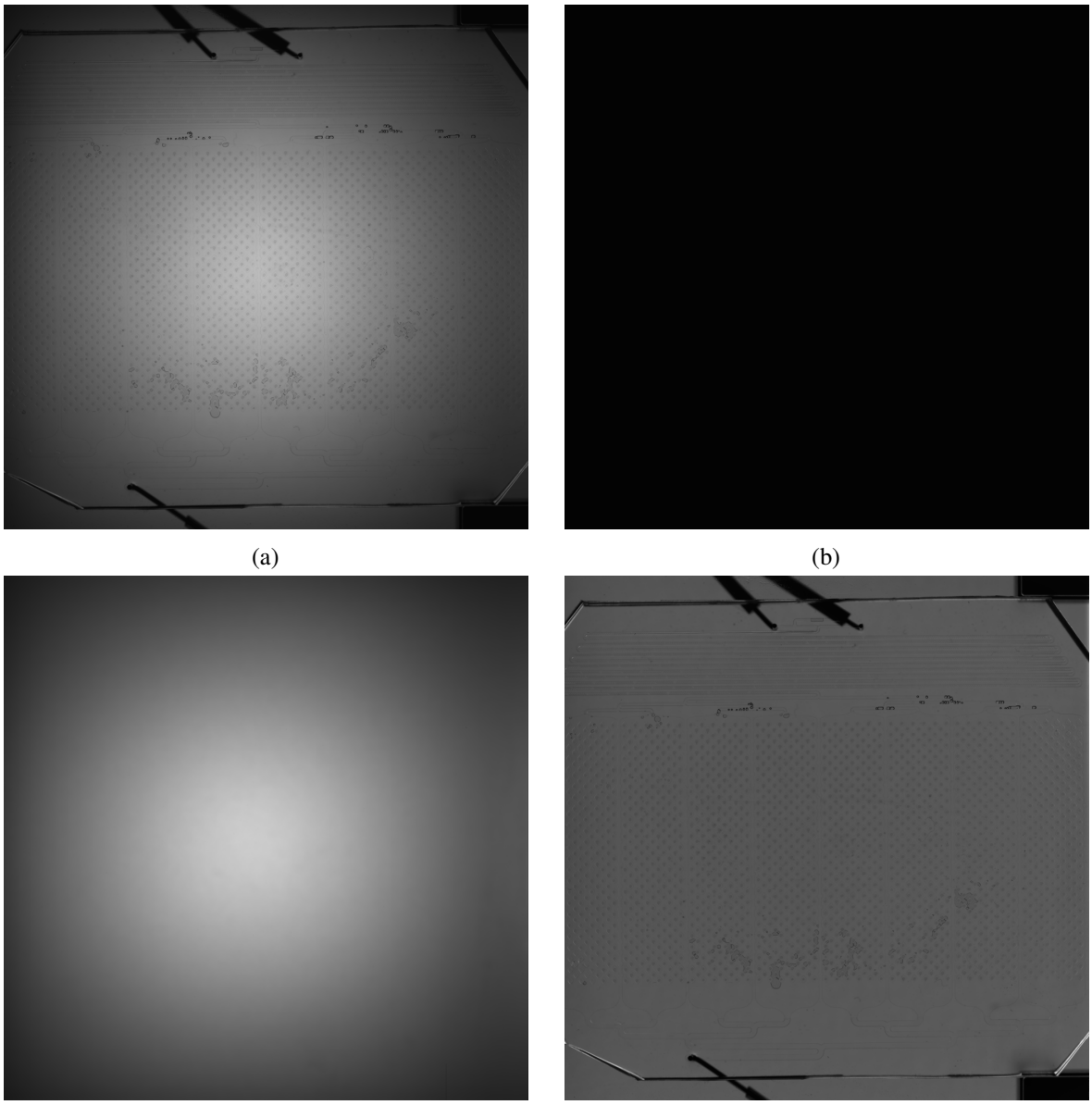
coupled device (CCD) array that may be unduly influenced by thermal noise. Then, by dividing through by $(F' - D')$, the dark-current corrected image $(R - D)$ is scaled to the interval $[0, 1]$. Since each pixel is scaled by its own dark-current corrected flat-field intensity, this step has the effect of actually correcting the vignetting caused by imperfect optics. The final step of multiplying by m scales the image back up to the interval $[0, 2^{16}]$ (see Fig. 3.5d).

Background signal removal

After substantial experimentation and thought, all such data were preprocessed in the following manner prior to any analysis or further processing:

$$x_{(t,\, s_i)} = \frac{x^{bulb_1}_{(t,\, s_i)} - x^{background}_{(t,\, s_i)}}{x^{background}_{(t,\, s_i)}} \tag{3.2}$$

where t refers to the current time point, s_i refers to the strain in device position i, $x^{bulb_1}_{(t,\, s_i)}$ is the flat-field corrected fluorescent signal from the $bulb_1$ region of position i at time t, and $x^{background}_{(t,\, s_i)}$



(a) (b)

(c) (d)

Figure 3.5: The DynOMICS automated flat-field correction process. (a) A raw TL image. The optical vignetting is apparent from the bright center and shadowed edges. (b) The TL dark current image. Taken on the same DynOMICs box and at the same exposure setting as the raw TL image, this image is taken with the camera's shutter closed. Since the shutter is closed, the only signal intensities that the camera's CCD integrates are from thermal noise or malfunctioning "hot pixels". (c) The TL flat field image. Taken with a clean glass diffuser plate as its subject, this image is a featureless representation of the bias due to vignetting. (d) The FFC'ed TL image. The same image as in (a), but now with the vignetting eliminated. The process for fluorescent channel images is analogous, but with a fluorescent quartz plate as its flat field image's subject.

refers to the flat-field corrected local background fluorescent signal at position i at time t.

The reasoning behind this preprocessing step is as follows. By subtracting the local background signal, we eliminate any local or regional fluctuations that are of an additive (or, analogously, subtractive) nature. Then, by dividing by the same background signal, we form a measure of the amplification of the signal over the local background. This final division is one method by which we reduce the box-to-box variability that inevitably occurs.

The result of this background correction was to produce a vector $\vec{x}_t$ representing the background-corrected fluorescent signals of all device positions at time t:

$$\begin{bmatrix} x_{(t,\, s_0)} \\ x_{(t,\, s_1)} \\ \vdots \\ x_{(t,\, s_{2174})} \\ x_{(t,\, s_{2175})} \end{bmatrix} = \vec{x}_t \tag{3.3}$$

Z-scoring by time point

Experiments on different boxes nearly always had different mean fluorescent background values. In addition, most experiments experienced at least some global fluorescent drift as a result of the maturing of the strains and their adjustment to our HM9 minimal media.

In order to account for intra-box fluorescent background intensity differences and global fluorescence drift, we used the following per time point z-score:

$$\frac{\vec{x}_t - \bar{x}_t}{\sigma_{\vec{x}_t}} = \vec{x}'_t \tag{3.4}$$

where $\vec{x}_t$ is the vector of the background-corrected fluorescent signal for all occupied strain positions at time t; $\bar{x}_t$ is the mean value for the vector at the same time point; and $\sigma_{\vec{x}_t}$ is the standard deviation of $\vec{x}_t$.

Exponential smoothing

Since we ultimately wanted to use a first derivative-based feature, we first smooth the signal prior to calculating the first finite-difference. The smoothing prevents the amplification of noise in the differentiation step. To smooth, we implemented the Holt-Winters second-order exponentially weighted moving average (HWMA), which de-noises the data based on the history of both the trajectory and the history of the trajectorys derivative. We chose to use the HWMA over other smoothing methods because it is rear-facing only, which means that it smooths the current time point based only on the previous time points. This is as opposed to other methods, which require future time points to adequately smooth the current time point. HWMA, therefore, is uniquely suited to real-time sensing applications in which dramatic changes in the derivative of the trajectory can occur [68].

The HWMA is calculated at each time point as follows:

$$\begin{aligned}
\vec{x}_1'' &= \vec{x}_1' \\
\dot{\vec{x}}_1'' &= \vec{x}_1' - \vec{x}_0' \\
&\vdots \\
\vec{x}_t'' &= \alpha \vec{x}_t' + (1-\alpha)(\vec{x}_{t-1}'' + \dot{\vec{x}}_{t-1}'') \\
\dot{\vec{x}}_t'' &= \beta(\vec{x}_t'' - \vec{x}_{t-1}'') + (1-\beta)\dot{\vec{x}}_{t-1}''
\end{aligned} \tag{3.5}$$

where $\vec{x}_t''$ is the smoothed version of $\vec{x}_t'$, and $\dot{\vec{x}}_t''$ is that point's first-finite difference. Given that samples were taken at a consistent frequency, $\dot{\vec{x}}_t''$ then becomes an approximation for the derivative of the trajectory at time t. α and β are weights that were empirically chosen from the interval $[0,1]$ and were constant for all experiments. α can be thought of as the relative importance placed on the most recent values, as opposed to values from further in the past. Similarly, β is the relative importance placed upon the function's most recent approximation of the first derivative.

First finite difference

The penultimate feature-engineering step was to take the first finite difference. This goal can be accomplished by either directly calculating the first finite difference or by taking the recorded approximations of the first derivative from the output of the HWMA function and using them as the first finite difference. Both methods were tested and seemed to result in close approximations of each other. The basic method for calculating the first finite difference is explicitly listed below:

$$\dot{\vec{x}}_t = \vec{x}'_t - \vec{x}'_{t-1} \tag{3.6}$$

Intuitively, the first finite difference is an excellent candidate for any sort of machine learning model because it is what the human brain instinctively monitors when looking for changes in strain-promoter behavior [6]. Any significant modification in the mean or variance of the first derivative of a given promoter while induced or uninduced could signify that the promoter is sensitive to that particular modification of its environment. The engineering of a first derivative-approximation feature amounts to distilling out the pure changes in the original feature's behavior, while effectively filtering out any noisy changes that could be due to extraneous local or global environmental influences.

Vector normalization

Finally, the final step that was used in obtaining our finished features for machine learning was to normalize each vector by itself. Since most derivatives had the characteristic that, for the majority of time points t, they were $\dot{x}_{(t,\, s_i)} << 1$, normalizing each time point by the mean of itself had the effect of making the new, normalized vector's norm equal to 1. First derivative values that were close to 0 remained close to 0, while

$$\left(\frac{2175}{\sum_{i=0}^{2175} \dot{x}_{(t,\, s_i)}} \right) \cdot \dot{\vec{x}}_t = \dot{\vec{X}}_t \tag{3.7}$$

The reason this step was an important final member of our feature engineering pipeline is because of the relatively high-dimensionality of our feature vectors. High-dimensionality means

that our 2176-dimensional feature vector is affected by the Curse of Dimensionality [41]. Scaling the mean of the vector helps to increase the volume lying between different regions of the feature space. Our intention in doing this normalization is to increase the volumetric separations between uninteresting features that remain close to zero and features that occasionally make excursions away from the baseline.

3.2.3 Additional data preprocessing

In addition to the data preprocessing and feature engineering that were explicitly enumerate in the previous subsection, other steps were often taken when training and testing any machine learning algorithms. These additional steps all dropped some portion of the final feature set, but *only after* the features had been calculated using the entire original experiments. Dropping these time points before calculating the features would have introduced potential discontinuities to any features approximating the first-derivative.

Trimming grow-up phase and end phase time points from feature sets

All experiments included transient periods over the first several days of the experiment. These transients were caused by the colonies' recoveries post-spotting shock, their growth-to-effluence within their individual traps, and their second recovering following the switch to minimal HM9 growth media. In addition, since most experiments were run until the microfluidic chips were deemed unusable due to clogging by biofilms, the final hours of most experiments did not yield high-quality water data. Since the beginnings and the ends of the experiments represented non-steady state local and global growth conditions, the features from these periods were dropped prior to analysis and machine learning. This trimming was always done using the same built-in *dynomics.Experiment.trim_growth()* and *dynomics.Experiment.trim_tail()* methods, with the default being to trim up until eighteen hours before the first induction and then dropping all features occurring four hours after the final non-water induction.

Dropping bad, unwanted, and special inductions

Inductions of poor quality or that had imaging or user errors associated with them were flagged as unusable, as denoted in the *good_induction_flag* column of the experiment's *induction_record* table in the MySQL database that stored all of our experimental data. These inductions were subsequently dropped post-feature processing. In addition, inductions using undetectable metals, such as arsenic, mercury, and antimony, and special inductions that introduced any non-standard inducers were dropped from the feature set post-processing.

All features were processed and cached in permanent memory. Only the cached features were used for any further analysis.

3.2.4 Implementation of Lubansky data differentiation

During the comparison of different 2k chip microfluidic designs for variability reduction, we implemented a method for numerically deriving the first derivatives directly from the experimental data itself. This method, which was first described by Lubansky et al. in their 2006 paper, transforms the problem of calculating the smooth first derivative of noisy, potentially irregularly-spaced experiment data into numerically solving an integral equation of the first kind [72].

$$y^C(x) = \int_{x'=x_0}^{x} (x-x')f(x')dx' + y_0 + (x-x_0)r_0 \tag{3.8}$$

When discretized, this equation becomes

$$\mathbf{y}^C = \mathbf{Bf} + \mathbf{1}y_0 + (\mathbf{x}^M - \mathbf{1}x_0)r_0 \tag{3.9}$$

where "**B** is an $N_D \times N_K$ matrix of known numerical coefficients arising from the approximation of the integral in [Eq. 3.8] by numerical quadrature such as the trapezoidal or the Simpsons rule; 1 is a column vector with each element equal to 1" [72]. Here, N_D is the number of original data points and N_K is a discretizing integer.

The authors' solution accomplishes this task via Tikhonov regularization, using a general form of "leave-one-out" cross-validation to choose the regularization parameter λ [38]. For further details on the method, interested parties should refer to the original paper.

Among the many advantages to this method of calculating smoothed first derivatives is that places minimal arbitrary assumptions on the nature of the data. This lack of assumptions can be especially useful when dealing with scientific data, where the researcher wants to avoid biasing any results by imposing their beliefs on the analysis of the data.

To the author's best knowledge, this algorithm has never before been implemented in an open-source language. Therefore, the implementation of this algorithm in Python represents a novel contribution to the scientific computing community. The module is available on GitHub for general use [43].

3.2.5 Initial supervised learning with random forests

In our preliminary exploratory efforts, we used *scikit–learn*'s implementations of random forest classifiers [2]. This algorithm was chosen for its versatility, ease of implementation, and consistently competent classification performance [22].

Random forests are composed of ensembles of decision trees. Each decision tree, however, is constructed using a random subset of the training data; after the tree is constructed, the subset of training data is replaced. Additionally, during the construction of a tree, each split is chosen not from all the features in the subset of training data, as it would be with a normal decision tree, but with a randomly chosen subset of the features. Splits are typically decided using the Gini impurity index, which seeks to maximize the information gained at each branching.

As a consequence of the method of their construction, random forests represent an excellent realization of the concept of bootstrap aggregating, or bagging, to deal with the bias-variance trade-off [39]. While this construction strategy slightly increases the bias of the overall classifier, it greatly decreases the variance once the predictions of each individual member tree are combined (Fig. 3.6).

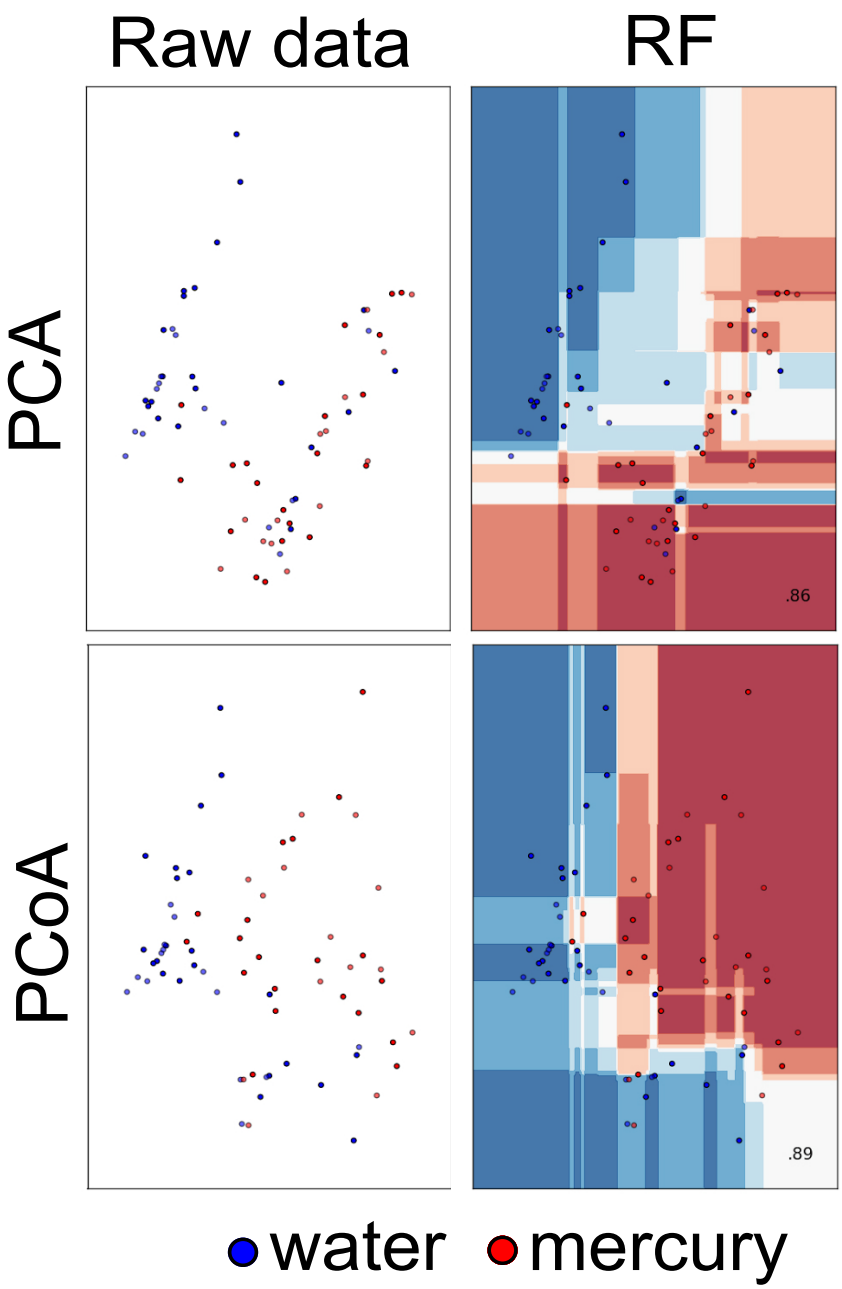


Figure 3.6: A visualization of the decicion boundaries of random forest classifiers classifying on specific sensor data. The above plots show the classification results produced by two random forest (RF) classifiers on reduced-dimension mercury and water induction data. This data was produced by our small biosensor with its engineered sensor strains. The raw data, which has been preprocessed via principal component analysis (PCA) and principal coordinate analysis (PCoA), is color-coded according to which class it represents. Blue dots represent time points where only water is present, while red dots represent time points where the strains are exposed to mercury. In the right-hand column, the prediction boundaries of the two associated RF classifiers can be seen. Areas colored blue indicate that the RFs predict any samples falling in those areas to be classified as water, while red indicates the prediction of mercury's presence. The varying intensities of each color indicates the strength of the prediction: dark indicates a high probability, while a light shading indicates much less certainty in those predictions.

3.2.6 Statistical distance distribution searches for the identification of arsenic- and mercury-sensitive promoters

Upon the failure of initial random forest learners to properly classify arsenic and mercury feature sets, a search was undertaken to identify any strains that were potentially sensitive to either of those metals. These searches were performed by first building distributions of statistical distances between a strain-position's baseline behavior in the time period before an arsenic or mercury induction and its subsequent behaviour during the induction. Then, the resulting distributions were then compared to the equivalent distributions from the control strains *lacZ* and *U139*.

While both the Wasserstein and Cramér distances were implemented as distance metrics, we eventually came to rely solely upon a version of the Cramér distance known as the *energy distance*. The energy distance, for two random vectors X and Y with cumulative distribution functions (CDF) F and G, respectively, is defined as:

$$\frac{1}{2}\sqrt{\int_{-\infty}^{\infty}(F(x)-G(x))^2\,dx} \tag{3.10}$$

Note that 3.10 is only true when $\int_{-\infty}^{\infty}F(x)dx = \int_{-\infty}^{\infty}G(x)dx$. In our case, since we are dealing with cumulative distribution functions, $\int_{-\infty}^{\infty}F(x)dx = 1$ for all CDFs F, which satisfies the preceding requirement. Also, note that this function is a true distance metric, whereby it satisfies the four required characteristics of such a metric: nonnegativity ($D^2(F,G) \geq 0$), symmetry, identity of indiscernibles, and that the triangle inequality holds for all relevant CDFs [117].

In our case, we defined two random variables W and M. These random variables represent measurable functions from the experimental outcome space to the ground truth label of water and our inducer-of-interest (usually a metal, hence the M), respectively. W and M have the cumulative distribution functions F and G, respectively, representing the strain-position's empirical behavior in the uninduced and induced states. The energy distance could then be calculated as

$$\frac{1}{2}D^2(F,G) = \frac{1}{2}\left(2\,\mathrm{E}\,\|W-M\| - \mathrm{E}\,\|W-W'\| - \mathrm{E}\,\|M-M'\|\right) \tag{3.11}$$

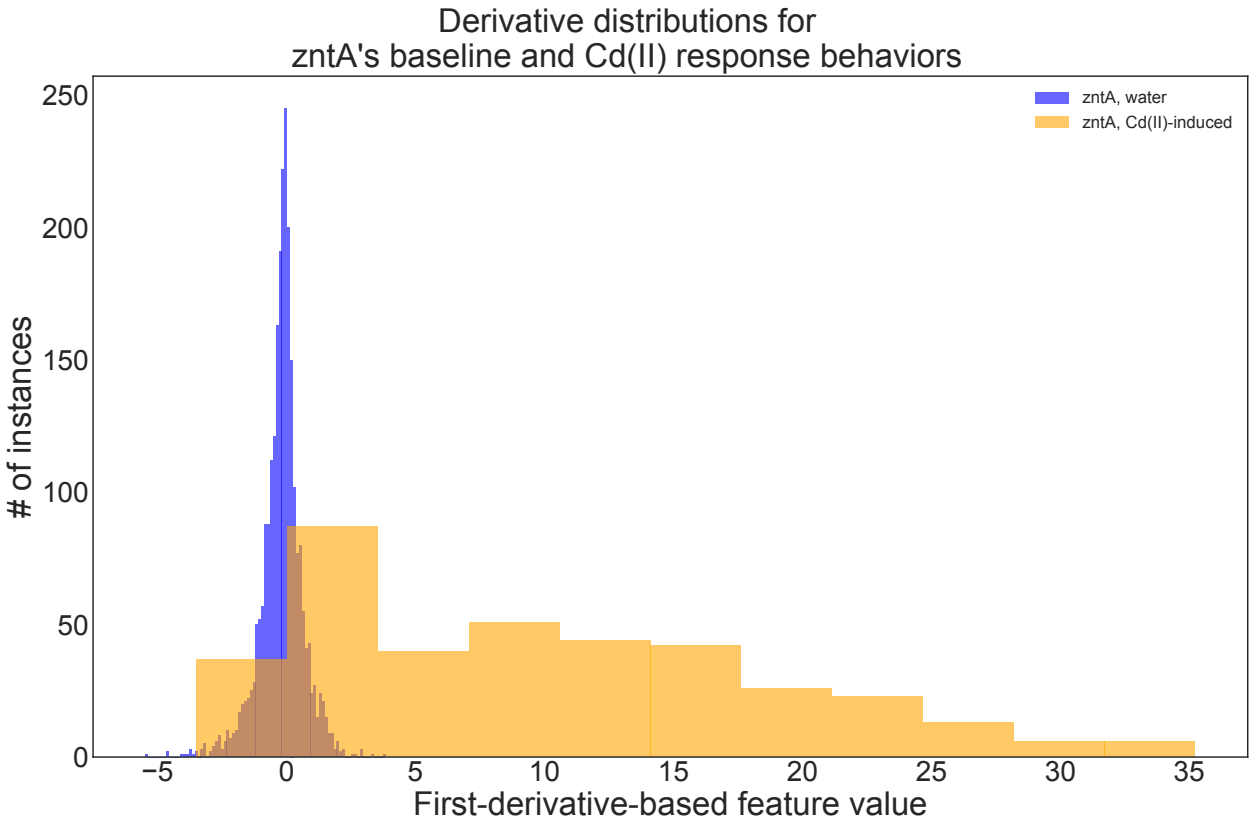


Figure 3.7: The response and baseline derivative distributions of a cadmium-sensitive strain. The two distributions displayed in this figure are the strain *zntA*'s induced and uninduced derivative distributions. The blue bins represent *zntA*'s behavior under normal growth conditions. The orange bins represent the strain's behavior during exposure to cadmium. By measuring the energy distances between many of these distributions, we can build a distribution of energy distances to compare with the analogous energy distances of the control strains.

These distances were calculated for each strain-position with only itself for each single induction-of-interest (Fig. 3.7). That distance then was a member of the set of statistical distances for that strain-position. The reason for performing this operation on an induction-by-induction basis was that, even for a single strain, collating all of that strain-positions behavior data for all inductions-of-interest across all experiments could result in a meaningless distance. Since baseline behaviors and induction behaviors could differ from experiment-to-experiment and even from induction-to-induction within the same experiment, it was important that we compare only the strain-position's induction behavior with the baseline behavior that immediately preceded the induction (Fig. 3.8).

Once calculated and plotted, the energy distance distributions for every strain-position on chip

were manually inspected and compared with the control strains' aggregated distance distributions (the control strains being the aforementioned *U*139 and *lacZ*). This procedure was done for all arsenic inductions and all mercury induction, with both energy and Wasserstein metrics; in total, 7,980 distance distribution comparison plots were inspected.

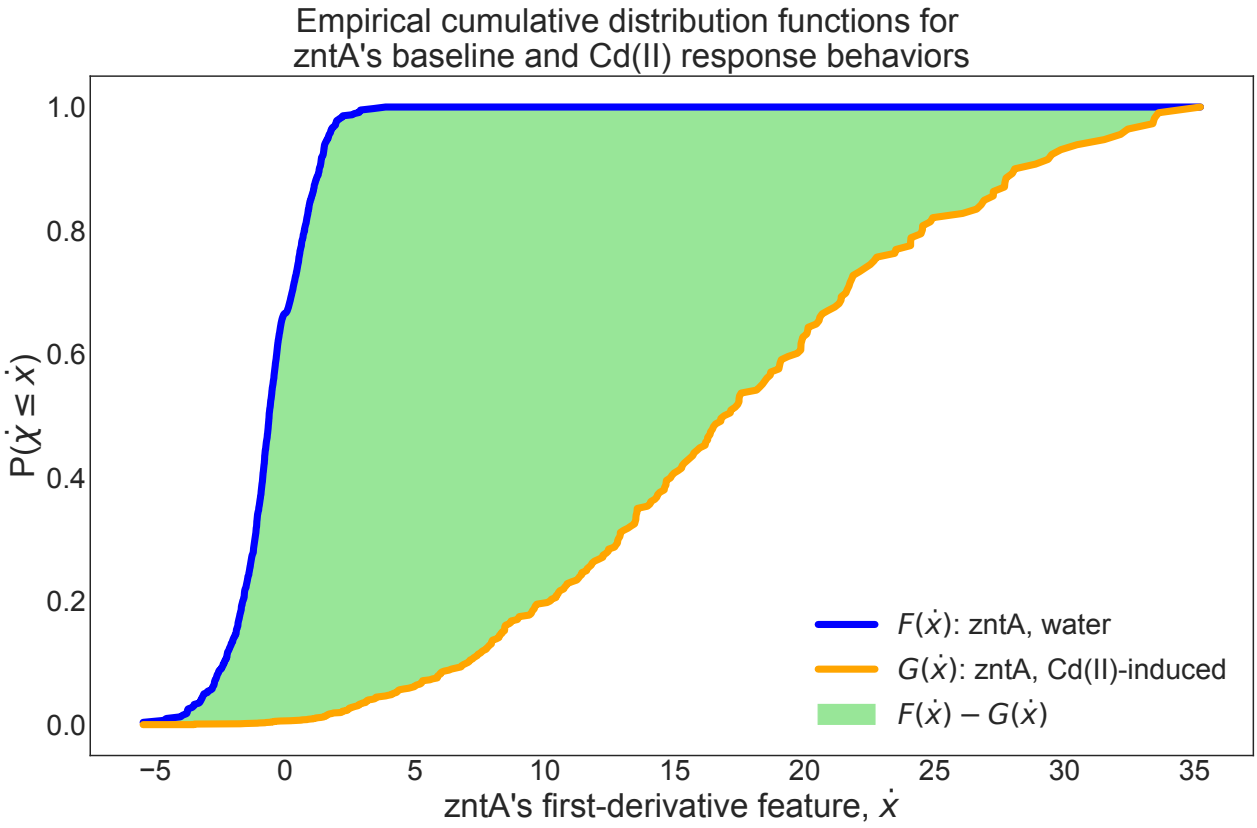


Figure 3.8: Cumulative distribution functions (CDFs) of induced and uninduced strain behaviors. The two CDFs displayed above are formed from the strain *zntA*'s features during the pre-induction and intra-induction time periods. Here, the energy distance is calculated as a function of the distances between induced (orange) and uninduced (blue) curves; those distances are represented by the region shaded green.

3.2.7 Leave-N-out cross-validation

During any and all classifier hyperparameter fitting or performance generalization evaluations, *leave-n-out cross-validation* was used to prevent overfitting and ensure the generalizability of the results [25].

Cross-validation (CV) is a routinely-used method in machine learning to estimate how well a classifier will perform when presented with data from outside of its training set [64]. During a single round of cross-validation, a subset of the data, known as the *test set*, is removed from the data set and held in reserve. The remaining portion of the data, now known as the *training set*, is then used to train the model. After training is complete, the new model is then run on the *test set* and the results are scored using a predeterimined evaluation metric, such as accuracy, precision, recall, AUROC, etc. Typically, many rounds of cross-validation is performed, with the mean of their scores calculated afterwards. This gives a more realistic accounting of how such a classifier would generalize to data outside of the data set in hand [106]. Cross-validation would be largely useless, however, if each round used the same training and testing sets. Thus, there are multiple methods by which to generate distinct training and test sets, the most popular of which is *k-fold cross-validation*. In *k-fold cross-validation*, the samples are randomly shuffled and then sorted into k subsets $\left\{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, ..., \mathbf{s}_k\right\}$. In the first CV round, $\mathbf{s}_1$ is designated to be the test set and training then proceeds on the remaining $k-1$ sets. In the second round, $\mathbf{s}_2$ is selected as the test set and training proceeds on the remaining sets. This process continues until every set $\mathbf{s}_i$ has served as the training set. When the process is complete, the k scores are then averaged together and taken to represent the expected value of the classifier's score on unseen data (Fig. 3.9).

k-fold cross-validation, however, relies upon the assumption that each individual sample is a realization of an independent and identically distributed random variable. In the case of samples drawn from time series data, this assumption no longer holds, since each timepoint is dependent upon the time points that have preceded it.

In case of time series data, then, it can be necessary to use *leave-n-out cross-validation*. In *leave-n-out cross-validation*, the test set is selected to be the minimally-independent unit; in the case of this project, that minimally-independent unit was one experiment, since the time points of one experiment are independent from the time points of another experiment.

Throughout this project, *leave-n-out cross-validation* in the case where $n = 1$ indicates that a single experiment would serve as the test set, while the remaining experiments then served as the

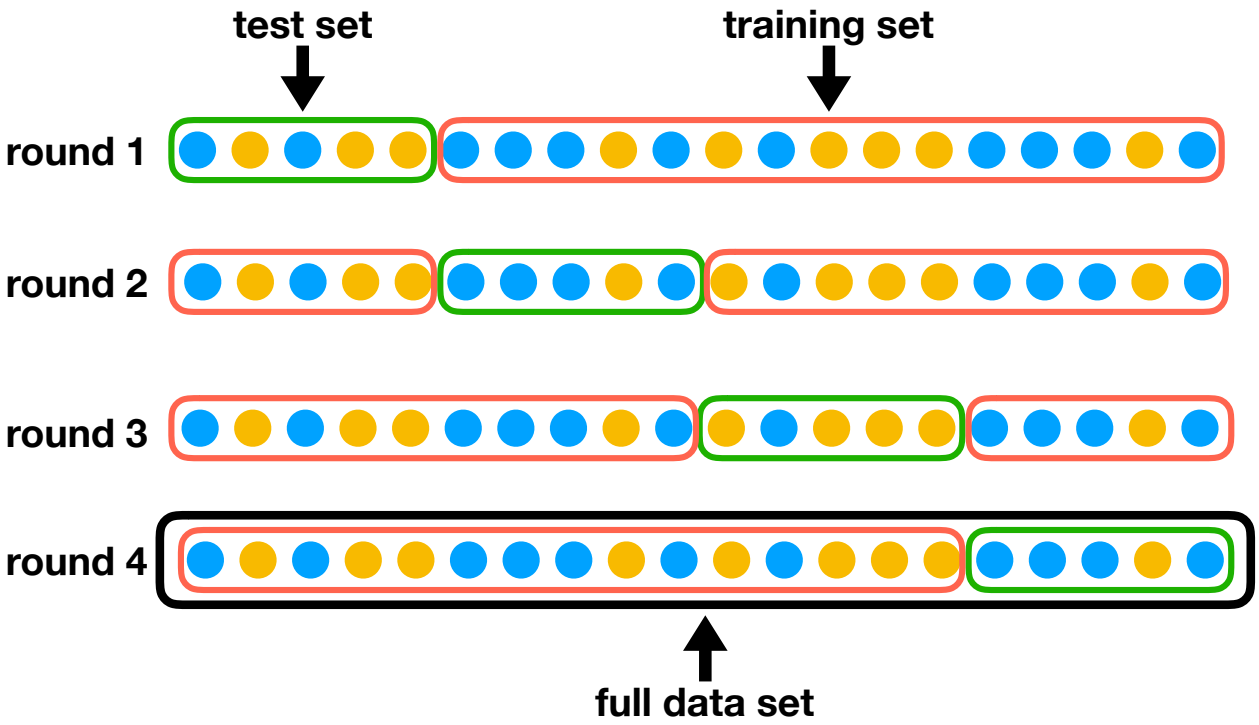


Figure 3.9: A depiction of four-fold cross-validation. The example above represents four-fold cross-validation on a data set containing twenty samples. Prior to starting, the data set is shuffled to randomize the order of the samples (not shown). Then, four subsets of five samples each are selected. These are the test sets and are highlighted by a green box. During each cross-validation round, the test set is held in reserve and the remainder of the data, known as the training set and highlighted by red boxes, is used to train the classifier. The classifier's performance is then checked on the test set before proceeding to the next round of cross-validation.

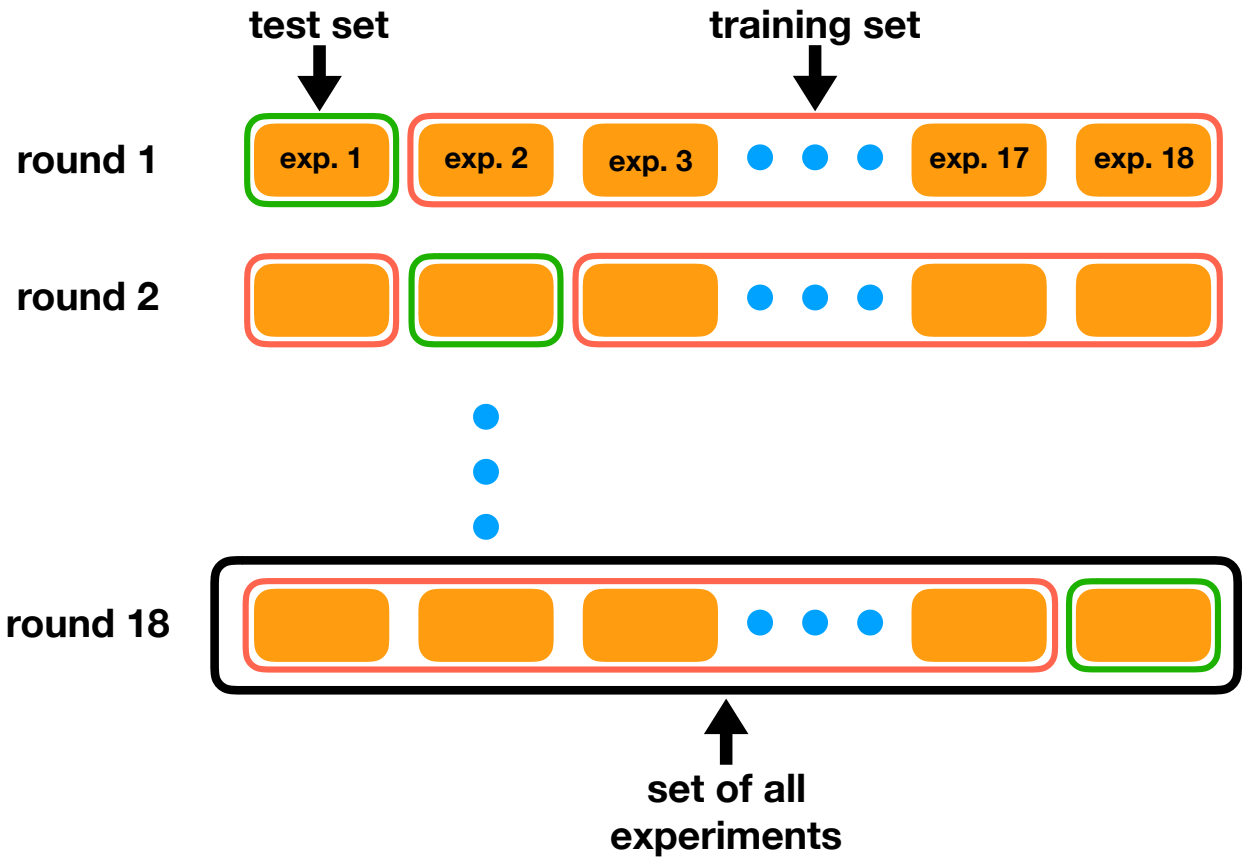


Figure 3.10: A depiction of leave-one-out cross-validation. In the course of this project, we heavy utilized leave-one-out cross-validation to prevent overfitting and to predict classifier generalizability. In a single CV round, all data pertaining to a single experiment would be designated as the test set, which are highlighted by the green boxes. The data that pertained to the remaining experiments were then designated to be the teset set. Training and testing with the classifier would proceed and then a new experiment's data would be designated as the next test set. In this way, we prevented the leak of information between training and test sets, which is essentially in this context if we were to accurately estimate how well our classifiers would perform on never-before-seen data.

training set. In the case of leave-1-out cross-validation, we would perform eighteen total rounds, since there were eighteen experiments in our standardized training set.

For all hyperparameter searches, we used *leave-n-out cross-validation* with $n > 1$. In these cases, the order of the experiments in the experiment set were first randomly shuffled and then divided into $\frac{18}{n} = k$ distinct subsets. k rounds of CV were then performed in order to evaluate the generalizability of the hyperparameter set (Fig. 3.10).

3.2.8 Classifier evaluation metrics for imbalanced datasets

The problem of imbalanced classes: ie, why accuracy is a poor performance metric

As noted in the previous, there are many separate evaluation metrics for determining how well or poorly a classifier performs on a given data set. In the general case, when a data set is composed of roughly equal parts of different classes, most evaluation metrics will give an easily interpretable account of classifier performance. However, in the case of a data set with large class imbalances, these metrics become more difficult to interpret.

For instance, in the case of an extreme imbalance, suppose a data set were composed of 9,999 examples of a class A and just one example of class B. In that case, a dummy classifier that simply labels any sample passed to it as class A will have an accuracy score of 99%. Conversely, a learner that always correctly classifies the single class B sample and occasionally misclassifies samples from the set of Class A would be less accurate than the dummy classifier.

This project, by necessity, is one with large class imbalances. Since our on-chip cells require up to twelve hours to recover from exposure to heavy metals, our between 85.2% and 88.9% of the samples in our data sets were labeled as H_2O (Fig. 3.11). The specific percentage varied depending upon which set of metals desired to predict and would consequently include in the classification data sets for training and testing.

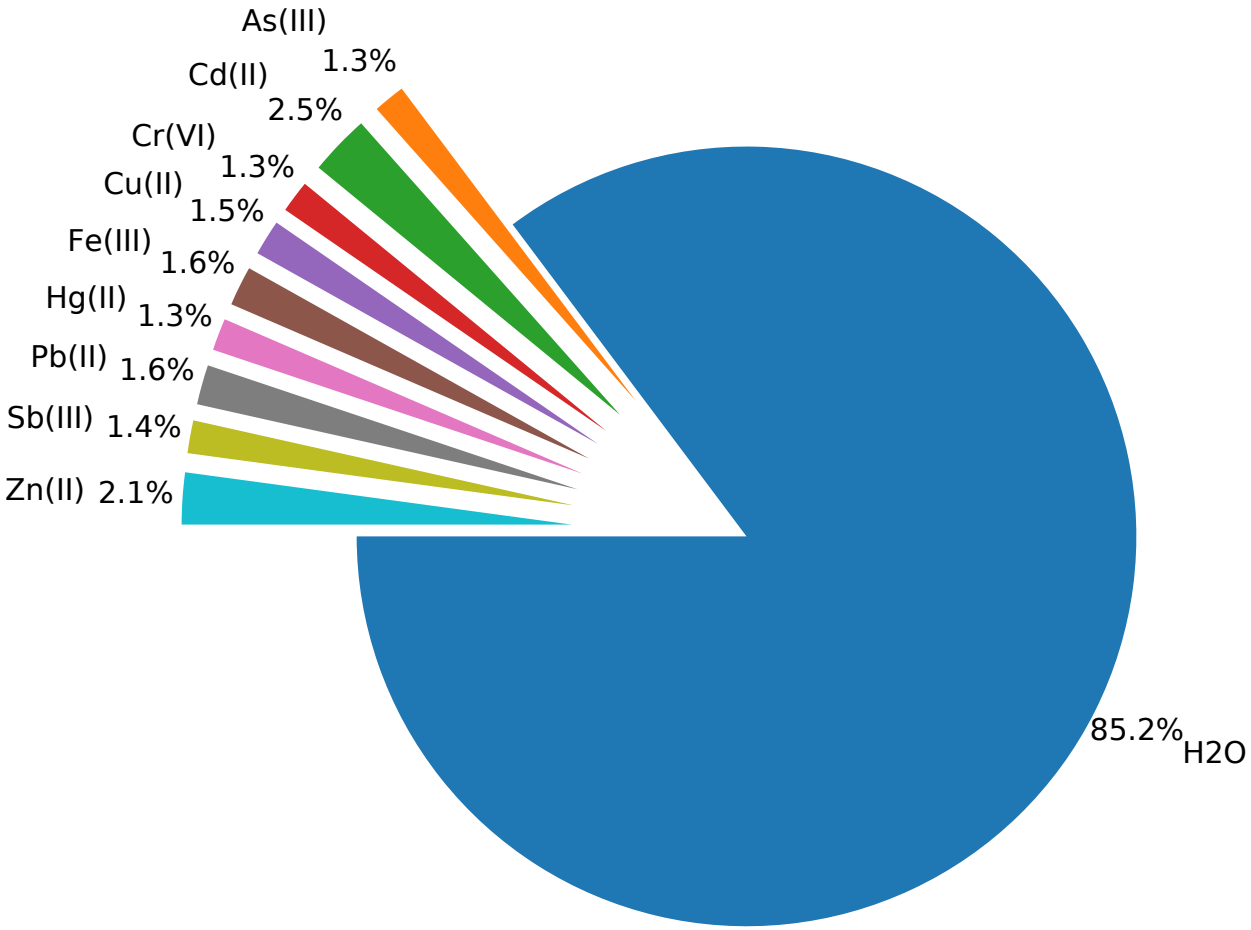


Figure 3.11: The imbalanced class composition of our data set. This pie chart represents the composition of our data set by class. While 85.2% of the data set is composed of pure water, the remaining 14.8% is composed of a set of nine metals: lead, antimony, zinc, arsenic, cadmium, chromium, copper, iron and mercury. Of these metals, cadmium is the most frequent class, comprising 2.5% of the total data set.

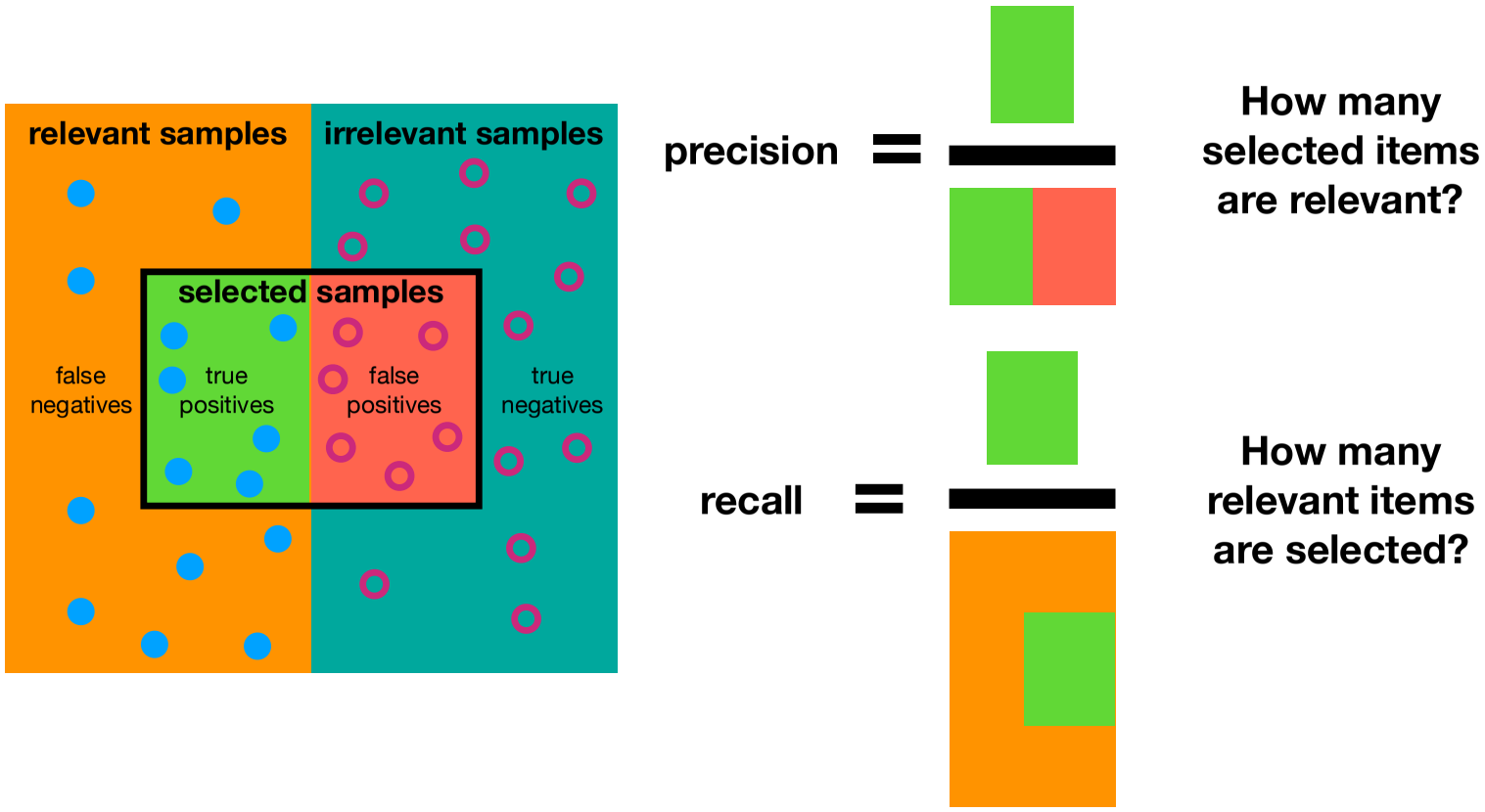


Figure 3.12: An abstracted visualization of precision and recall as classifier evaluation metrics. Precision and recall are the two metrics from which the F_1 score is computed. In the case of a multiclass problem, the relevant samples would be those samples whose class is currently being presented to the classifier, while all other classes' samples would be considered irrelevant. In this sense, the current class is relevant because that is the class that we would like our model to predict, since we are currently presenting it with a member of that class. Precision, then, is the ratio of true positives to the total number of samples predicted to be the current class. Recall is the ratio of true positives to the total number of samples of the current class present in the full data set. This inspiration for this image originally came from an excellent illustration on *Wikipedia*'s "Precision and recall" article.

Multiclass F_β function as an evaluation metric

As a result of the severe class imbalances within our data set, we used a performance measure that works well in an imbalanced setting: the F_1 score. The F_1 score is the harmonic mean of the classifier's precision and recall (Fig. 3.12). Precision, for a classifier and a given class, is defined as the number of true positives divided by the total number of samples that are classified as the class-of-interest. In the following definitions, TP indicates *true positives*, FP means *false postitves*, and FN means *false negatives*.

$$Pr := \frac{TP}{TP + FP} \tag{3.12}$$

Recall, denoted as *Re* below, is the number of true positives divided by the total number of samples of the given class present in the data set.

$$Re := \frac{TP}{TP + FN} \tag{3.13}$$

For imbalanced classes, the F_1 metric is a logical choice, since evaluating a classifier with it forces the classifier to seek out members of the minority class while not becoming overzealous and classifying members of the other class as the rarer one. The F_1 score is a member of the F_β metric family, which is defined as

$$F_\beta = \left(\frac{Re^{-\beta} + Pr^{-\beta}}{2} \right)^{-\beta} \tag{3.14}$$

In the case where $\beta = 1$, the metric becomes

$$F_1 = 2 \cdot \frac{Pr \cdot Re}{Pr + Re} \tag{3.15}$$

In the case of multiclass problems, there are two frequently-used variants of the F_1 score: the F_1-macro and the F_1-micro scores. The F_1-macro measure is calculated by calculating the F_1 score for each individual class separately and then taking their mean. The F_1-micro score is calculated by

taking the total number of true positives, false positives and false negatives post-classification and then calculating the total F_1 score directly from them [104].

Each version has their advantages and disadvantages. The F_1-macro, denoted above as F_1^M, treats each class' score with equal weight, thereby preferring classifiers that perform better on rarer classes in data sets with large class imbalances. The F_1-micro score, denoted by F_1^μ, on the other hand, rewards better all-around, per-instance performance; this quality rewards classifiers that treat all classes equally, which is preferable in a setting where classes are equally balanced. In the case where a rarer class is exceedingly difficult to classify correctly, the F_1^μ will incentivize the perverse classification strategy of always incorrectly classifying that rare, difficult-to-detect class as the majority class, as a consequence of how it is calculated [71].

As a result of the nature of our data set, the F_1^M metric was used in all hyperparameter searches.

3.2.9 Supervised learning with extreme gradient boosted trees (XGBoost)

Extreme gradient boosted trees (XGBoost) are a tree ensemble method invented by Tianqi Chen in 2014. As opposed to random forests, which build their ensembles using bootstrap aggregating and additional randomization strategies, XGBoost builds its tree ensembles in an additive manner. Instead of each tree being built in isolation from the others, as with random forests, XGBoost builds the next tree in the ensemble by finding the tree that optimizes the current ensemble's objective function, when added to that ensemble [111]. This strategy can be thought of as building the next tree such that it does its best to minimize the existing model's error, while respecting any regularization requirements that are present (Fig. 3.13.

XGBoost has become extremely popular over the last five years and has been used to win several notable machine learning contests (see Kaggle's contest results, for instance). We chose it as the next type of learner to train because of its abilities to deal proficiently with high-dimensional, nonlinear data sets such as ours.

All XGBoost functions were implemented in Python. All hyperparameter searches and

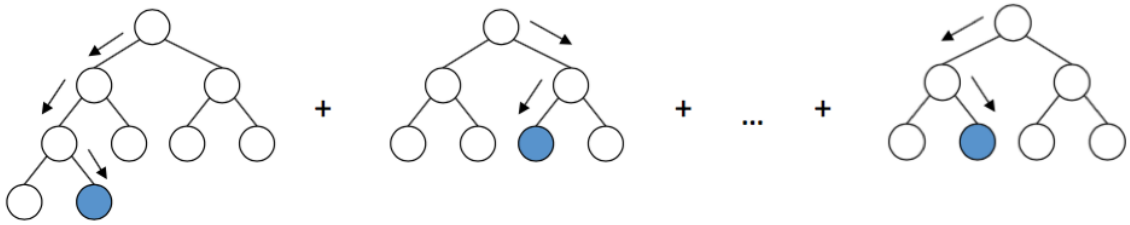


Figure 3.13: A conceptual depiction of an XGBoost learner. In this XGBoost classifier, the next tree in the ensemble is trained in such a way as to try and eliminate the error of the previous trees. This strategy sets it apart from random forests, in which every tree is trained independently [91].

cross-validation rounds were parallelized and run on the multicore in-lab server. Hyperparameter optimization took place both by hand and with a Bayesian optimizer. Implementation of the Bayesian optimizer will be discussed in the next section.

Hand-tuning of XGBoost learners

We optimized the hyperparameters of the initial XGBoost learners by hand based on recommendations by several data scientists [57]. The general process, which yielded competent learners, is enumerated below. At each step, we used early stopping to reduce over-fitting. In addition, at each step we used leave-one-out cross-validation and then averaged the results after each full cross-validation round to produce the final score for a specific parameter set.

1. We optimized the number of trees for a learning rate of 0.1;

2. Then, we used a grid search to optimize the maximum tree depth and the minimum child weight over reasonable ranges of both parameters. If any values at an extreme limit of a range were selected, the grid-search was repeated over a appropriately-shifted ranges of both parameters;

3. Grid search was then used to find the optimal value for the regularization parameter γ;

4. The number of trees in the ensemble was then re-tuned with early-stopping;

5. We performed a grid search to optimize the subsample and column-sample-by-tree parameters;

6. The final classifier parameter set was then cross-validated over the entire experiment set to produce the final evaluation metric score and multiclass confusion matrix.

In machine learning, the hand-tuning of hyperparameters is typically the realm of experts and is specific to the type of learning algorithm and intrinsic characteristics of the data set in question. It should be noted that such hyperparameter tuning can be useful when the researcher is seeking to rapidly explore a classifier space. However, it is inherently inferior to other search methods, such as random search and Bayesian optimization, the latter of which will be discussed in the following section.

3.2.10 Bayesian optimization of classifier hyperparameters

Training single machine learners involves the fitting of many low-level parameters, which are unique to the training data in question. Examples of these parameters are the weights for a logistic regressor or the splitting criteria for the branches of a single decision tree. For today's supervised machine learning algorithms, these low-level parameters always have associated with them a (usually convex) objective function to be optimized [16]. Thus, the problem of training a single machine learner usually reduces to searching for the global minimum of the objective function, which can be done via numerical approximations of the gradient of the objective function [93]. Examples of these optimization methods are ordinary gradient descent, used for logistic regression learners, subgradient descent, used for support vector machines, and stochastic gradient descent, which is used to train artificial neural networks.

Machine learning algorithms, however, usually have several or, sometimes, many additional parameters that must be set prior to the training process [30]. The optimization of these high-level parameters, known as *hyperparameters*, is one of, if not the, single most important steps to producing a maximally-functioning machine learner for a given data set [88]. However, unlike the low-level parameters learned during the training process, these high-level parameters rarely have a clear

objective function with a computable landscape associated with them. This fact makes learning the optimal hyperparameters for any supervised learning problem is usually an impossible task.

Traditionally, in order to find an optional set of hyperparameters, researchers have resorted to exhaustive searches. Typically, these take the form of a grid search, whereby candidate learners are constructed using n-dimensional tuples of candidate hyperparameters from an n-dimensional grid [54]. Each candidates' performance is then estimated by fitting them using cross-validation. This approach is sometimes viable for low-dimensional hyperparameter spaces; for instance, finding the optimal values of C and γ for a support vector machine can usually be accomplished efficiently with a traditional grid search.

However, grid search, being a form of exhaustive search, is an inefficient and, therefore, time consuming method. As a result of these limitations, grid search rarely yields the optimal hyperparameter combinations for higher-dimensional spaces [17]. This problem is yet another example of the "curse of dimensionality". In 2012, Bergstra and Bengio showed both empirically and theoretically that a random search of the hyperparameter space was more efficient than either hand-tuning the hyperparameters or optimizing them through a grid search [17]. Since then, however, sequential model-based optimization (SMBO) methods have come to the forefront of hyperparameter optimization, of which Bayesian optimization has become the most widely regarded [56]. Unlike grid search, which, upon completion of a function evaluation, simply moves to the next hyperparameter tuple in its queue, SMBO methods utilize the history of their evaluations to inform their next choice of hyperparameters.

Out of SMBO methods, Bayesian optimization, has become widely regarded as the most efficient search strategy for optimizing hyperparameters [103]. Using Bayesian statistics to find solutions to costly problems, whether these problems cost time or money, had existed since the 1970s [82] [83]. Its application to machine learning since 2012 has contributed to significant advances in the field.

In any optimization problem, we seek a global minimizer $\mathbf{x_{min}} \in X$ some objective function f:

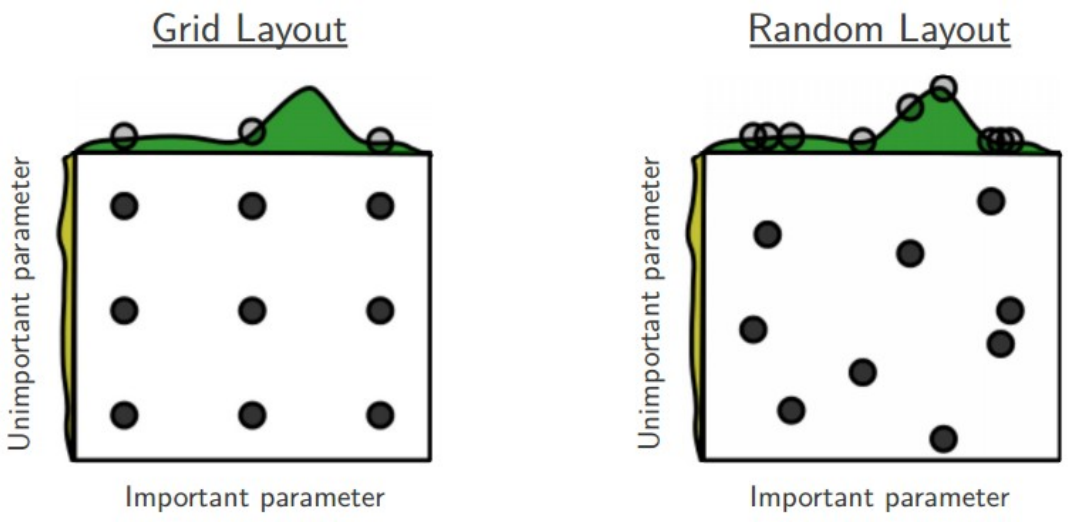


Figure 3.14: Grid search versus random search strategies for classifier hyperparameter optimization. This figure depicts why random search is more efficient than a straightforward grid search. The space of hyperparameters is large and only a small fraction of the space actually results in higher evaluation scores for the classifiers, as represented by the green curve on the top of each subfigure. While a grid search spends most of its time in low-scoring areas of the space, a random search is more likely to efficiently stumble upon a high-scoring region of the space [59].

$$\mathbf{x_{min}} = \arg\min_{\mathbf{x} \in X} f(\mathbf{x}) \tag{3.16}$$

In our case, X is our space of candidate hyperparameters and f is an evaluation metric of our choosing. In the course of this project, the $f = F_1 - macro$ most frequently, for the reasons discussed previously.

In order to most efficiently direct the search for $\mathbf{x_{min}}$, any Bayesian optimization algorithm forms what is known as a *surrogate model* to approximate f. In addition to a surrogate model, the Bayesian optimizer also uses an *acquisition function* $\alpha_n : X \to \mathbb{R}$ that directs exploration of the hyperparameter space by recommending the next set of hyperparameters $\mathbf{x}_{n+1}$ for evaluation. To find $\mathbf{x}_{n+1}$, α_n uses the surrogate model's *prior probability distribution*. Upon calculating the new output $f(\mathbf{x}_{n+1}) = y_{n+1}$, the optimizer incorporates $\left(\mathbf{x}_{n+1}, y_{n+1}\right)$ into the search history $\mathcal{D}_n$. Finally, it updates the surrogate model's probability distribution using the *posterior probability distribution*, which is calculated using $\mathcal{D}_{n+1}$ (Fig. 3.15). This process is repeated until a stopping criteria has been met [66] [33] [40].

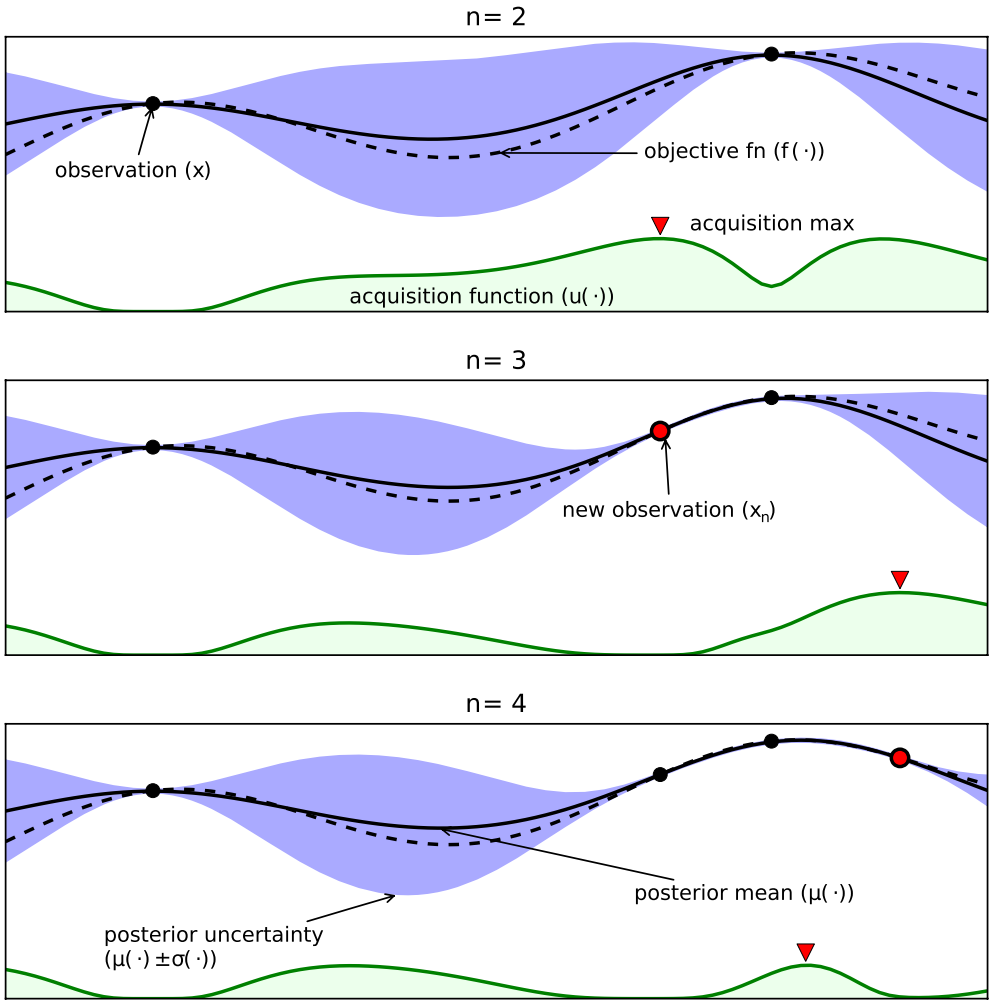


Figure 3.15: The learning process of a Bayesian optimizer. These three panels represent three sequential learning steps taken by a Bayesian optimizer as it searches for optimal hyperparameter value. The dashed line is the actual form of the objective function, which is unknown to the optimizer, but which the optimizer seeks to learn. The solid black line is the posterior mean of the optimizer; ie, it is the optimizer's model of the objective function. The blue regions are the confidence bounds associated with the posterior distribution. As new observations are added, the optimizer's acquisition function alternates between exploration and exploitation to try and learn the location of the best hyperparameter values. After each observation, the optimizer updates its posterior distribution with the new information. The posterior distribution then becomes the prior distribution for the next step in the sequence [94].

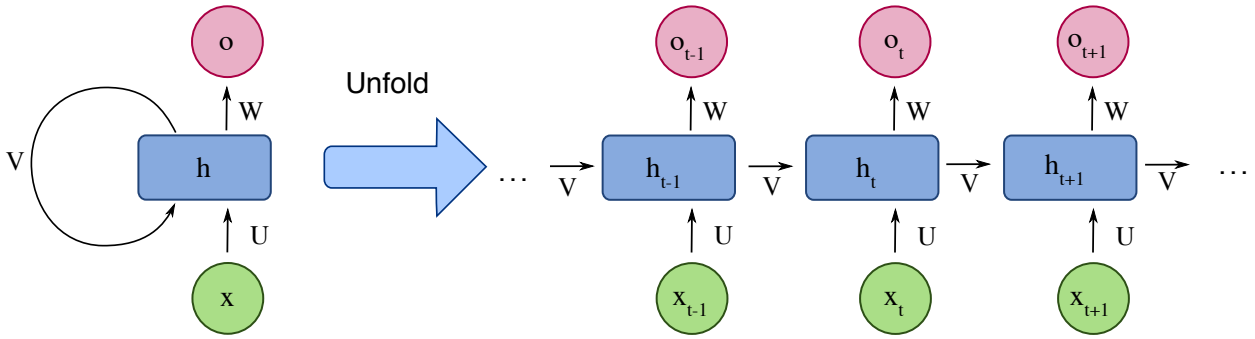


Figure 3.16: A generic recurrent neural network (RNN). This figure demonstrates how a recurrent neural network is unfolded through time to form a deep neural network. An RNN's node's built-in memory unit allows that node to essentially be a single deep learning network. By leveraging information about prior inputs, RNNs can learn patterns that are far too complex for other machine learning algorithms [80].

For both XGBoost and the deep neural network classifiers implemented in the course of this dissertation, the popular Bayesian optimization library *hyperopt* was used to conduct the searches [18]. A general search harness was created and used for both the XGBoost and deep learning classifiers. In both cases, a search was conducted using randomized leave-three-out cross-validation over a total of 1000 hyperparameter candidate sets.

3.2.11 Supervised deep learning with long short-term memory recursive neural networks

Given the inherent temporal nature of our data, classification algorithms that can utilize sequence context to augment classification performance would be especially advantageous. Long short-term memory recurrent neural networks (LSTM), a variety of deep neural network, are widely recognized to be the best performer for context-dependent problems and are heavily employed in speech recognition, dictation, text interpretation and generation, and the prediction and classification of time series data [67]. Famous examples of LSTM-RNN-based learners are Google DeepMind's AlphaGo and AlphaStar, both of which outperformed the top human experts in the board game *go* and the real-time strategy video-game *Starcraft II* [99, 13]. Both of these problems were ground breaking at the time and relied heavily on LSTM-RNN deep learners.

Recurrent neural networks, the family of deep neural networks to which LSTM-RNNs belong, are a "class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence" (Fig. 3.16) [80]. Long short-term memory networks were invented in 1997 to overcome the problem of exploding and vanishing errors that plagued recursive neural networks during training [53]. Nodes in a feed-forward neural network (FFNN) are composed of an activation function that uses the product of weights and the node's inputs to compute the node's output. As opposed to the nodes of an FFNN, nodes in an LSTM-RNN have four interacting interior layers:

1. *cell state*: the *cell state* is the node's memory. It is a vector that holds information between inputs and is updated by a combination of information from the *forget gate* and the *input gate*.

2. *forget gate*: the *forget gate* is the layer of the node that decides, based on its weights, the node's last output, and the node's current input, which portions of the current *cell state* to set to 0 (ie, to forget).

3. *input gate*: the *input gate* uses the last output and the current input to decide which components of the *cell state* to update.

4. *output gate*: the *output gate* combines the last output, current input, and updated cell state to decide what the node should output to the next layer.

To build our deep learning network, we used the Python APIs of the popular libraries *tensorflow* and *keras* [7, 29]. A general training harness was constructed and passed to our Bayesian optimization harness, which then performed the hyperparameter and network architecture search over 1000 search instances. The entire optimization search took approximate ten days. The result was a stateless LSTM-RNN (ie, the LSTM-RNN would reset the cell state to 0 between inputs) with a 80-node LSTM-RNN layer, followed by a 763-node feed-forward layer, and finally a 7-node output layer utilizing a softmax activation function. Regularization strategies were utilized during training to avoid overfitting.

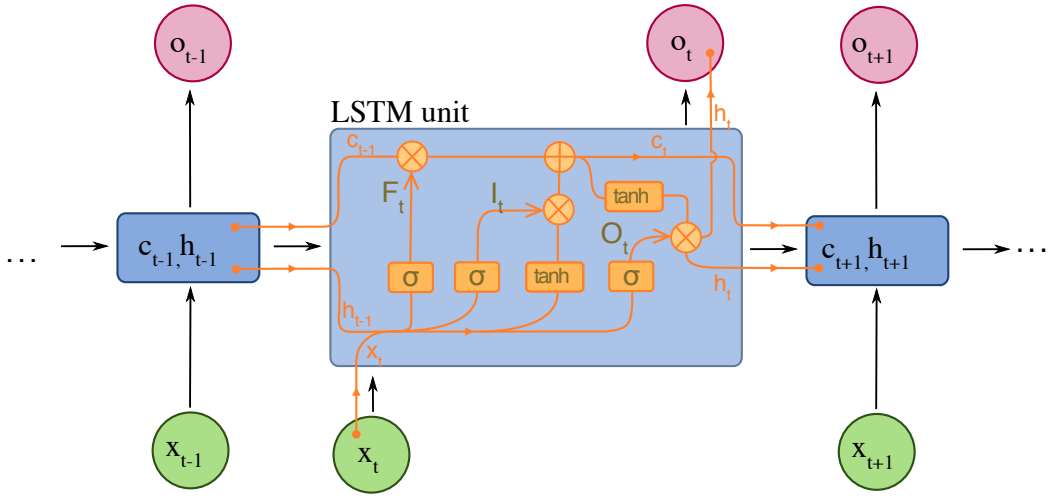


Figure 3.17: A long short-term memory recurrent neural network (LSTM-RNN) node. This diagram represents a single LSTM-RNN node, where c_t is its cell state, F_t is the forget gate, I_t is the input gate, O_t is the output gate, o_t is the output, and h_t is the hidden state (equivalent to the output, but stored for the next calculation). Joining flow lines indicate concatenation. Orange circles indicate pointwise addition or multiplication. σ indicates a sigmoid activation function and *tanh* indicates a hyperbolic tangent activation function. In this diagram, the previous time point, $t-1$, informs the node's current output via the operations of the forget, input, and output gates upon the input, cell state, and hidden state. For an excellent introduction to LSTM-RNNs, see [84]. [80].

Cloud hardware and server

All training of LSTM-RNN learners, including the Bayesian hyperparameter optimization search, was performed on a cloud-based Paperspace virtual Linux Ubuntu server with 8 CPUs, 30 GB RAM, 250 GB of storage space, and an NVIDIA Tesla V100 GPU, which itself had 16 GB RAM. All backpropagation steps were executed on the GPU.

Feature creation, pre-caching, and data generator implementation

In order to avoid additional computational and, more important, memory load during training, all features were generated and cached on-disc before training began. Since the LSTM-RNN was stateless (ie, reset its cell state between discrete inputs), the input features were $(num\_timesteps) \times (num\_strain\_positions)$ matrices, with an associated class label that was the label of the middle time point [19]. Thus, each input can be thought of as time window, where each row represents a

single time point of an experiment and each column is the time series of length *num_timesteps* of a specific strain-position (Fig. 3.18). For this project, the length *num_timesteps* was always odd; therefore, there was always an equal number of time points before and after the middle point in each window. As a result, each time window could be thought of as representing that middle point and was accordingly labeled with the point's class. Since the number of time points to be used as an input was also a hyperparameter in need of optimization, a range of time windows were generated and cached in permanent memory. We then implemented a range of custom Python generator classes and incorporated into the training harnesses [1, 114]. In any training or prediction call, a new instance of the generators would be passed to a model and would load the pre-cached features and labels on-the-fly [12]. At the end of these calls, the data would be dumped from memory, thereby preventing out-of-memory errors.

Standardized experiments

The Bayesian hyperparameter optimization and subsequent LSTM-RNN training were performed solely on data from the 18 standardized experiments. No data from either the urban water samples nor the San Juan River samples were included in either case. The only metals used in either case were cadmium, lead, copper, chromium, iron, and zinc.

Classification of urban water samples

All predictions on urban water sample experiments were made using a multiclass, single-label LSTM-RNN that had been fit on the eighteen standardized experiments. Its hyperparameters were the optimal set identified by the Bayesian optimization search, with the softmax activation function being retained in the final layer. The softmax was retained as the activation function because it enables multiclass, single-label classification:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},\ z = (z_1, \ldots, z_K) \in^K \tag{3.17}$$

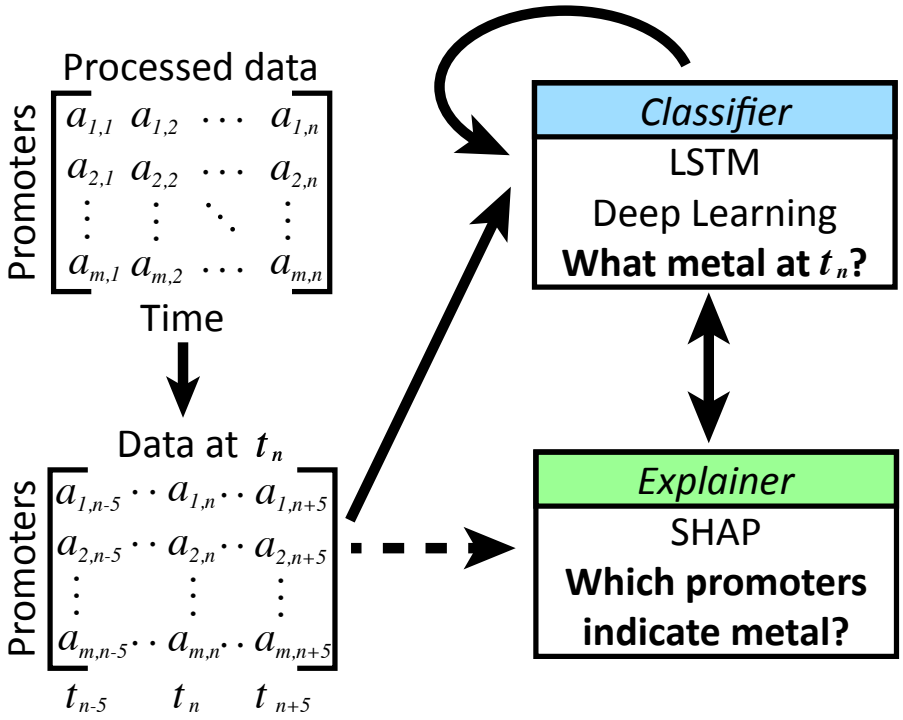


Figure 3.18: LSTM-RNN feature engineering and explainable AI (XAI) workflow. The feature engineering via time-windowing is clearly visible here, where a single column of the "Processed data" array becomes its own array, centered around t_n. In this case, just as with our actual LSTM, the time window includes five time points before and after t_n. Those features are then used to train our deep learners and, in addition, to help train our XAI learner, which we refer to as the "Explainer".

In Eqn. 3.17, K is the number of classes [42]. This function forces all values to nearly equal 0 or to nearly equal 1, while requiring that the entire function sum to unity. These properties force the softmax function to effective predict a single class at each output.

Eighteen rounds of training and predictions were performed using early stopping to prevent overfitting, with each of the eighteen standardized experiments serving once as the early stopping routine's test set. Predictions were then averaged.

Classification of San Juan River samples

Since the San Juan River water samples represented a multiclass, multilabel classification task, the LSTM-RNN's output layer's activation function was changed from the softmax function to the sigmoid activation function. Prediction tests were then performed in the same manner as for the urban water samples.

Unlike the softmax function (Eqn. 3.17), the sigmoid activation function's outputs are independent of one another (ie, not constrained to sum to unity):

$$\sigma(\mathbf{z}) = \frac{1}{1+e^{-\mathbf{z}}} \tag{3.18}$$

Since they are constrained to the interval $(0,1)$, they make an excellent choice in a multiclass, multilabel classification problem where probabilities of presence can sum to more than 1 [48].

3.2.12 Explainable artificial intelligence (XAI) with Shapley additive explanation values

Over the past two decades, our abilities to introspect and interpret complex machine learning algorithms have become more and more obfuscated as the same algorithms have grown ever more powerful [63]. The causes of complaints and warnings surrounding our inability to explain the basic reasoning of even moderately complex AI algorithms are more than mere aesthetics; in the areas of medicine, law enforcement, criminal justice, and warfare, the ability to understand why an AI is

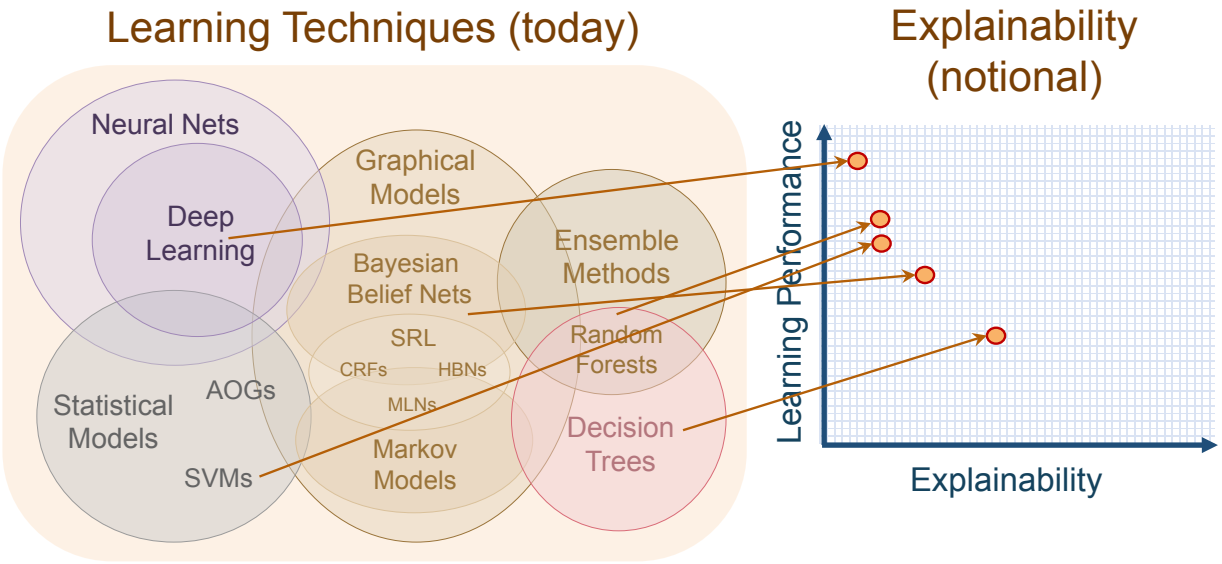


Figure 3.19: A conceptual depiction of the AI complexity-interpretability trade-off. The conceptual scatter plot on the right shows the current AI black-box dilemma. Generally, as AI performance increases, our ability to understand how and why they come to the conclusions that they do generally decreases [47].

making the recommendations it is could literally be a matter of life and death [49, 77]. AI cannot be given a *carte blanche* to make decisions with such potentially grave and final consequences.

Similarly, while complicated scientific AI algorithms can be remarkable in their abilities to learn complex patterns from noisy data sets, their value is greatly reduced because of their inherent resistance to introspection [78]. While relatively-less complicated AI models, such as decision tree ensembles, offer more intuitive ways to interrogate individual feature contributions,

As a result of the need for explainable artificial intelligence (XAI), the past three years has witnessed an explosion in the number of XAI algorithms, with each model temporarily claiming supremacy over its predecessors [9]. In 2017, however, Lundberg and Lee published an XAI model, based on cooperative game theory, that unified and advanced several of the leading popular methods. Their framework, which they refer to as Shapley Additive Explanations (SHAP), is based on the Lloyd Shapley's work in game theory and calculates a unique set of additive feature importance measures for a given model and data set [75].

Shapley values were introduced in 1953 as a solution to the question of how to distribute

gains obtained through a coalitional game to the participating players, assuming that those players all cooperate with each other [95]. If one desires a "fair" distribution or payment rule, then it ought to possess the following properties:

1. *symmetry*: *symmetry* requires that if two players contribute equally to the game, then their payouts must be equal. This property is essential to our concept of fairness;

2. *efficiency*: *efficiency* requires that the sum of distribute gains must be equal to the gain of the coalition. In other words, there cannot be anything left undistributed, indicating that all players have received their maximum possible payment;

3. *linearity*: also called the "law of aggregation" by Shapley, *linearity* requires that if two independent coalitional games are combined, then the sum of their two total gains must equal the distributed gains summed player by player. This property is required in order for Shapley values to be calculated for a machine learner's features.

Shapley demonstrated that, for a unique coalitional game, there exists only one value, $\varphi_i(v)$, that satisfies all three of these criteria:

$$\varphi_i(v) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} (v(S \cup \{i\}) - v(S)) \tag{3.19}$$

where i refers to the i^{th} player, v is the coalitional worth function, F is the total coalition of players, and S is a subset of N that does not contain player i [52].

Lundberg and Lee extended the scope of Shapley values to machine learning models by introducing the paradigm of *explanation models*. For a specific machine learner f, they define an *explanation model* g to be a function of binary variables that is an "interpretable approximation of the original model"; that is, g closely approximates f, but in such a way that it is interpretable [75]:

$$g(z') = \varphi_0 + \sum_{i=1}^{M} \varphi_i z'_i \tag{3.20}$$

where $\varphi_i \in \mathbb{R}$ and M is the number of simplified binary features z'. By equating the machine learning model in question to a coalitional game, they showed that it is possible to use Eqn. 3.19 to calculate unique Shapley values for each feature for any given input sample. By leveraging this model-agnostic approach, essentially every machine learning model in existence becomes explainable.

We trained our XAI models over both our XGBoost and LSTM-RNN learners, using Lundberg and Lee's Python library *shap* [74].

Model-specific SHAP algorithms

In order to introspect the tree ensemble learners that trained on the standardized experiments, we utilized *shap*'s TreeExplainer. TreeExplainer leverages the structure of the component trees to greatly increase the calculation of the model's SHAP values [76]. Analysis of these SHAP values was conducted in Jupyter Notebooks on a 2012 Mac Pro with 24-cores and 48 GB RAM.

Similarly, the SHAP explainer models for the LSTM-RNN deep learners was built using *shap*'s DeepExplainer algorithm [75, 97]. All training was conducted on the aforementioned cloud server, in order to utilize the server's GPU. As a result of the algorithm's memory requirements, another custom generator class was created and training was only performed on an experiment-by-experiment basis. Training a separate explanation model on each experiment is not a problem, however, as a consequence of the Shapley values' linearity property, as described in List 3.2.12. All calculated SHAP values were cached in permanent memory for subsequent analysis.

Analysis of SHAP values (combining redundant strains, etc.)

When analyzing the SHAP values from both XGBoost and LSTM-RNN learners, the explanation models for each experiment were aggregated into a meta-explanation model. Since the inputs to the LSTM-RNN models were 2-D time windows, rather than 1-D vectors representing a single time point, the SHAP explanation model returned a 2-D matrix of corresponding SHAP values. Recall that, for a machine learning model f and its corresponding explanation model g,

that the sum of the SHAP values at a given time point will closely approximate the output of f (see Eqn. 3.19). Thus, in order to assign a single SHAP value to each time point in the experiment, the explanation model's $(num\_timesteps) \times (num\_strain\_positions)$ matrix output were summed row-wise to produce a $1 \times (num\_strain\_positions)$ row vector. This row vector then represented the SHAP values of each strain-position for the given time point.

Also, since a certain subset of reference strains occupied redundant positions, we analyzed the XGBoost and LSTM-RNN explanation models in two separate ways. In the first method, we analyzed each strain-position as an independent feature, regardless of whether the strain occupied redundant positions or not.

In the second method, we took each redundant strains' positions' SHAP values and calculated their means on a strain-by-strain basis. In addition, we calculated the mean each redundant strains' positions' features. By carefully tracking indices in a number of different arrays, we were able to drop all redundant strain-positions from both the SHAP values and the feature sets; we then concatenated together the single-strains' SHAP and feature values with the redundant strains' newly calculated mean SHAP values and mean features.

3.3 Results and discussion

3.3.1 Flat-field correction

The automated flat field correction pipeline effectively corrected all optical vignetting in our DynOMICs optical systems. This result was checked in a variety of ways.

First, we performed a visual inspection in both the fluorescent and transmitted fields of pre- and post-correction images (Fig. 3.5a and Fig. 3.5d). While these corrections appeared satisfactory for all three DynOMICs devices, we inspected pre- and post-correction heatmaps of the local background intensities across the device. These heatmaps revealed that, while some variation still persisted, it was markedly reduced, especially in the transmitted light channel (Fig. 3.20a and

Fig. 3.20b).

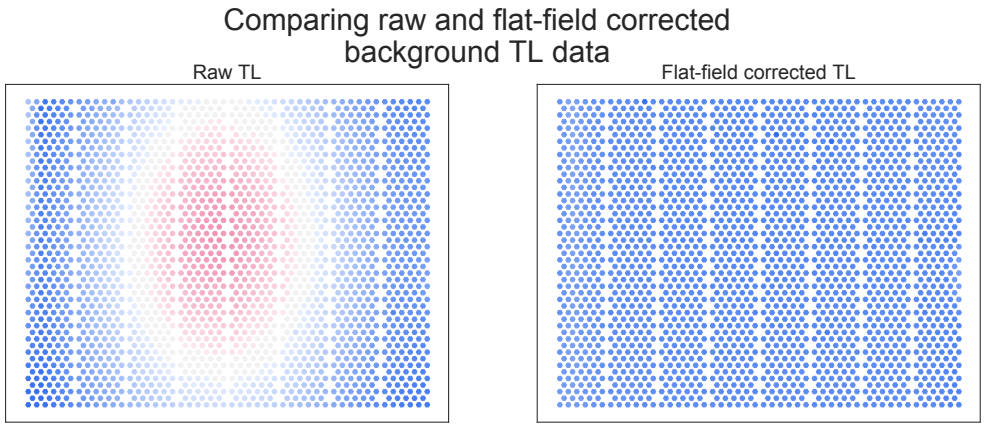
Finally, for each DynOMICs device, we calculated the mean and standard deviations of the intensities for a series of images of static microfluidic devices, identical to the images in Fig. 3.5a and Fig. 3.5d. These statistics confirmed that the standard deviations in the FFC-corrected images were less than the mean of the variance of the strain-positions' FL and TL trajectories (Table 3.1). This indicated to us that the remaining optical vignetting could no longer bias our time series data.

3.3.2 Data preprocessing and feature engineering

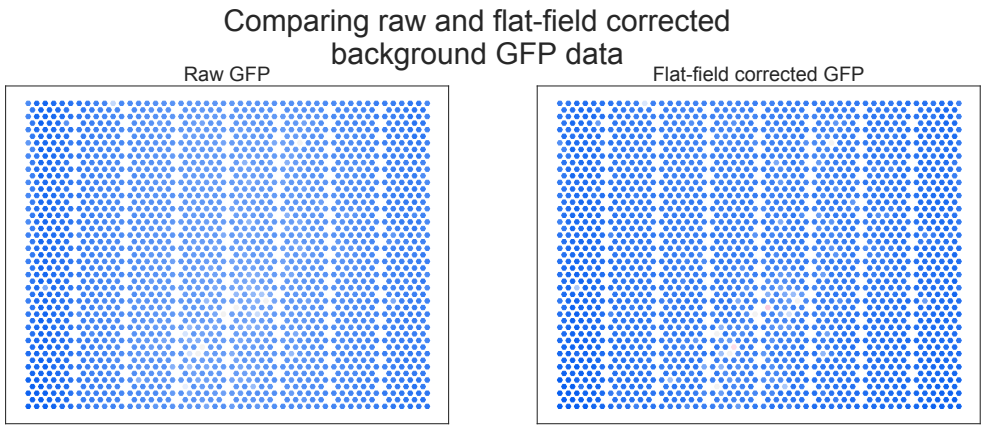
In any task involving data analysis and machine learning, the preprocessing of the data and subsequent feature engineering (or absence of it) represents the most influential part of the process. Choices made here can either selectively mask information and amplify noise or, far more preferably, silence noise in the data in order to better emphasize features with high information content. While deep learning neural networks have demonstrated the ability to work with raw data and extract their own standardized features, an ability known as representation learning, most machine learning algorithms do not posses this capability [69]. Our project faced the daunting obstacles of a combination of fluidic, thermal, electrical, and biological noise sources. Indeed, we even had to account for rasterization-induced noise that was caused by the diurnal thermal expansion and contraction of the building in which the DynOMICs devices were housed. Thus, it was critical to our project that we engineer meaningful features.

Table 3.1: A statistical summary of the effects of flat-field corrections. This table presents the means and variances of the images, both from before and from after flat field correction. The reduction of both transmitted light and fluorescence channels' variances to well below the mean strain variances in those channels satisfied us that the flat field correction was performing well.

	variance	**mean**
background TL	9,488	31,880
strain TL	911	23,541
background TL-FFC	614	24,847
background FL	465	2,701
strain FL	2,829	2,709
background GFP-FFC	352	1,538



(a)



(b)

Figure 3.20: Heatmaps of both raw and flat field corrected TL and FL backgrounds by device position. (a) The TL signal quality improved significantly after the automatic FFC pipeline was implemented. These two heatmaps, which show the near-total elimination of TL vignetting, are comparable for all three of the DynOMICs boxes. (b) The quality of fluorescence data improved as well, albeit only slightly. There is a minimal amount of vignetting in the center of the raw fluorescence map; this vignetting is eliminated in the flat field corrected fluorescence heatmap.

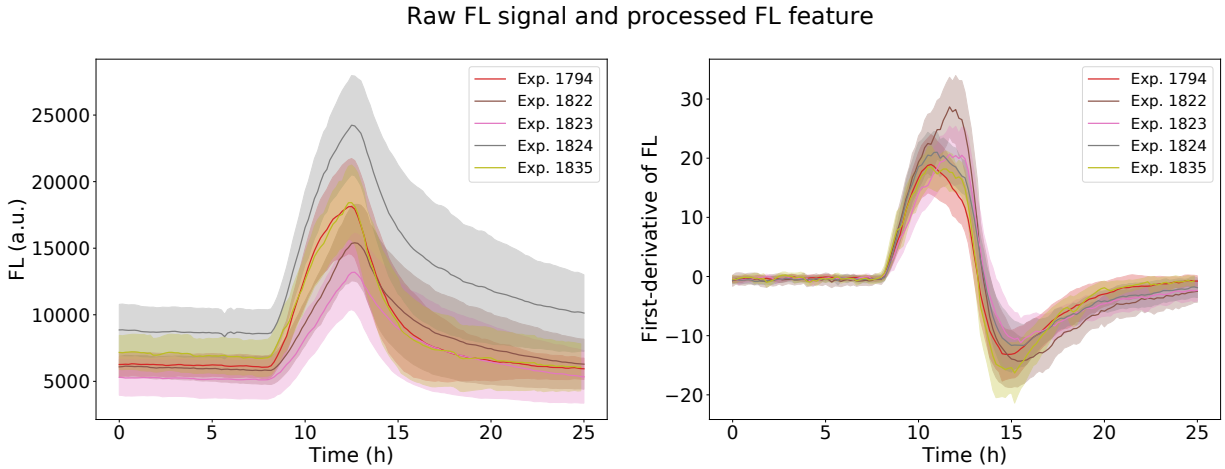


Figure 3.21: A comparison of the raw and processed fluorescent signals. Our feature engineering eliminates a significant amount of intra-experiment variability by rendering the raw signal into a first derivative-like feature. This variability is due to differing hardware between DynOMICs devices, among other sources.

Through a combination of domain knowledge and experimentation, we successfully engineered meaningful features that were based on the first derivative of the time series data. A comparison of raw traces and engineered features from two different DynOMICs boxes anecdotally demonstrates how the feature processing pipeline eliminates differences due to hardware, collapsing otherwise incomparable trajectories onto one another (Fig. 3.21).

3.3.3 Implementation of Lubansky data differentiation

The Python implementation Lubansky differentiation served as an excellent sanity-check that we were not biasing our data via assumptions implicit in our feature engineering methods (Fig. 3.22). Since the Lubansky method relies minimal assumptions about the nature of the data and does not need regularly sampled data for successful differentiation, it is an excellent option for any researcher in need of derivative-based features.

However, since the output of our feature extraction pipeline proved so similar to the output of Lubansky differentiation performed on the same raw data, we decided to forego the lengthy process of converting our Python implementation to an implementation in C and subsequently

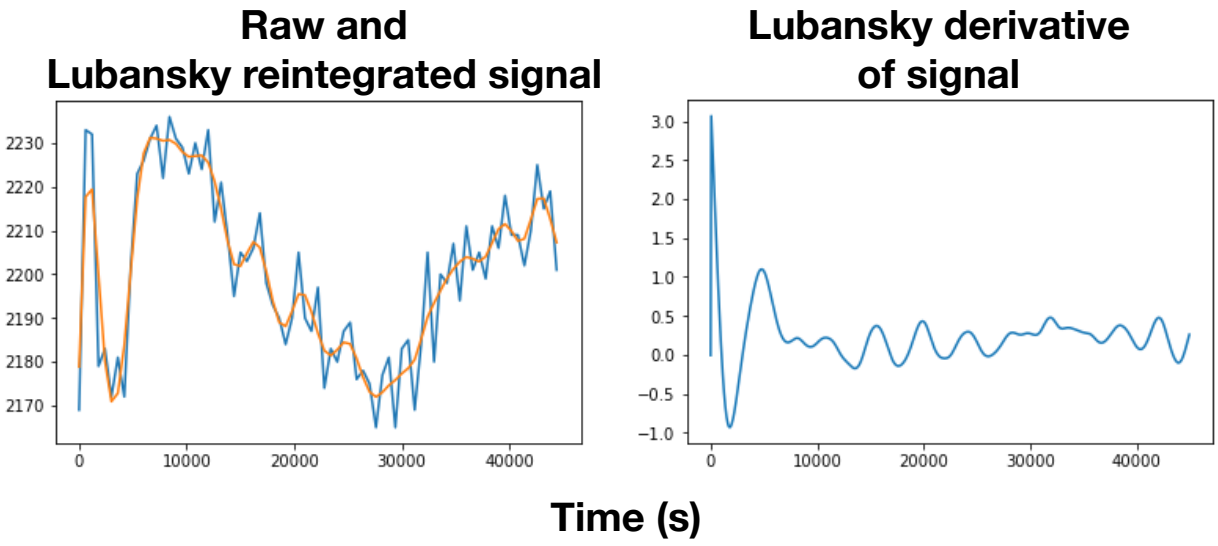


Figure 3.22: First derivatives of Lubansky differentiated experimental data. The left-side panel displays the strain's raw fluorescent signal (blue) and the smoothed signal that results from numerically integrating the calculated first derivative (orange). The plot to the right displays the Lubansky-calculated first derivative of the raw experimental data. One remarkable property of this technique is that *the researcher makes no explicit assumptions as to the nature of the data or the noise*. The only assumption that is made is the one imposed by the Tikhonov regularization step: that the simplest solution is the correct solution.

optimizing that code. Converting and optimizing would have created computational speed increases that would have allowed for efficient use of Lubansky differentiation. In addition, since Lubansky differentiation requires an entire experiment's data set in order to differentiate that data, it is not suitable for a real-time sensor application. Therefore, we made the decision to continue relying upon our aforementioned feature extraction pipeline for the remaining analysis and machine learning.

3.3.4 Initial supervised learning with tree ensemble learners

Our initial attempts to use machine learning classifiers to detect heavy metal inductions in our data were met with mixed success. As shown in Fig. 3.23, results from random forest classifiers suggested that our classifiers could readily detect cadmium, zinc, copper, and iron.

However, detection of lead and chromium was barely better than random guessing, while detection of arsenic and mercury was on par with and sometimes, when using the F_1-micro evaluation metric, performed worse than random guessing. As discussed in the Methodology section, performing

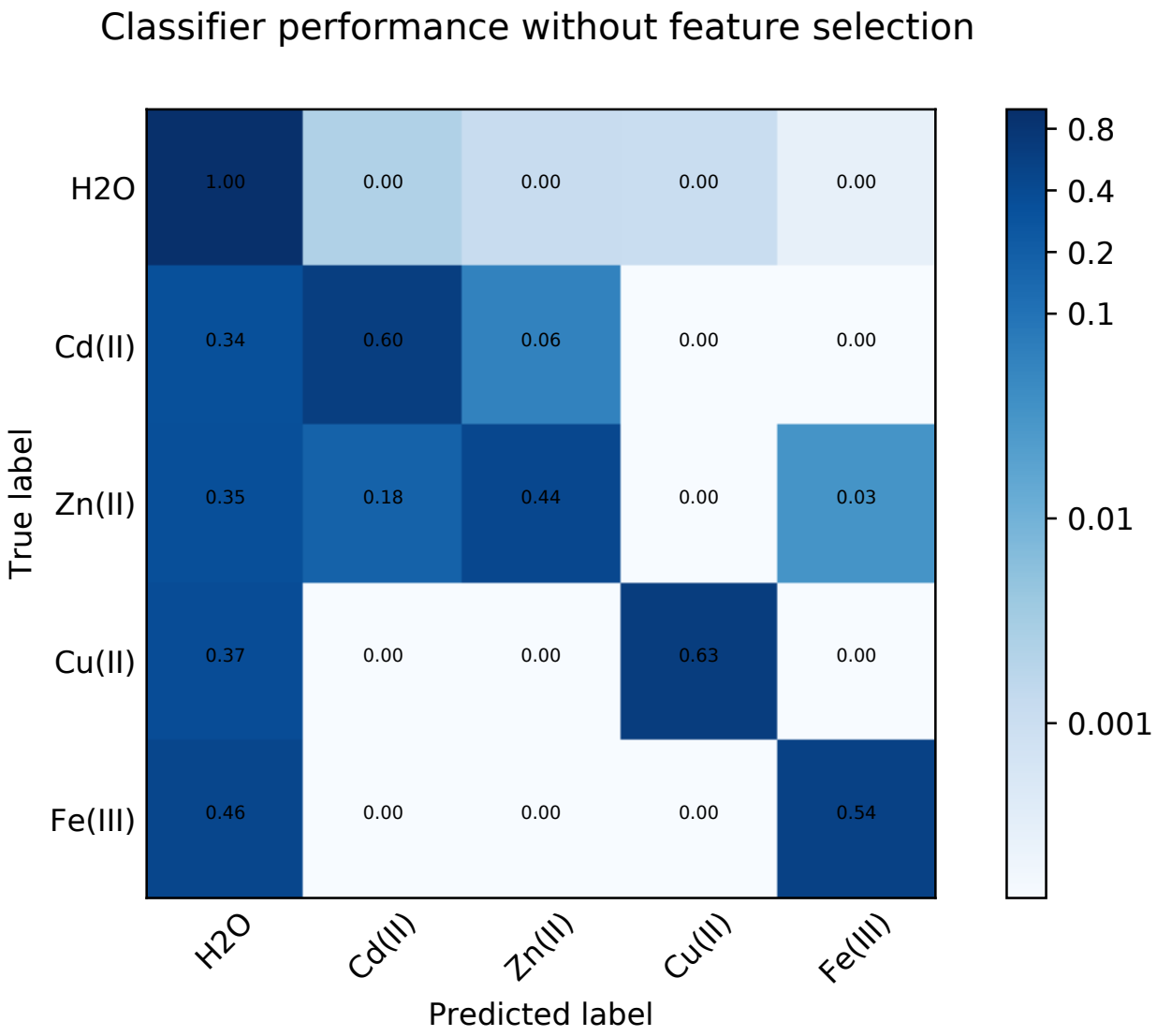


Figure 3.23: Resulting recall scores of optimized random forest classifiers. This confusion matrix displays the cross-validated recall scores for the optimized classifiers. We successfully classified four metals without feature selection. However, while both hexavalent chromium , Cr(VI), and divalent lead, Pb(II), appeared to be detected, the results were so weak that they were dropped from further classification by random forests. Instead, we determined that only rigorous feature selection enabled effective classification by these tree ensemble learners. Note that the color bar is logarithmic.

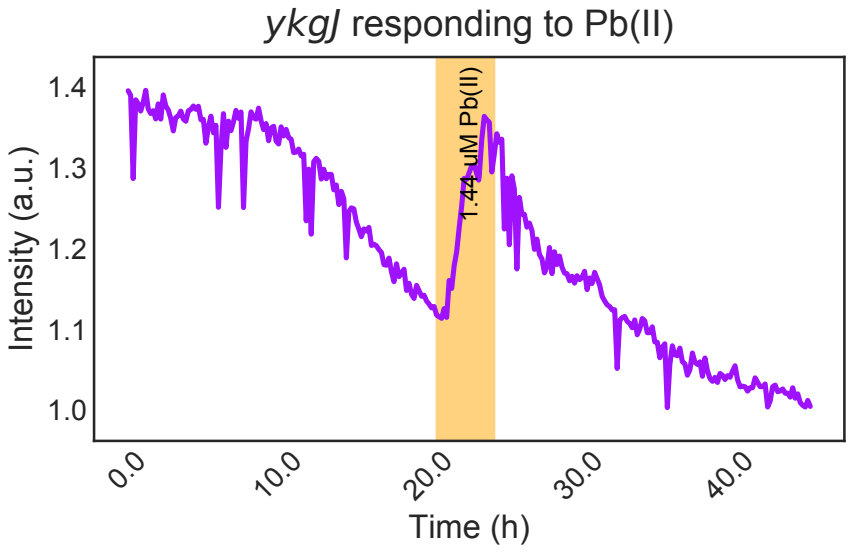


Figure 3.24: The strain *ykgJ* responding to Pb(II). *ykgJ* is just one of several strains, including the promoter strain representing the σ-factor gene *rpoH*, that were reliably sensitive to lead. Chromium had even more strains that responded reliably to it. Therefore, we found it odd that our classifiers performed so poorly on these metals. Their poor performance on these metals called into question whether they truly could not detect arsenic or mercury, or if it were simply that the signals of the relevantly responsive strains were being drowned out by noise.

worse than random guessing while using the F_1-micro function as the evaluation metric would only occur if the learner could truly not perform any better than random guessing.

We thought these results were odd for several reasons. In the case of lead and chromium, we had manually identified several reliably responding strains for each (Fig. 3.24). That we could infrequently classify chromium correctly, but never lead, we thought was strange, given the identification of the lead-sensitive strains. especially since the *E. coli* genome contains the *ars* and *mer* operons. While our *E. coli* GFP promoter library did not possess any promoters from either of these operons, we thought that our classifiers would be able to detect at least a transcriptional perturbation in other promoter sets from the activation of both *ars* and *mer* operons [120].

The fact that the exploratory classifiers were failing to reliably detect lead and chromium, which both had responsive strains, cast doubt on whether arsenic and mercury were truly undetectable. One plausible hypothesis was that they were detectable, but that their signals were too weak to be picked out of such a high-dimensional, noisy data set. If this scenario was indeed true, then

classification performance would be able to improve with a more powerful machine learning algorithm or with better feature selection or with both a more powerful classifier utilizing a more informative suite of features.

In order to test this hypothesis, we decided to first test a more powerful classifier, since this approach is usually less time-consuming than identifying more informative features (the machine learning equivalent to finding a ”needle in a haystack”). For our next classifier, we selected the tree ensemble classifier algorithm known as *extreme gradient boosted trees* (XGBoost). XGBoost has a reputation as being one of the most capable non-deep learning AI algorithms. In addition, it is inherently somewhat interpretable due to its component trees. Therefore, we decided to use it to attempt binary classification on these four most difficult-to-detect metals: chromium, lead, arsenic, and mercury.

After subsetting each of these metals down to the simplified binary case of metal versus water, only chromium was successfully detected (Fig. 3.25). Even then, detection was poor at best, barely outperforming a dummy classifier pursuing a strategy of guessing according to class frequencies in the two-class data set. As a consequence of this outcome, the enhanced feature selection strategy appeared to be our only option for guaranteeing that our inability to reliably detect these metals was not a false negative.

3.3.5 Statistical distance distribution searches for the identification of lead-, arsenic- and mercury-sensitive promoters

In order to rule out the possibility that Pb-, As- and Hg-responsive strains were being overlooked by the exploratory classifiers, we plotted and then manually inspected the energy distance distributions for every strain. This approach allowed us to inspect, simultaneously, all inductions across all experiments for a given metal. In the case of strains with redundant positions, we combined their energy distances into a single distribution.

Strains that were known to reliably respond to specific metals were plotted and compared

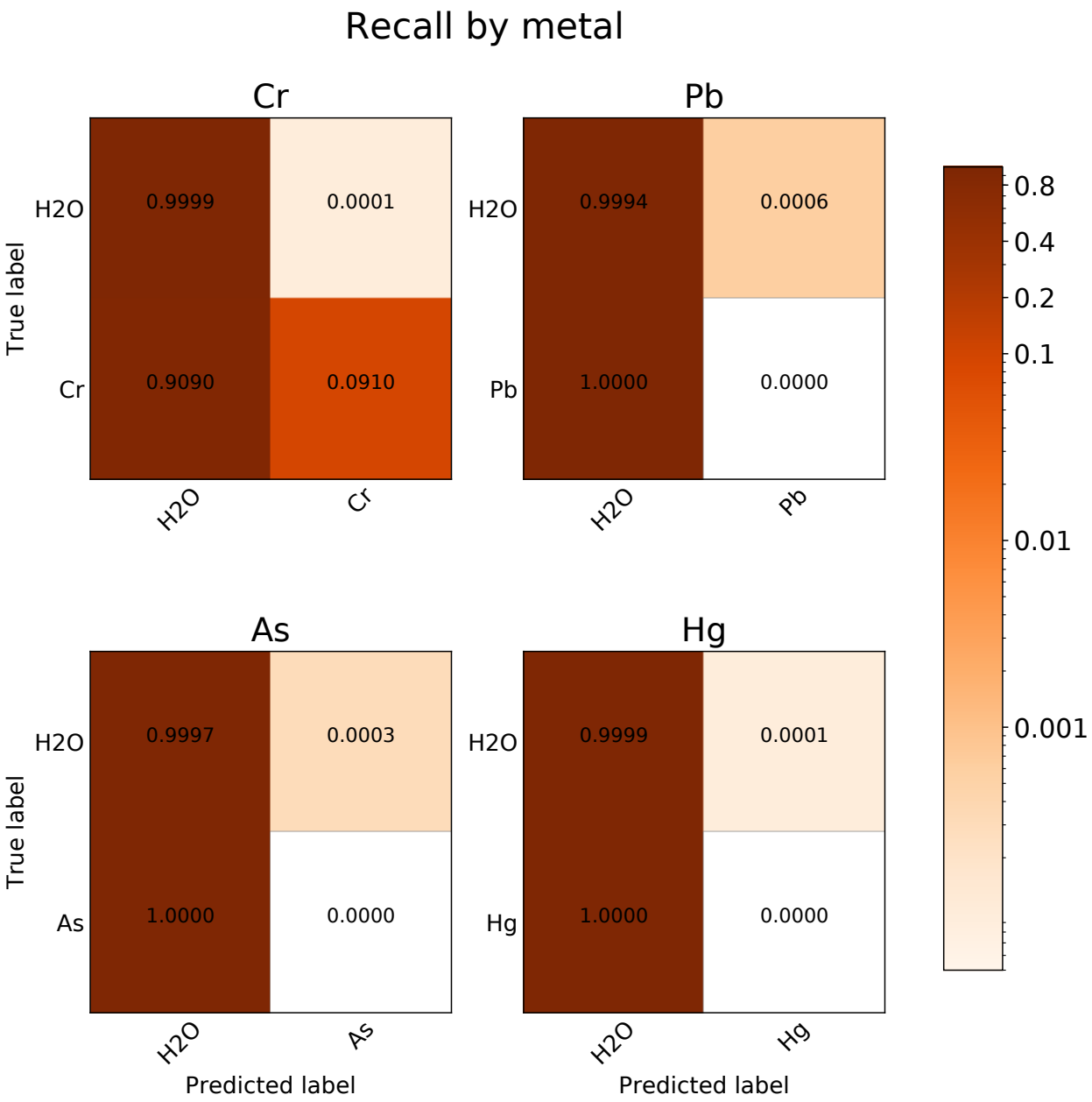


Figure 3.25: The exploratory tree ensemble classifiers failed to detected lead in a binary unoptimized classification setting without feature selection. We found this result surprising, since lead was known to have several reliably-responding strains. Cadmium, on the other hand, was detected a small percentage of the time. This result suggested that cadmium classification performance could improve with better feature selection. The combination of these two results cast doubt on whether the classifiers' failures to detect arsenic and mercury were false negatives, similar to lead, where the classifiers were not able to identify responsive strains, or actually true negative results. This line of reasoning was reinforced by the fact that the *ars* and *mer* operons are the only two metal-sensing operons in the *E. coli* genome dedicated to detecting metals without biological function. While neither operon had representative strains on-chip, we assumed that the classifiers could at least observe a transcriptional perturbation resulting from the activity of these operons.

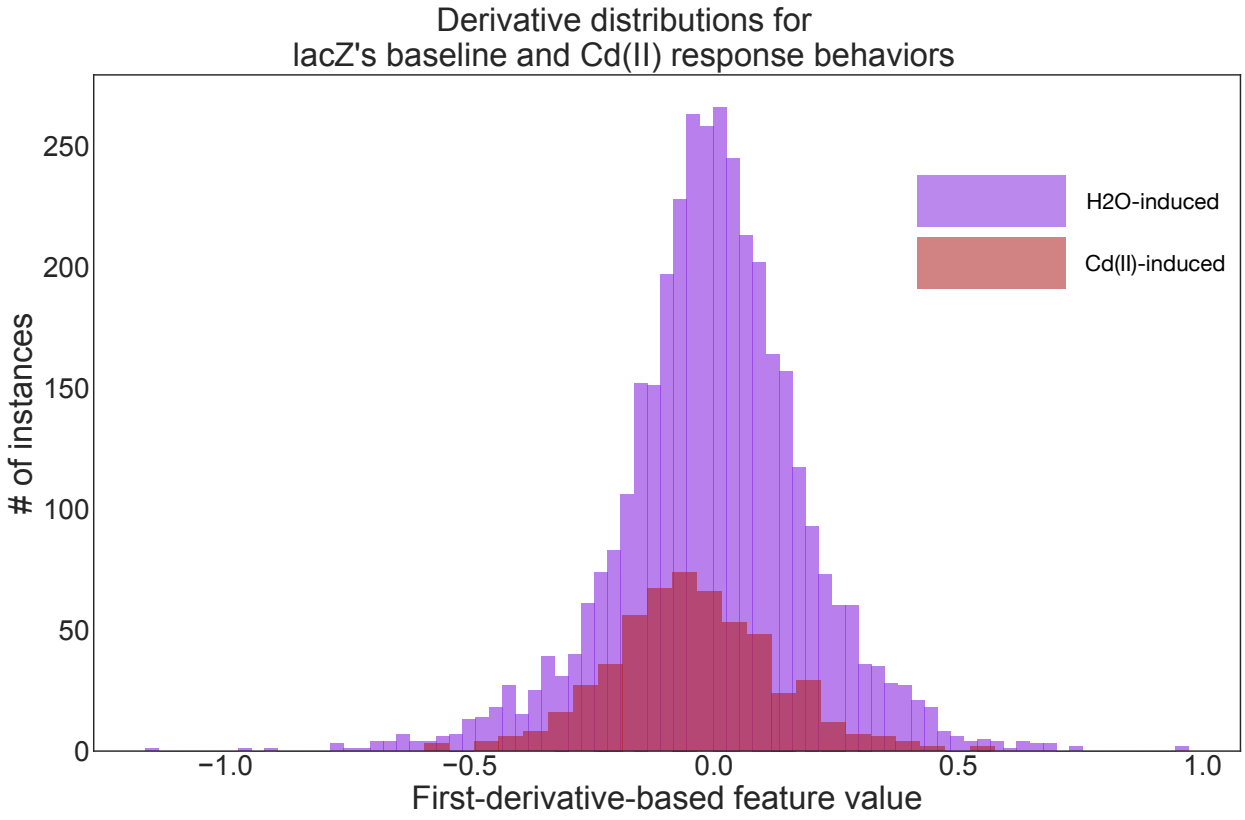


Figure 3.26: The control strains' water- versus cadmium-induced first-derivative feature distributions. These two co-plotted distributions are exemplify how, for a typical metal induction, the behaviour of control strains such as *lacZ* are indistinguishable from their behaviour under normal growth conditions. Water's high count number is a consequence of our data set's severe class imbalance.

against the behavior of two control strains: the promoter-less, negative control strain *U139* and positive control strain *lacZ* (Fig. 3.26). Each of those strains occupied multiple positions in the original version of the Alon *E. coli* GFP promoter library, for the express purpose of serving as redundant controls [120].

We visually inspected each energy distance distribution for all of the 1,807 unique on-chip strains across all relevant experiments and inductions. We performed this inspection for lead, chromium, arsenic and mercury induction behaviours. While we identified several additional strains that were potentially lead or chromium-sensitive, none were identified for arsenic or mercury (Fig. 3.28).

To be certain that this result was indeed true, we finally plotted the trajectories of every strain during all arsenic and mercury inductions (Fig. 3.29). The result was that we manually inspected 16,800 different distributions and trajectories for mercury and arsenic sensitivities. In the end, we were confident that no strain exhibits a consistently detectable genomic response to either arsenic or mercury.

One possible reason for the seeming obliviousness of the observed portion of the *E. coli* genome to induction by arsenic and mercury is the special role that these two metals have played in the evolution of *E. coli* metal tolerance. Arsenic and mercury are the only two metals that play no biological role within *E. coli* that nevertheless have operons exclusively dedicated to detecting, chelating, and exporting these two metals; these operons are the *ars* and *mer* operons, respectively [27]. Other heavy metals without discernible biological roles in the species are sensed and exported via dual-purpose pathways, evolved for managing concentrations of physiologically-relevant metals such as iron and zinc [79]. One example of this species of pathway is the *znt* operon in *E. coli*, which manages intracellular zinc concentrations. While *znt* appears to evolved primarily for zinc, which is an essential metal in *E. coli*, the operon is also responsible for sensing, capturing and exporting cadmium, a purely toxic metal that plays no known role in an living organism [20, 37].

Indeed, further literature searches revealed that the tested concentrations of arsenic and mercury, while at or above the safety limits set by the Environmental Protection Agency, are still

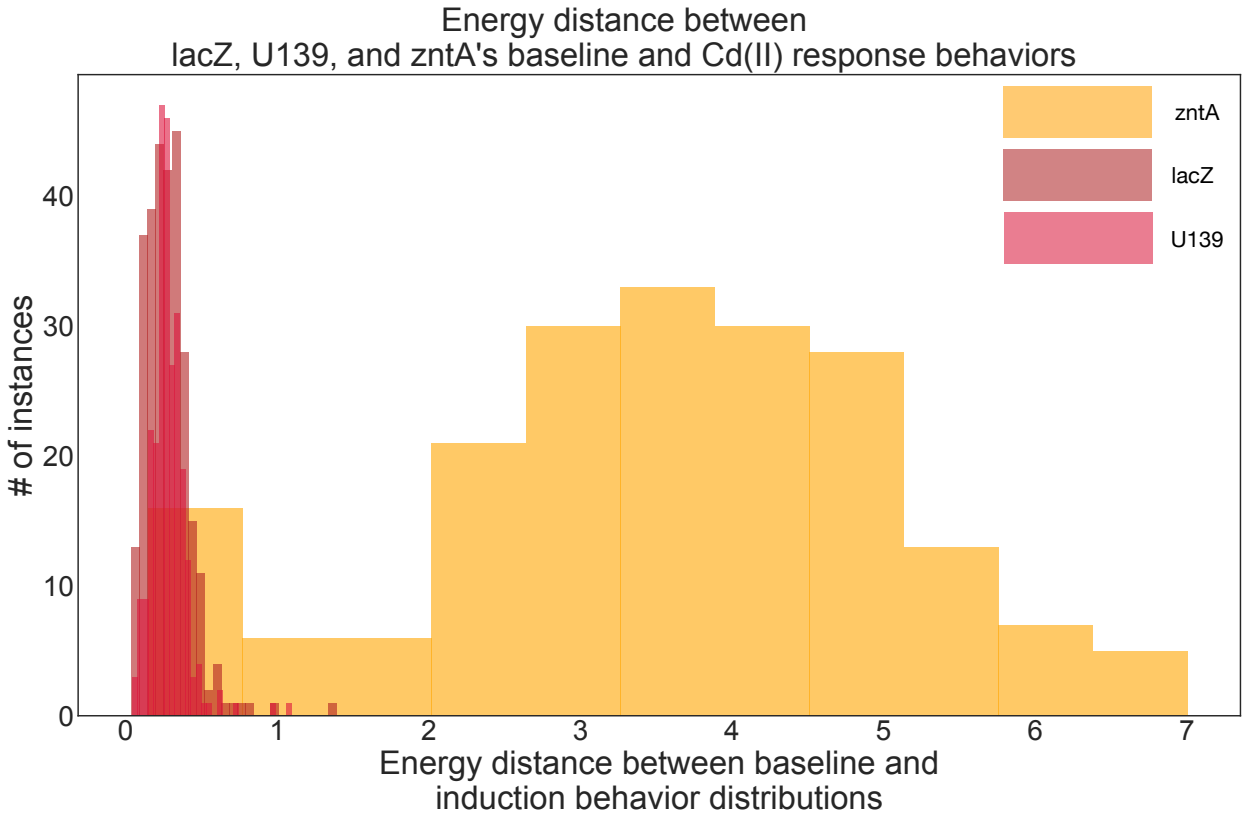


Figure 3.27: A comparison of induced-uninduced energy distances between a metal-sensitive strain and the on-chip control strains. The energy distance distributions that co-plotted here are an efficient and intuitive method for simultaneously evaluating a strain's sensitivity to a single metal across all of that metal's inductions across all eighteen standardized experiments. Here, the control strains neatly overlap with one another, indicating the equivalency of behaviours indicative of metal insensitivity. The cadmium-sensitive strain *zntA*, however, has an energy distance distribution that is markedly different than those of the control strains'. By examining approximately 8,000 of these plots, we were able to identify candidate responding strains for Pb, Cr, As, and Hg across our entire data set.

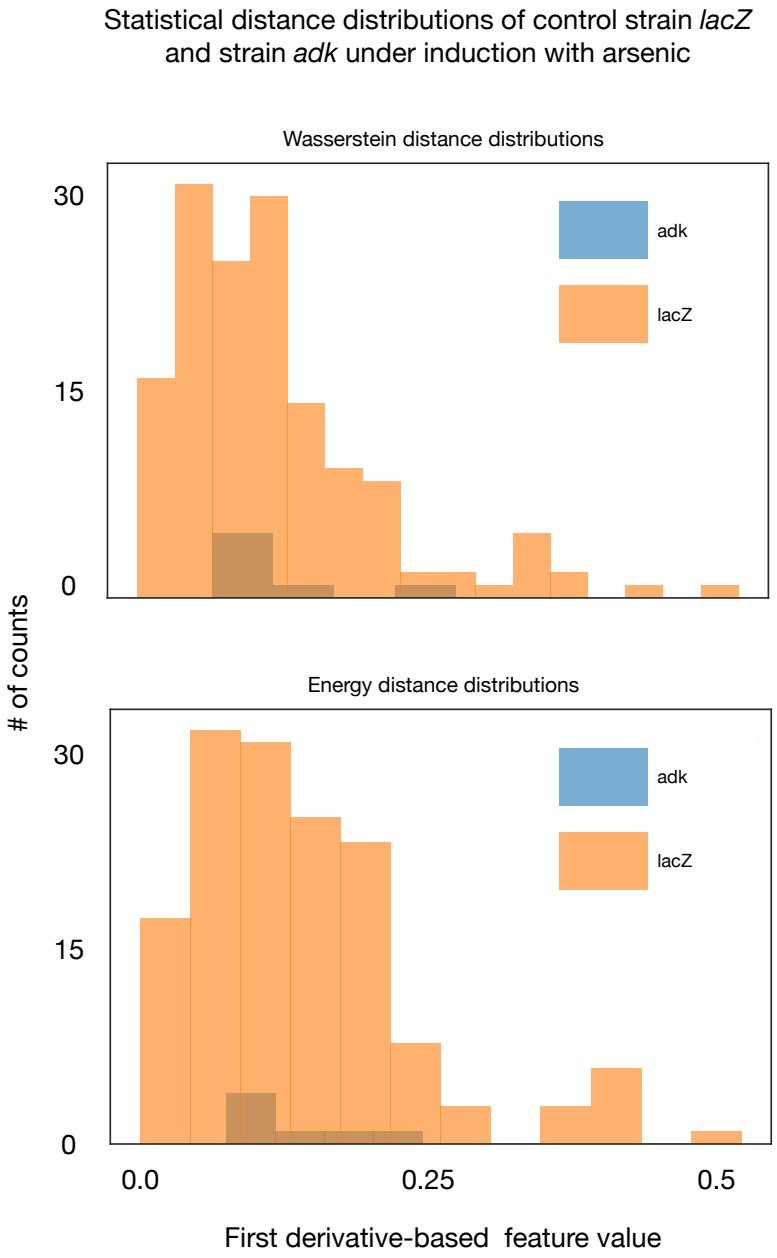


Figure 3.28: Statistical distance distributions of an arsenic-insensitive strain. The overlaid Wasserstein and energy distance distributions depict the behaviors of the strain *adk* and the control strain *lacZ* when their arsenic-induced behaviors are compared with their behaviors growing on pure minimal media just prior to said inductions. The *adk* distributions are fully enveloped by the distributions of the control strain. As the control strains were verified to be completely unresponsive to arsenic exposure in every experiment, the lack of separation between the two distributions indicates that *adk* is also unperturbed by exposure to the concentrations of arsenic used in our experiments.

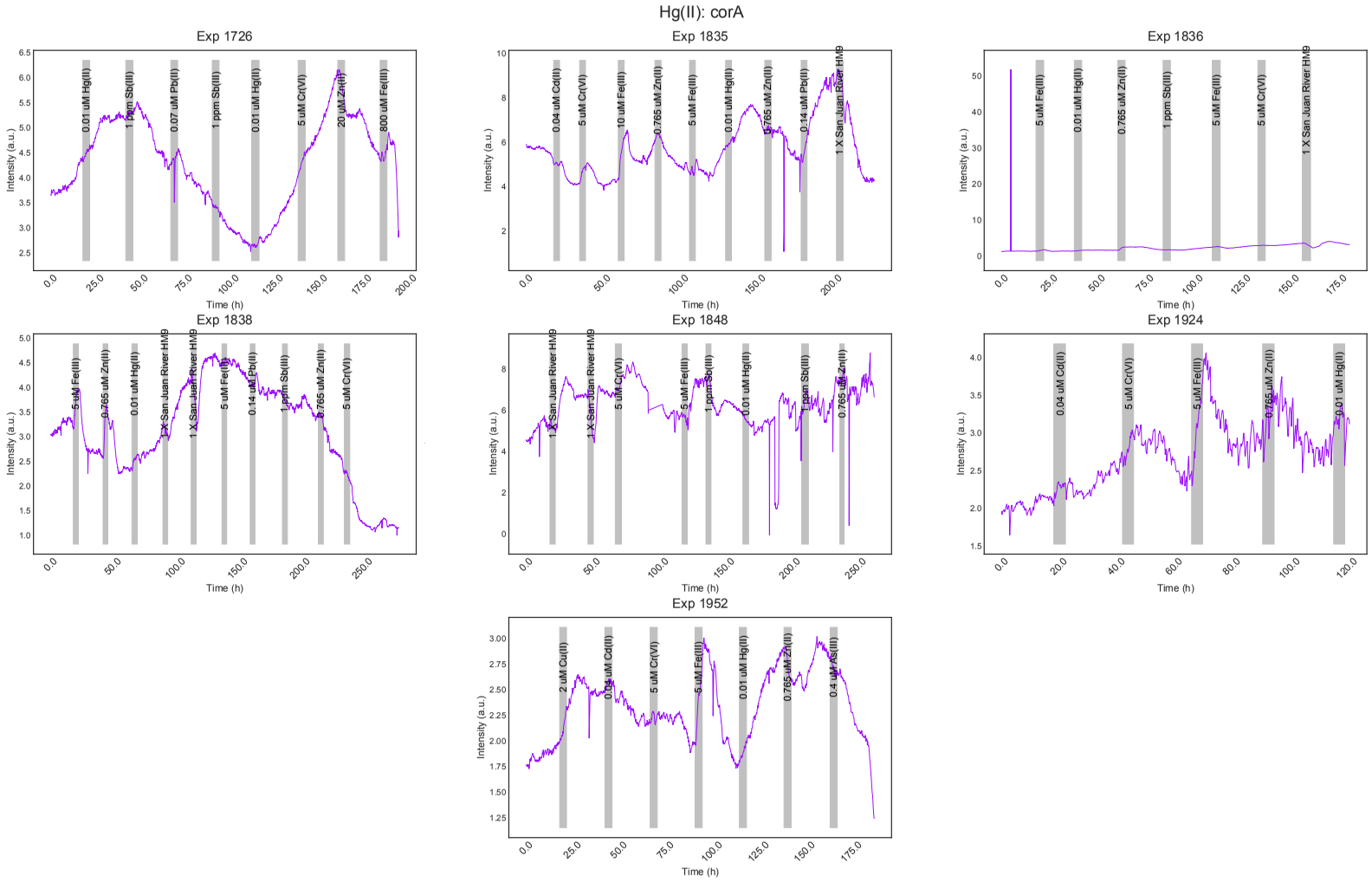


Figure 3.29: Mercury response trajectories. The above plots represent the raw fluorescent signals of the arbitrarily-chosen strain *corA* in all seven of the mercury-induction experiments. Identical plots were generated for each of the 1,995 non-empty device positions for all mercury- and arsenic-induction experiments to guarantee that the energy distance surveys did not miss a responsive strain. In the end, we were confident that no such strain existed in our library. Note that this plot has been rotated 90 degrees in order to allow for a higher resolution image.

well below reported minimum inhibitory concentrations (MIC) for *E. coli* [101]. Arsenic was tested at a maximum of 0.4 μM, while its reported MIC in some *E. coli* strains is 1.75 mM; mercury was tested at a maximum of 0.8 μM, while its reported MIC is 3 μM on minimal media [101] [8]. Since these concentrations are so much lower than the MIC for most *E. coli* strains, it is unlikely that the extremely efficient *ars* and *mer* operons encountered any difficulties in rapidly effluxing the toxic metal ions from the cells. However, most of the reported MIC were determined via testing on agar plates or in batch cultures, so further testing would be required in order to determine these metals respective MIC in microfluidic environments.

3.3.6 Supervised learning with extreme gradient boosted trees (XGBoost)

Once it had been determined that there were indeed no strains sensitive to arsenic and mercury, all arsenic, mercury, and antimony labels were dropped from the data set and we proceeded with classification on the remaining metals. These metals were iron, copper, zinc, cadmium, chromium, and lead.

Once we had finished manually tuning the XGBoost hyperparameters, we were possessed two different classifiers with varying properties. The first classifier was composed of 22 manually selected feature strains, comprising 118 unique device positions. These strains had been observed in the course of the project to respond with varying degrees of reliability to the metals-of-interest. The resulting classifier represented our best baseline against which to compare classifiers constructed from the entire library and represented the eventual specific sensor that could be constructed by beginning with a nonspecific sensor and learning the stimuli-of-interest.

The behavior of this best-case classifier can be evaluated by inspecting the confusion matrices in Fig. 3.30. Note that these matrices are in log-scale format, as opposed to the later matrices of the LSTM-RNN, which are depicted using a linear scale. The relatively high degree of similarity between the training and testing matrices in these plots indicates that the manual hyperparameter tuning process, while probably not optimal, still managed to prevent overfitting on the training sets. The confusion matrices were created via the aforementioned leave-one-out cross-validation

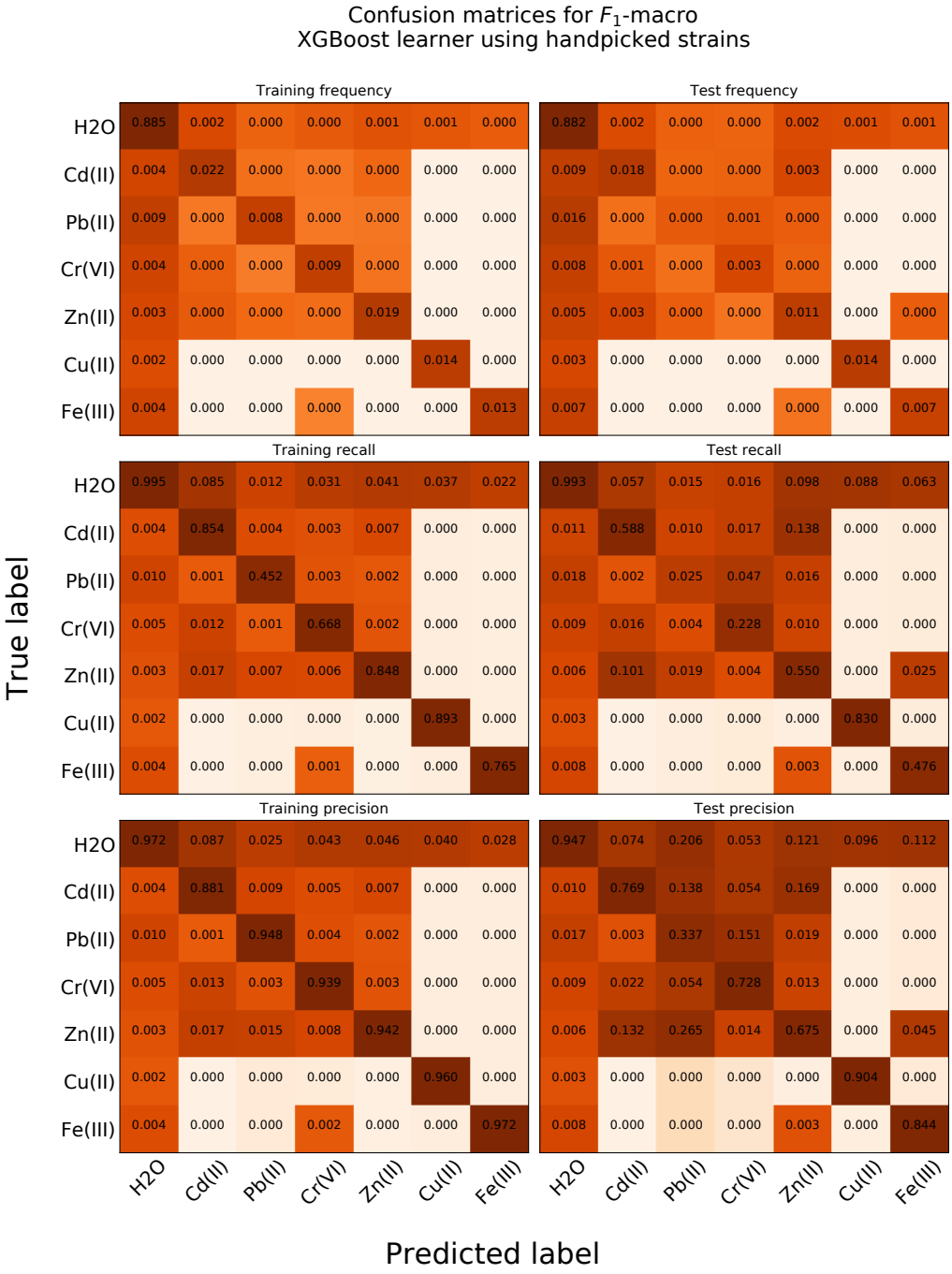


Figure 3.30: A summary of the performance of an XGBoost classifier using handpicked strains. These confusion matrices represent the performance of a manually tuned XGBoost classifier using handpicked strains as features, as opposed to the full library. The hand-tuning did an excellent job at avoiding overfitting, as can be discerned by the relatively high degree of similarity between the training and testing matrices. However, in the case of toxic metals, it would be desirable to have higher recall scores, since recall is a good metric for how frequently the classifier fails to detect the presence of a specific metal. In the case of toxic metals, sacrificing precision for recall is preferable (ie, better safe than sorry). Note that all shading is done via a log scale and that all numbers were rounded to three decimal places, in order to preserve legibility. These classifiers were created by using leave-one-out cross-validation and taking the mean of the results.

procedure that gives a fair estimation of how well such as classifier would generalize to new data.

The XGBoost classifier represented by Fig. 3.30 is performs far better than random, but still falls short of what we the performance levels we would like to see for a multi-metal field-deployable sensor. For instance, the test recall for cadmium is 58.8%, indicating that the sensor detects the presence of the metal nearly three-fifths of the time that it is actually present. However, it is only able to do the same for lead approximately 2.5% of the time, which is less than ideal. These results are better than an optimized XGBoost classifier that was trained on the entire library, without feature selection. Although still better than random, this classifier has all around far worse recall and precision scores. As a result, we moved to testing the long short-term memory recurrent neural networks as a candidate sensor AI.

3.3.7 Supervised deep learning with long short-term memory recursive neural networks

While deep learning networks require specialized hardware and, usually, longer durations of time to train and optimize, it was well worth the time invested. The optimized LSTM-RNNs, using a surrounding time window of $\pm$ 1 hour, proved themselves to incredibly adept at detecting heavy metal stress while using the full library. Comparing the confusion matrices in Fig. 3.31, we can see that its performance far outstrips that of even the best-case XGBoost learner (see Fig. 3.30). *Notice that while Fig. 3.30 is using a log scale color scheme to emphasize the cell values, Fig. 3.31 is not!*

In addition to using confusion matrices, we visualized each classifier's performance using the prediction versus-time plots seen in Fig. 3.32. In each experiment's plot, the top row, labeled "Actual", indicates which substance is present in the sensor throughout the course of an experiment. A colored bar indicates the presence of a metal (as denoted by the color codes in the legend), while no color indicates that the cells were growing on pure HM9 minimal media at the time. The second row, labeled "Predicted", indicates the real-time prediction of the deep learning network as it is passed the never-before-seen experiment's data. A color match indicates that the LSTM-RNN is

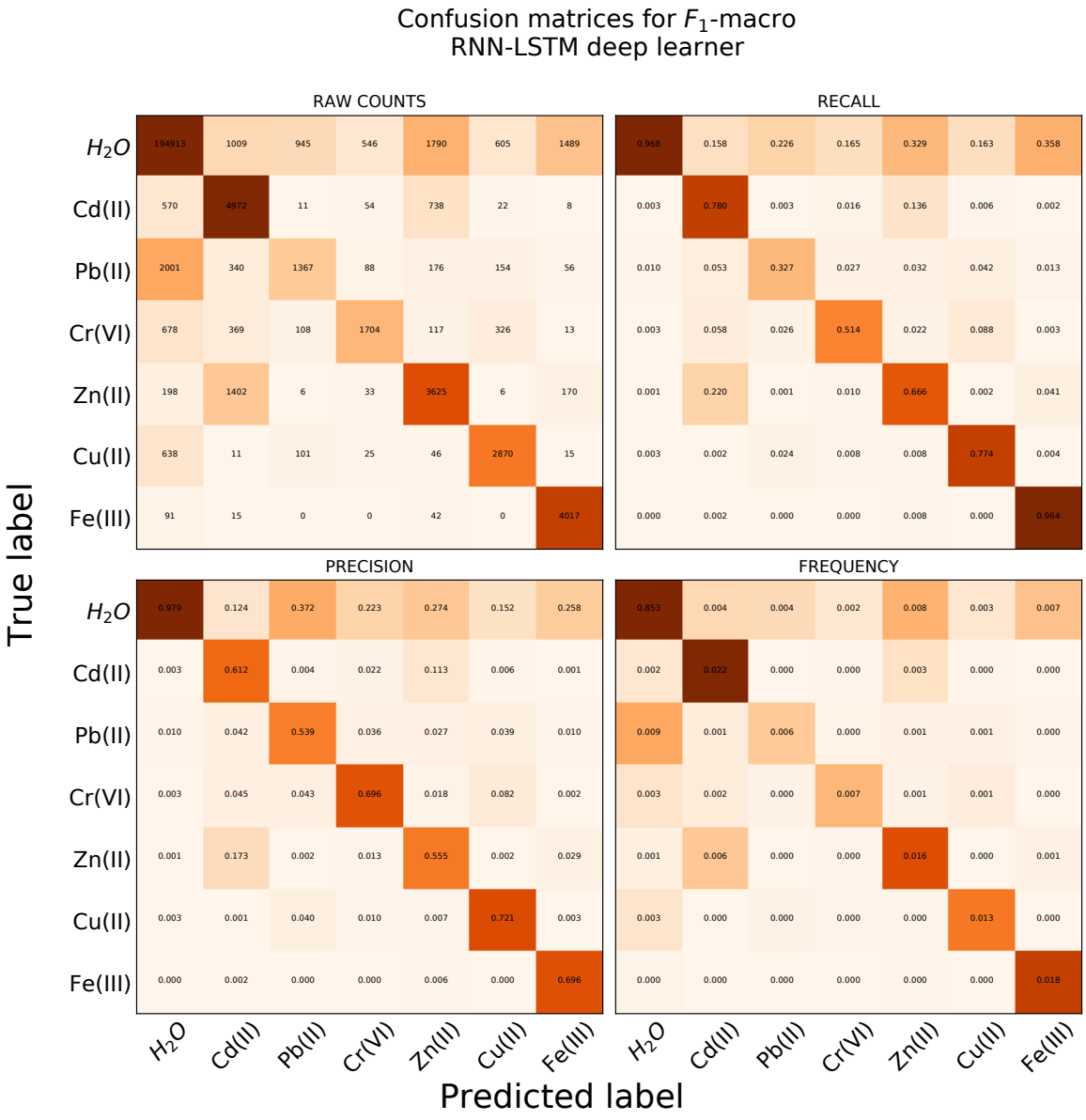


Figure 3.31: The classification results for an LSTM-RNN. These confusion matrices represent the cross-validation results of our optimized LSTM-RNN classifiers. Note that these matrices are plotted using a linear shading scale, rather than a log scale as in Fig. 3.30. The recall and precision, as compared with those of the best-case XGBoost classifiers, are dramatically improved. Although recall is not perfect, the LSTM-RNN rarely (at most 1% of the time) mistakes any metal for water when the metal is actually present, as can be observed in the left-hand column of the recall matrix. This result is an especially desirable one for a sensor: even if it is wrong about which metal is present, it is still detecting that something is contaminating the water. Similarly, the high rate of precision for water, at approximately 98%, is another desirable sensor quality, as it indicates a low false positive rate.

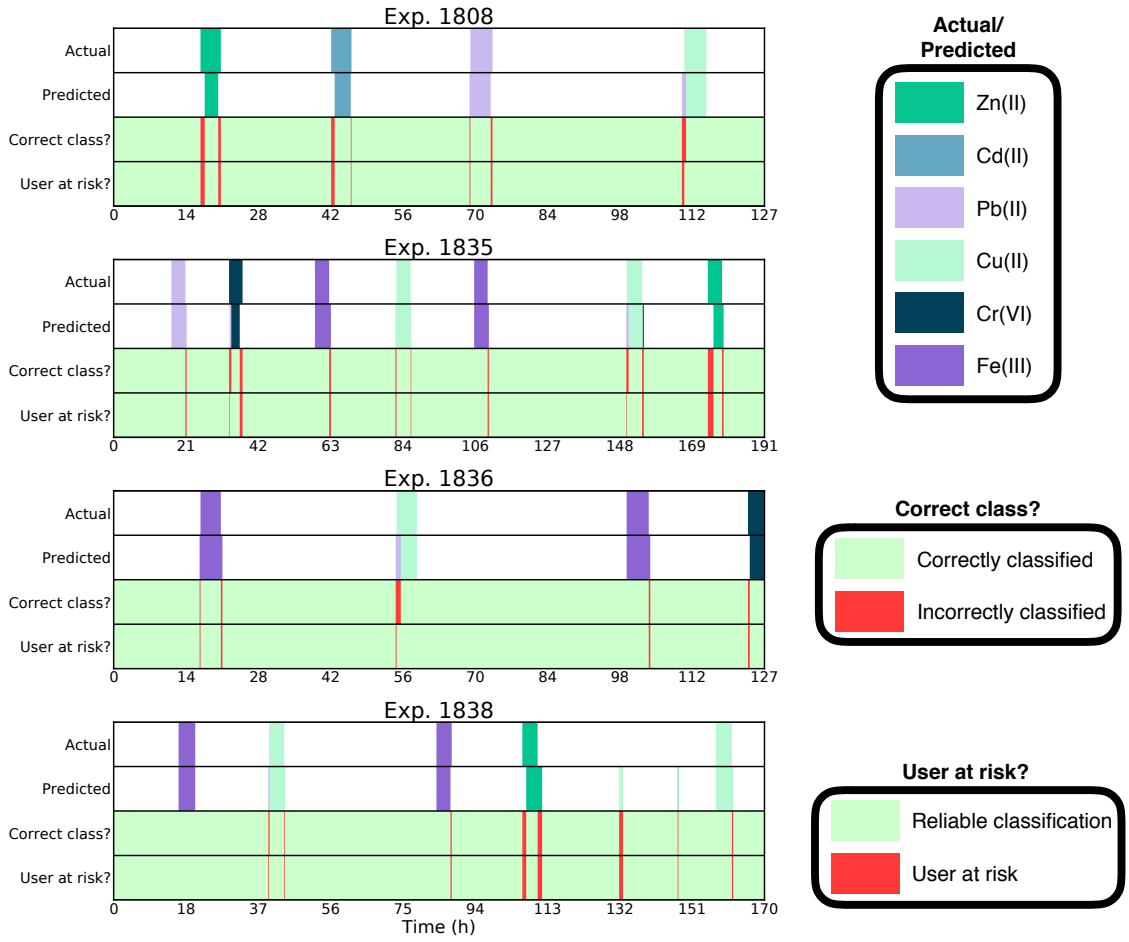


Figure 3.32: Deep learning network predictions on standardized experiment data as a function of time. These four experiments are a subset of the eighteen standardized experiments that made up the majority of our data. Each plot was produced by training an LSTM-RNN on all experiments except the depicted one, and then by testing the resulting classifier on the experiment in question.

predicting the correct metal, while a color mismatch indicates an incorrect prediction. The third row, ”Correct class?”, indicates time points with a correct prediction with green and incorrectly classified time points as red. Finally, the fourth row, which is labeled ”User at risk?”, indicates whether the classifier correct in predicting whether a point is water/not-water. For instance, in Exp. 1808, the classifier mislabels time points at the beginning and end of the first cadmium induction. If the concentration of cadmium were high, that mislabeling could potentially put someone relying on that water source temporarily at risk. Similarly, in Exp. 1838, we see that the classifier mislabels water as potentially copper later on in the experiment. While copper is not necessarily dangerous to humans, this false positive could cause a user to temporarily suspend their use of the water supply. A protracted false positive could, in the wrong situation, lead to a critical water shortage.

The recall scores for the extremely toxic metals cadmium, lead, and chromium are 78%, 33%, and 51%, respectively. A closer inspection of the LSTM-RNN's predictions as a function of time, depicted in Fig. 3.32 reveals that the harder-to-sense metals are never, in fact, not detected when presented to the sensor. Instead, the lower recall scores come from time points similar to those of the chromium induction in Experiment 1835, where the classifier has difficulty distinguishing the exact time when the transition from water to cadmium occurs and vice versa (Fig. 3.32). These results were consistent across all eighteen of the standardized experiments.

Classification of urban water samples

As a consequence of the varying water chemistries of different municipalities around the United States, it was by no means guaranteed that the standardized experiment-trained LSTM-RNN's performance would hold when made to predict on data from the urban water experiments. However, its performance once again was excellent (Fig. 3.33). Its only faulty predictions occurred when confusing lead in San Diego's water supply for copper. This result is possible due to its much higher concentrations of calcium carbonate solutes and dissolved solids, as compared to the other municipal water samples. Since many of these solutes are known to interact biologically, it is possible that they had some sort of impact upon the transcriptional profile of the *E. coli* genome in the presence of lead.

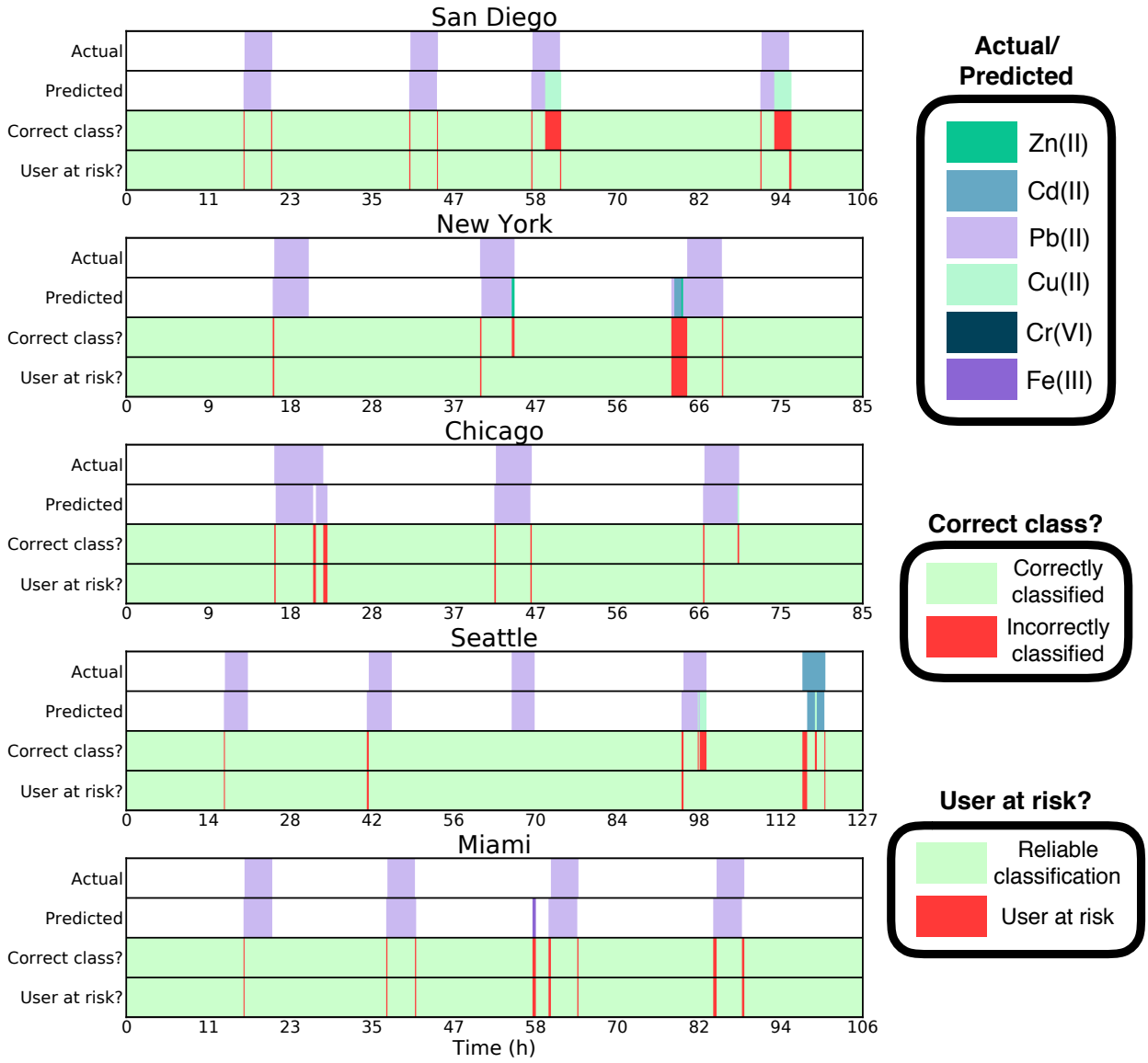


Figure 3.33: Deep learning network predictions on urban water samples as a function of time. Here we see the predictions of an LSTM-RNN that has been presented with the urban water sample experiments. Each experiment is labeled with the name of the city whose water the *E. coli* library is being grown on and to which we added lead and cadmium. These experiments were classified by the LSTM-RNNs depicted in Fig. 3.32. Each prediction represents the mean of eighteen cross-validation rounds, each round of which used a different of the standardized experiments as an early-stopping verification data set. Overall, the deep learners were proficient at detecting the presence of spiked-in lead contamination in these samples, despite the varying water chemistries. San Diego appeared to present the classifiers with the largest challenge, most like as a result of its high concentrations of calcium carbonates and dissolved solids. Even then, it still predicted the presence of a metal the vast majority of the time that lead was actually presence. These results suggest that the DynOMICs system would, at the very minimum, serve as an excellent binary water/not-water sensor.

Classification of San Juan River samples

The LSTM-RNN was then exposed to repeated inductions by samples of San Juan River water taken from the Gold King Mine Spill plume as it traveled through Utah. In these cases, however, the output layer of the network was swapped from a softmax function to a sigmoid function, effectively allowing the LSTM-RNN to become a multiclass, multilabel classifier (as opposed to only predicting a single class at each time point, as it had done with both standardized and urban water experiments).

The resulting probability predictions, as seen in Fig. 3.34, are in extremely close agreement with the concentrations that were determined by an ICP-MS analysis of the samples. In the figure, the probabilities for the metals begin to increase starting exactly one hour before we introduced the river water, which corresponds precisely to the LSTM-RNN's features' time window. Simultaneously, the probability of pure water being present begins to drop. Once the classifier has enough evidence, it slowly rapidly determines that iron is the most likely constituent in the sample. This prediction corresponds exactly with the chemical analysis of the water, which showed that iron's concentrations were approximately 100 times higher than the next most-ubiquitous metal, zinc. While these experiments do not indicate that the DynOMICs system, in its current state, could have been used as a real-time sensor in the days after the 2015 disaster, it still serves as a promising proof-of-concept.

3.3.8 Explainable artificial intelligence (XAI) with Shapley additive explanation values

XGBoost SHAP values

Once we had demonstrated the efficacy of the DynOMICs devices as heavy-metal sensors on a variety of water samples from the field, we explored its potential as a scientific instrument for studying the transcriptional dynamics of the *E. coli* genome. We did this by using a cutting edge explainable artificial intelligence (XAI) method known as Shapley additive explanatory values or SHAP [75]. The results were both affirming and revealing. We first examined the overall effect that

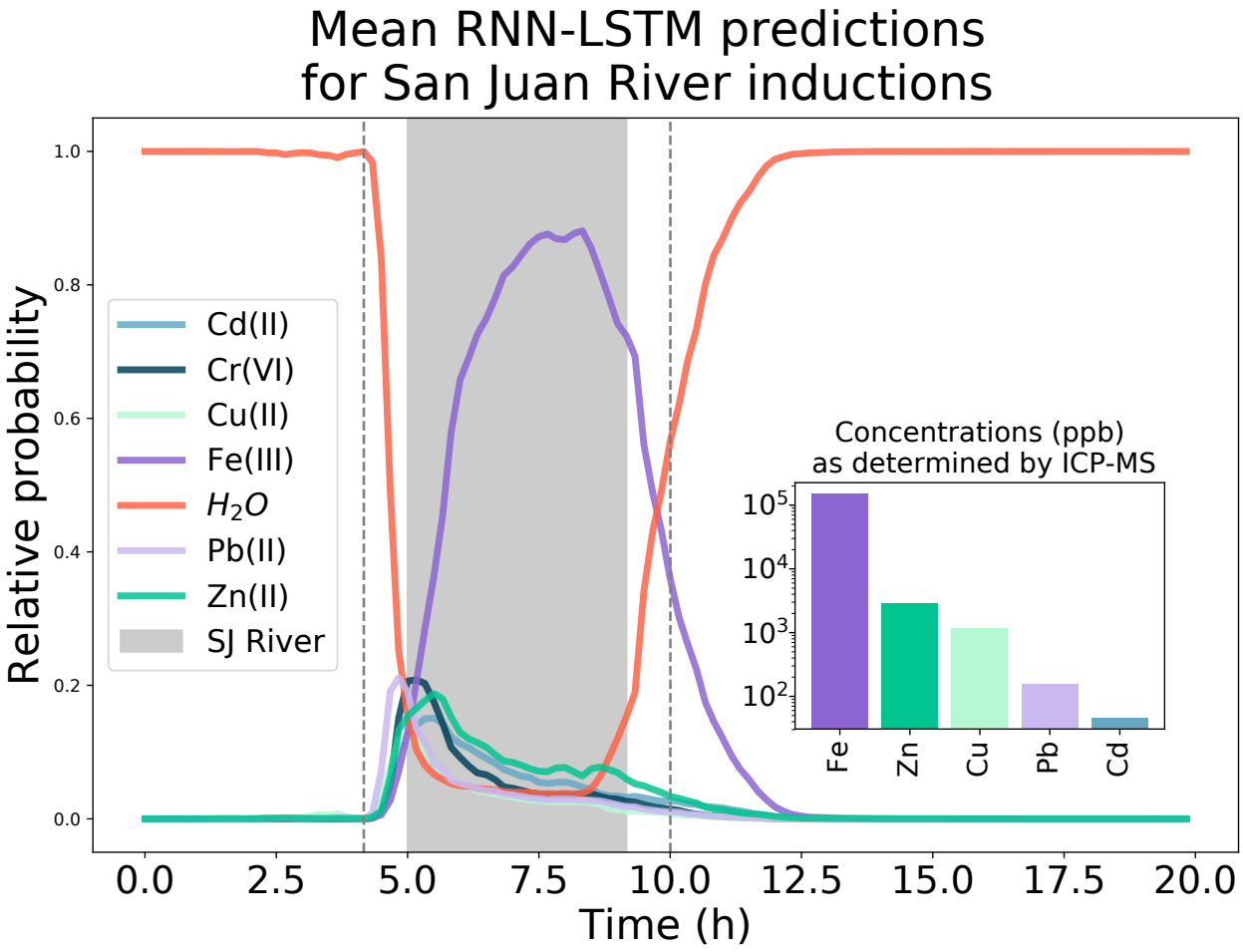


Figure 3.34: Deep learning network predictions on San Juan River samples containing contamination from the Gold King Mine Spill as a function of time. These trajectories represent the predictions of a multiclass, multilabel LSTM-RNN on samples, collected contemporaneously in the field, from the San Juan River that contain contaminants from the 2015 Gold King Mine Spill. Created by averaging together the predictions from three different inductions over three experiments, we see the relative probabilities change in a logical and intuitive way. As the probability of water decreases, the probabilities of all other metals rise. This process continues until the classifier has enough information to begin distinguishing which metals it believes are more likely to be present (rather than simply predicting not-water). Comparison with the concentrations in the inset bar graph, which are the actual concentrations of these samples as determined via analytical chemical analysis, shows that the classifier appears to predict the presence of iron correctly. While its predictions for the presence of zinc and cadmium are far less confident, they still are greater than water's probability. In order to actually distinguish itself as a field-worthy multilabel sensor, the DynOMICs system would need to undergo further development. However, this result is an encouraging preliminary finding.

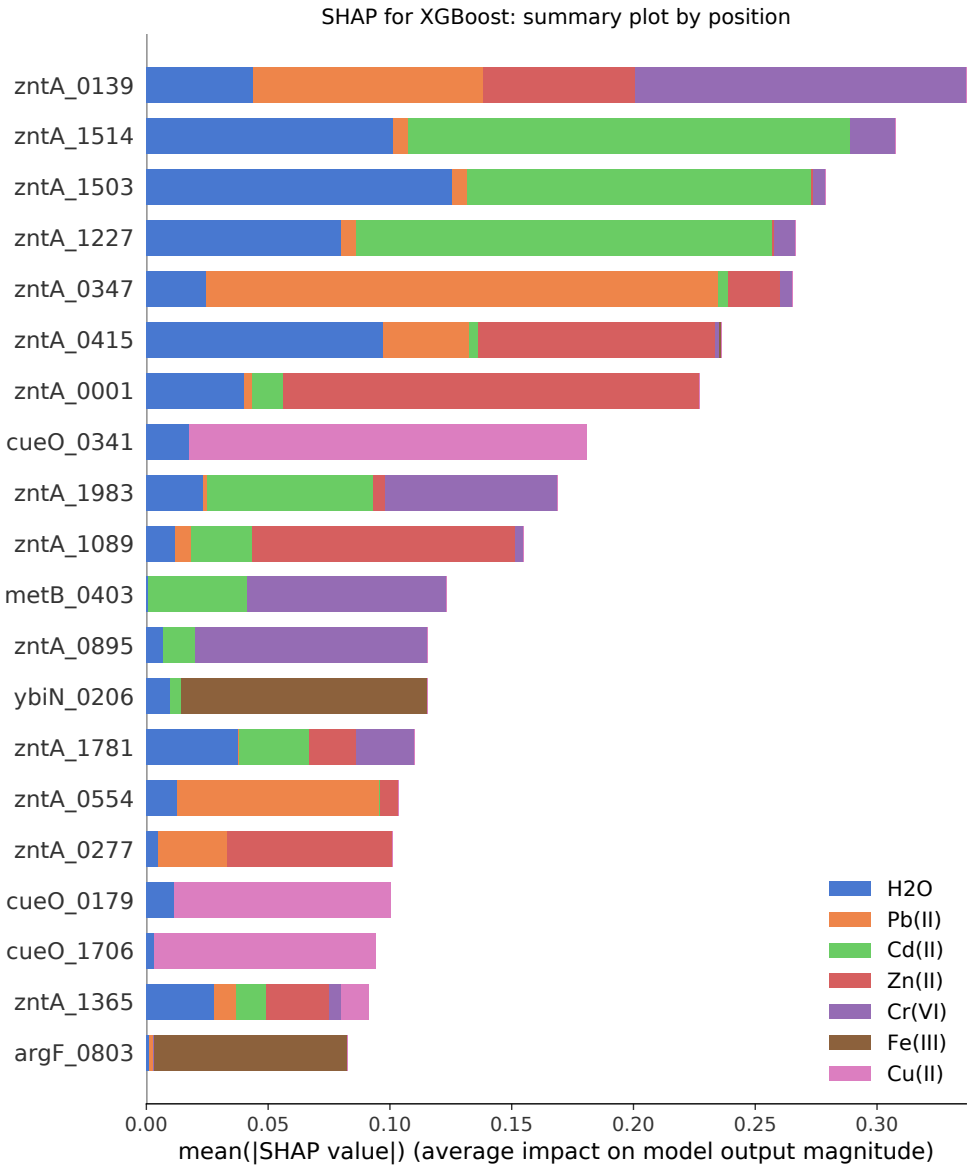


Figure 3.35: A SHAP XAI's summary of the XGBoost feature importances by device position. In this figure, we see a succinct summary of a SHAP learner's findings with regards to how XGBoost would decide to classify various classes. Each distinct row along the vertical axis is non-empty device position, ranked in descending order from most impactful to least. The names are a concatenation of the strain name and the device position number; ie, "zntA_0139" is the *zntA* strain located in device position 139. Each color represents a different class' mean of the absolute value of the SHAP scores for that specific strain. The purpose of taking the mean of the absolute value of the SHAP scores is that some strains have large negative SHAP scores, indicating that they contribute strongly to predicting the absence, rather than the presence, of specific classes. This plot acts as an excellent sanity-check, since we see that the most of the listed strains are either *zntA* or *cueO*, both of which were reliable multi-metal- and copper-responsive strains, respectively. Already, though, we begin seeing several previously-unknown strains upon which the classifier relied to predict a variety of metals, including iron, cadmium, and chromium.

each kind of classifier attributed to each feature. We did this both by position, where we examined the contribution of each non-empty device position across all experiments, and by strain, where we took the mean SHAP values of any redundant strains' positions.

The summary by device position served as an excellent sanity-check (Fig. 3.35). The first seven most important positions are filled by redundant *zntA* positions, which is known to show a strong response to both cadmium and zinc, while also demonstrating a weaker response to a variety of the other metals. This kind of response profile made the strain our most reliable multi-metal indicator. The eighth position of the top twenty strains is occupied by a member of the *cueO* positions. *cueO* is the only promoter from the copper-sensing *cue* operon in the Alon GFP library. Thus, it is logical that the *cueO* positions present in the plot are shown to be overwhelming contributors when it comes to copper detection.

Although these kinds of reliable metal sensing strains dominate the by-position SHAP rankings, already we can see new strains whose designated sensitivities we had yet to notice. *metB* and *argF* showed up throughout these SHAP analyses, for both classifiers, as sensitive to chromium and iron, respectively (Fig. 3.35).

However, comparing the SHAP values on a strain-by-strain basis, rather than a position-by-position basis, makes it easier to notice strains with more subtle responses (Fig. 3.36). Upon examining the same form of summary plot, we see that *zntA* is indeed chosen by the classifier as an excellent multi-metal sensor. *metB* rises to the rank of second most-impactful strain, as a result of its sensitivities to cadmium and chromium. Notice, too, that two lead-sensitive strains are immediately obvious. This result was surprising because of the difficulty with which XGBoost classified that metal. It also suggests that SHAP is a potentially excellent method by which to perform feature selection, in order to build a better heavy metal sensor.

LSTM-RNN SHAP values

In order to compare the observations of the two types of classifiers, we trained a separate SHAP learner on the finished standardized experiments' LSTM-RNN. The by-position summariza-

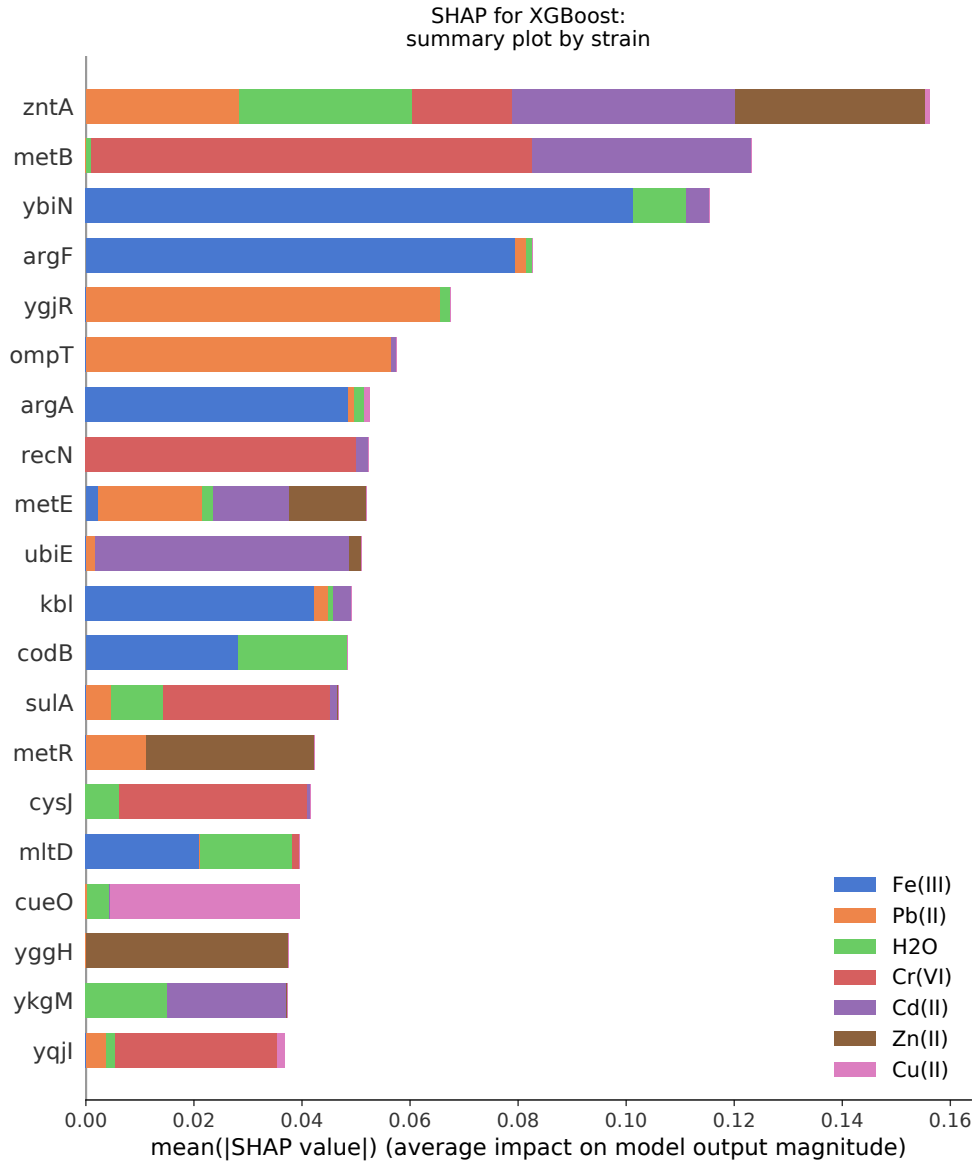


Figure 3.36: A SHAP XAI's summary of the XGBoost feature importances by strain. In contrast to Fig. 3.35, this figure depicts a SHAP learner's analysis of the XGBoost classifier's feature attributions by individual strains. To accomplish this, each redundant strain's various positions were averaged together in a time point-by-time point way. Doing so reduces the clutter caused by redundant strain positions and allow us to analyze the contributions of strains that inhabit single positions within the microfluidic device. While several of the metal sensing strains remain, such as *zntA* and *cueO*, other cellular processes begin to be made visible. For instance, many promoters of the *met* operon contribute to sensing lead, zinc, chromium, and cadmium. Similarly, a promoter from the *arg* operon is relied upon. Both of these operons are responsible for synthesizing essential amino acids (methionine and arginine, respectively), which heavy metals are known to deleteriously chelate.

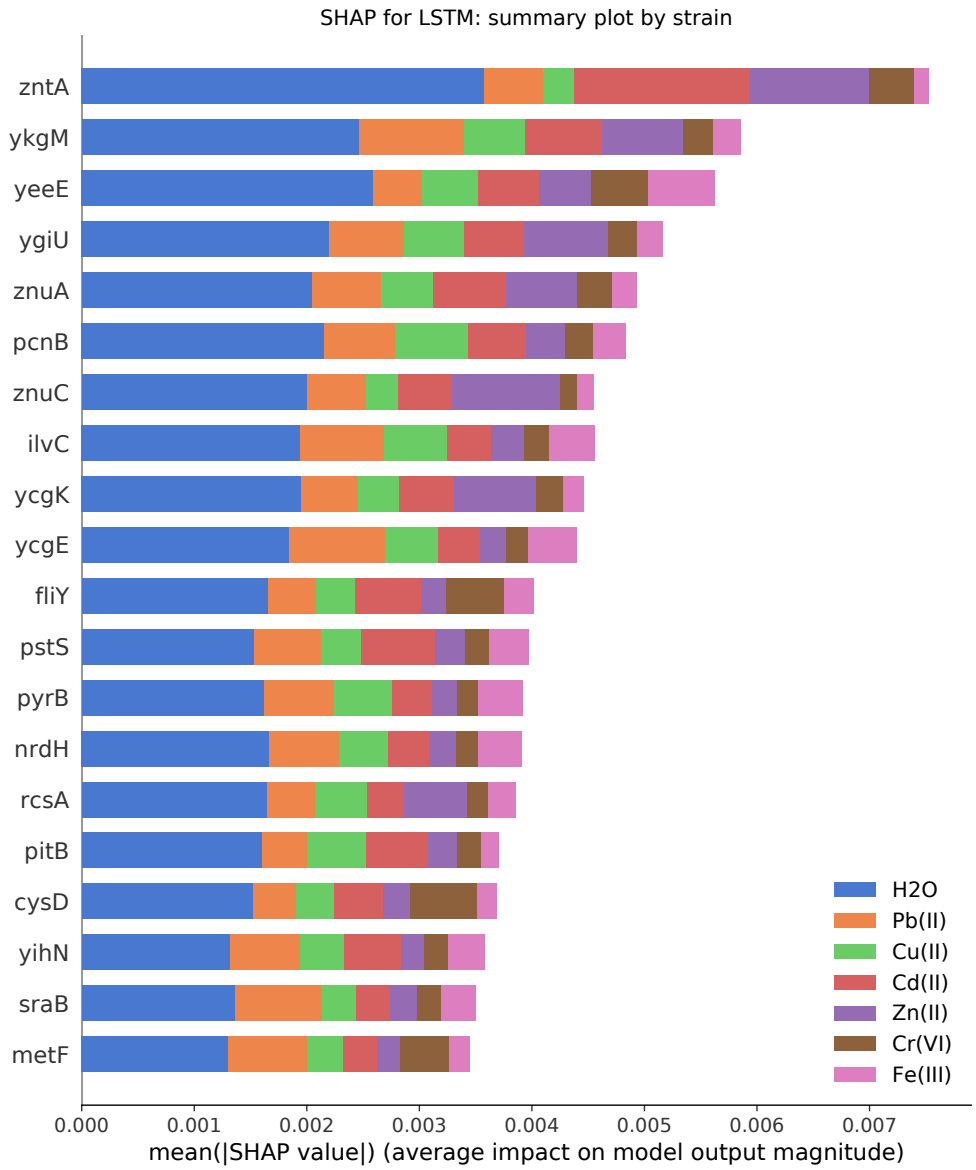


Figure 3.37: A SHAP XAI's summary of the LSTM-RNN feature importances by strain. The XAI summary depicted here is of the features with the most impact upon the prediction process of an LSTM-RNN learner. While only sharing the strains *zntA* and *ykgM* in common with the XGBoost learner, both classifiers nevertheless rely upon some of the same operons, including the *met* and *cys* operons. Additionally, these summary plots seem to suggest that the two different classifiers take different paths to reaching the same prediction decisions. While XGBoost appears to rely more heavily on single-metal responders, the deep learner appears to prefer using strains whose predictive abilities are distributed evenly across many of the classes.

tion results were similar those of the XGBoost learner, but with an even heavier reliance on the redundant *zntA* positions. However, the by-strain analysis was remarkably different, with an almost completely non-overlapping set of the twenty most-impactful strains (Fig. 3.37). A closer inspection of the raw signals of many of these strains showed that they were indeed sensitive to their attributed metals. Frequently, though, the experimental data was so noisy that their responses were not clearly visible. It seems that the deep learner was quite capable of cutting through the noise of the data set to find truly responsive strains. LSTM-RNNs are well-known for this ability and are favored in many problems where the researcher may not have enough domain knowledge to engineer meaningful features.

One of the stranger results from this analysis was that *cueO* did not appear in this list, as we would have intuitively expected it to do (recall its inclusion in Fig. 3.36). To understand why this occurred, we checked if *cueO* was used in deciding whether copper were present or not. The LSTM-RNN indeed relied heavily upon the strain in making decisions on copper, as was evidenced by its sudden accumulation of large SHAP values at the beginning of the copper inductions, followed by a loss of the same values after the induction ended. What appeared to have been happening instead is that the LSTM-RNN was discounting *cueO*'s usefulness because of its specificity with regards to copper. Note that, while some strains that the deep learner considers the most useful have oversized utilities for specific classes (such as *zntA* for water and cadmium), the majority of these strains have fairly equally distributed impact attribution values across classes. It is probably that *cueO*'s lack of sensitivity to other metals, as noted by the XGBoost learner in Fig. 3.36, rendered it less useful to the LSTM-RNN's decision style, which seems to prefer equally-weighted inputs from many multi-metal-sensitive strains (Fig. 3.37).

Comparison between the two

Delving deeper into how each learner made decisions on a time point-by-time point basis revealed novel insights into both *E. coli* transcriptional dynamics in response to heavy metal stress and how each of these learning algorithms made its decision.

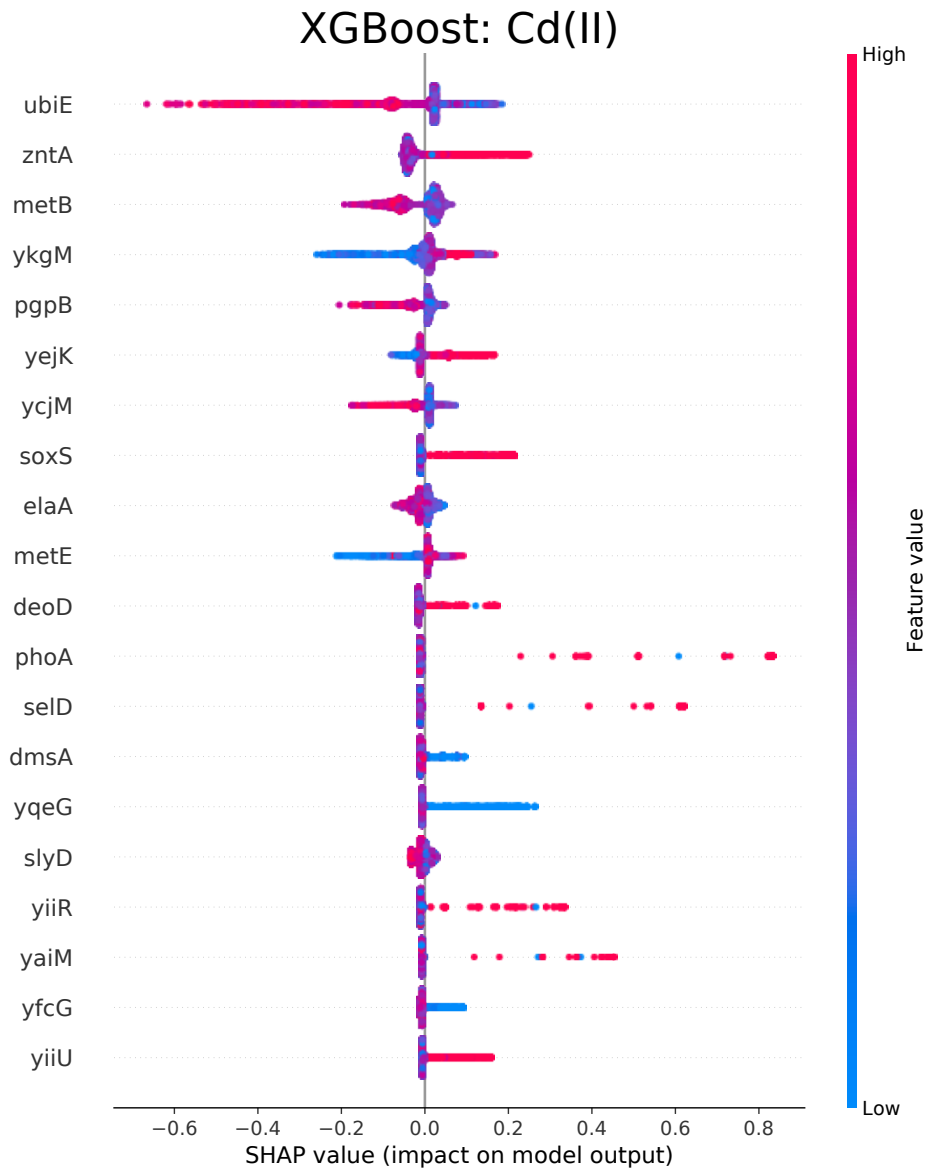


Figure 3.38: A SHAP XAI's summary of the XGBoost feature importances for cadmium detection. In addition to all-class summary plots such as Fig. 3.37, we examined more detailed summary plots on a class-by-class basis. Here we see the SHAP XAI's calculated strain impacts on the detection of cadmium for our XGBoost learner . These plots allow us to see which feature values in particular contribute to which cadmium-related decisions. For instance, the tree ensemble learner interprets higher first derivative feature values for *zntA*, as denoted by the red points in the second row from the top, as indicating that cadmium is more likely present in the strains' growth media. Conversely, the strain in the upper row, *ubiE*, displays an entirely different kind of impact on the model. Instead of higher derivative values implying cadmium's presence, they actually inform the classifier of cadmium's probable absence from the media; low values for the same strain's features make it likely that cadmium is indeed present. This kind of information demonstrates how DynOMICs is a unique scientific instrument, capable of capturing transcriptional dynamics data that few, if any, other technologies could gather with the same resolution.

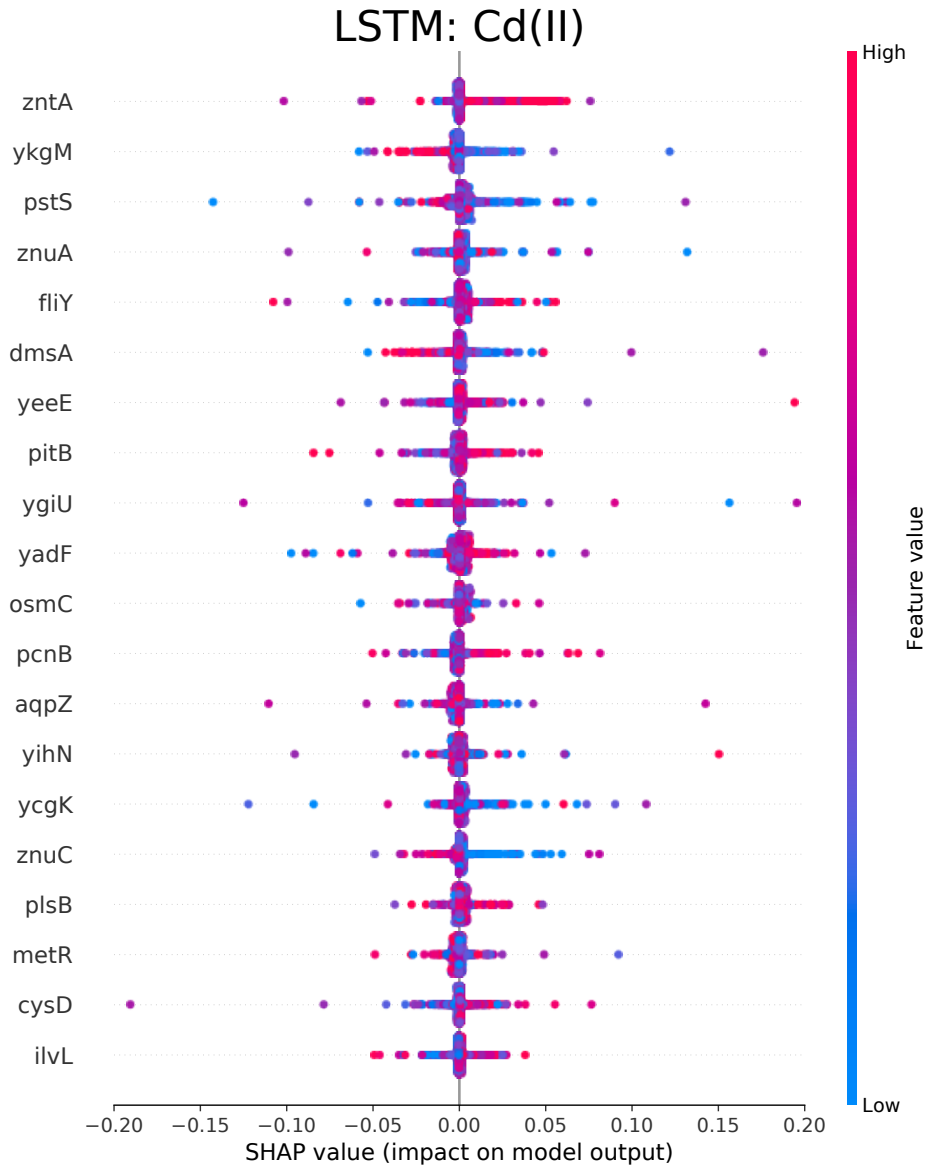


Figure 3.39: An LSTM-RNN SHAP XAI's attribution plots for cadmium predictions. This plot, the LSTM-RNN equivalent to Fig. 3.38, reveals the remarkably different paths that the two kinds of algorithms can take to reaching the same detection conclusions. While only sharing three strains in common (*zntA*, *dmsA*, and *ykgM*), they actually appear to rely on genes with similar cellular functions. For instance, the *znu* operon, which the LSTM-RNN learner relies on the determine cadmium's presence/absence, is involved in the same DNA repair and stress-induced mutagenesis pathways as *ykgM* [11]. Thus, it is plausible that these AI are learning two different dynamic representations of the same cellular processes.

Cadmium served as an excellent starting point, since it was the metal with the highest recall scores for the LSTM-RNN learner (Fig. 3.31). In addition, it produced interesting results from both learners. In the case of the XGBoost SHAP learner, it found that previously unnoticed strain, *ubiE*, was the most impactful overall in predicting cadmium's presence and absence (Fig. 3.38). We were surprised by this result, given *zntA*'s extremely consistent responses to the metal. Upon closer inspection, we observed that *ubiE* acted as a negative reporter: time points with higher derivative values (indicated by the red dots in Fig. 3.38) lessened the likelihood of cadmium being present, while a low derivative value increased the likelihood of predicting cadmium. In biological terms, *ubiE* would exhibit dynamic transcriptional behavior in the absence of cadmium, but then cease transcriptional activity in its presence. This result is notable in that it most likely would not be found with scientific instruments other than the DynOMICs system, since finding it relies purely on detecting changes in the statistical characteristics of that gene's transcriptional dynamics.

Comparing the LSTM-RNN's SHAP learner to that of XGBoost's showed large differences between the two, at least with respect to which strains they considered the most impactful. In both cases, the XAI learner's monitored *zntA* closely; however, their similarities did not extend far beyond that strain. Except for two other genes (*dmsA* and *ykgM*), each XAI relied upon a completely disjoint set of seventeen strains of their twenty most-impactful in order to detect cadmium. A closer inspection of the functions of the two disjoint sets, though, demonstrated some interesting links between the two. For instance, the LSTM-RNN relied upon multiple genes from the *znu* operon for its decisions. Multiple studies have observed the *znu* operon and *ykgM* to be closely involved in DNA repair and mutagenesis [11] [65]. In addition, the *znu* operon In addition to these similarities, both XAI observed that the XGBoost learner and the LSTM-RNN learned each utilized promoters from the *met* operon. The *met* operon is known to be up-regulated in under cadmium stress as a result of the cadmium-induced amino acid limitations within the cell [37]. Anecdotally, methionine has been studied extensively as a treatment for cadmium poisoning in humans and animals; it is thought to be an effective treatment because of its ability to effectively chelate unbound cadmium [46] [36] [85]

Thus, while the XAI reveal large differences in how each of the machine learning classifiers come to their respective conclusions in the classification of cadmium, the differences are not as large as they first appear. The known functional relationships between the two promoter sets indicate that the machine learning classifiers, rather than learning specific transcriptional behaviors, may be learning two different *representations* of the coordinated cellular functions associated with response to cadmium stress. That they seem to be recognizing the same phenomena from two different perspectives is promising, as it potentially provides two different avenues by which to observe transcriptional dynamics.

3.4 Conclusion

Machine learning has long held promise to help untangle and understand the intricate data arising from biological experiments, especially those studies that examine noisy gene expression data. However, until recently there has not been a straightforward and universal method by which to introspect and understand the process by which an artificial intelligence learns and recognizes meaningful patterns in these data. This lack has been felt more acutely in studies that utilize more complex and powerful classifiers, such as deep neural networks. The advent of explainable artificial intelligence algorithms, which are able to learn from another AI how that other AI recognizes the patterns that it does, has begun to shift that paradigm. Instead of having to choose between a powerfully predictive model and an interpretable one, now scientists can have both. In the DynOMICs project, we see both AI and XAI combined with genomic-scale microfluidics to address the problem of real-time heavy metal sensing and as a method to study the transcriptional dynamics of the *E. coli* genome. To our knowledge, this project represents the first time anyone in our field have simultaneously brought all of these tools to bear on either of these problems, much less than on both of them simultaneously. DynOMICs has already demonstrated its usefulness as an in-line, real-time heavy metal sensor. While the system is still in its infancy with regards to scientific instrumentation, it has already uncovered novel dynamic transcriptional behavior and shown that it can be used to

discover relationships between related gene networks in the *E. coli* genome. Moving forward, we look forward to pursuing these analyses via further analysis of the XAI results.

3.5 Acknowledgements

This chapter, in part, is currently being prepared for submission of the material by with the following individuals: Graham, Garrett; Csicsery, Nicholas; Stasiowski, Elizabeth; Thouvenin, Gregoire; Hasty, Jeff. The dissertation author was the primary researcher and author of this material.

Bibliography

[1] Python Generators. *Python Wiki*.

[2] sklearn.ensemble.RandomForestClassifier.

[3] Navajo Crops Drying Out as San Juan River Remains Closed After Toxic Spill — Forgotten People, 2015.

[4] Interpretable ML Symposium - NIPS 2017, 2017.

[5] Gold King Mine spills spurs 5th lawsuit filed against EPA over 2015 waste spill, 8 2018.

[6] Rensink R A., O'Regan J K., and Clark J J. To See or Not to See: The Need for Attention to Perceive Changes in Scenes. *Psychological Science*, 8(5):368–373, 1997.

[7] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[8] Asma Rokbani Achour, Pascale Bauda, and Patrick Billard. Diversity of arsenite transporter genes from arsenic-resistant soil bacteria. *Research in Microbiology*, 158(2):128–137, 3 2007.

[9] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[10] Ata Akcil and Soner Koldas. Acid Mine Drainage (AMD): causes, treatment and case studies. *Journal of Cleaner Production*, 14(12-13 SPEC. ISS.):1139–1145, 2006.

[11] Abu Amar, M Al Mamun, Mary-jane Lombardo, Chandan Shee, Andreas M Lisewski, Caleb Gonzalez, Dongxu Lin, Ralf B Nehring, Claude Saint-ruf, Janet L Gibson, Ryan L Frisch, Olivier Lichtarge, P J Hastings, and Susan M Rosenberg. Identity and Function of a Large Gene Network Underlying Mutagenic Repair of DNA Breaks. *Science*, 338(December):1344–1348, 2012.

[12] Afshine Amidi and Shervine Amidi. A detailed example of data generators with Keras, 2018.

[13] Kai Arulkumaran, Antoine Cully, and Julian Togelius. AlphaStar: An Evolutionary Computation Perspective. pages 3–4, 2019.

[14] Ziv Bar-Joseph, Anthony Gitter, and Itamar Simon. Studying and modelling dynamic biological processes using time-series gene expression data. *Nature Reviews Genetics*, 13(8):552–564, 2012.

[15] Tamar Barkay, Susan M. Miller, and Anne O. Summers. Bacterial mercury resistance from atoms to ecosystems. *FEMS Microbiology Reviews*, 27(2-3):355–384, 2003.

[16] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[17] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. 13:1–25, 2012.

[18] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D. Cox. Hyperopt: A Python library for model selection and hyperparameter optimization. *Computational Science and Discovery*, 8(1), 2015.

[19] Mike Bernico. *Deep Learning Quick Reference*. O'Reilly, 2018.

[20] Marie R.B Binet and Robert K Poole. Cd(II), Pb(II) and Zn(II) ions regulate expression of the metal-transporting P-type ATPase ZntA in ¡i¿Escherichia coli¡/i¿. *FEBS Letters*, 473(1):67–70, 5 2000.

[21] Ptacek C. J. Jambor J. L. Weisener C. G. Blowes, D. W. Treatise on Geochemistry: The Geochemistry of Acid Mine Drainage. pages 149–204, 2003.

[22] Leo Breiman and Adele Cutler. Random forests Classification description: Random forests, 2007.

[23] Matthew Brown. 50M gallons of polluted water pours daily from US mine sites, 2 2019.

[24] Diogo M. Camacho, Katherine M. Collins, Rani K. Powers, James C. Costello, and James J. Collins. Next-Generation Machine Learning for Biological Networks. *Cell*, 173(7):1581–1592, 2018.

[25] Gavin C. Cawley and Nicola L. Talbot. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

[26] S. Cerminati, F. C. Soncini, and S. K. Checa. A sensitive whole-cell biosensor for the simultaneous detection of a broad-spectrum of toxic heavy metal ions. *Chemical Communications*, 51(27):5917–5920, 3 2015.

[27] Pete Chandrangsu, Christopher Rensing, and John D. Helmann. Metal homeostasis and resistance in bacteria. *Nature Reviews Microbiology*, 15(6):338–350, 6 2017.

[28] Karletta Chief, Janick F Artiola, Sarah T Wilkinson, Paloma Beamer, and Raina M Maier. Understanding the Gold King Mine Spill. *Fact Sheet*, 2015.

[29] Franois Chollet and others. Keras. \url{https://keras.io}, 2015.

[30] Marc Claesen and Bart De Moor. Hyperparameter Search in Machine Learning. pages 10–14, 2015.

[31] James J Collins, Timothy S Gardner, and Charles R Cantor. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403(6767):339–342, 2000.

[32] Nicolas Dénervaud, Johannes Becker, Ricard Delgado-Gonzalo, Pascal Damay, Arun S Rajkumar, Michael Unser, David Shore, Felix Naef, and Sebastian J Maerkl. A chemostat array enables the spatio-temporal analysis of the yeast proteome. 110(39), 2013.

[33] Ian Dewancker, Michael McCourt, and Scott Clark. Bayesian Optimization Primer. pages 2–5, 2015.

[34] Paul Driessen. EPAs gross negligence at Gold King — NMPolitics.net, 2015.

[35] M B Elowitz and S Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–8, 1 2000.

[36] Anete Corrła Esteves and Judith Felcman. Study of the Effect of the Administration of Cd(II), Cysteine, Methionine, and Cd(II) Together with Cysteine or Methionine on the Conversion of Xanthine Dehydrogenase into Xanthine Oxidase. *Biological Trace Element Research*, 76(1):19–30, 2000.

[37] P. Ferianc and A. Farewell. The cadmium-stress stimulon of Escherichia coli K-12. *Microbiology*, 144(4):1045–1050, 4 1998.

[38] Jens Flemming. Tikhonov regularization. In *Frontiers in Mathematics*, pages 55–57. 2018.

[39] Scott Fortmann-Roe. Understanding the Bias-Variance Tradeoff.

[40] Peter I. Frazier. A Tutorial on Bayesian Optimization. (Section 5):1–22, 2018.

[41] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[42] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[43] Garrett Graham. GitHub Lubansky Differentiation Repository.

[44] Casey S. Greene, Jie Tan, Matthew Ung, Jason H. Moore, and Chao Cheng. Big Data Bioinformatics. *Journal of Cellular Physiology*, 229(12):1896–1900, 12 2014.

[45] Denis Gris. Understanding Modularity in Molecular Networks Requires Dynamics. *Science Signaling*, 2(81), 2009.

[46] U. Gubrelay, P. Srivastava, R. Mathur, N. Tripathi, and S. J.S. Flora. Effects of thiamin and methionine administration in preventing cadmium-induced biochemical alterations and metal concentration in male rats. *Journal of Trace Elements in Medicine and Biology*, 12(2):86–90, 1998.

[47] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[48] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995.

[49] Karen Hao. AI is sending people to jailand getting it wrong, 2019.

[50] Nan Hao and Erin K. O'Shea. Signal-dependent dynamics of transcription factor translocation controls gene expression. *Nature Structural and Molecular Biology*, 19(1):31–40, 2012.

[51] Joe J. Harrison, Howard Ceri, and Raymond J. Turner. Multimetal resistance and tolerance in microbial biofilms. *Nature Reviews Microbiology*, 5(12):928–938, 2007.

[52] Michiel Hazewinkel. Shapley value, 2001.

[53] Sepp Hochreiter and Jrgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[54] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification. 2003.

[55] Karen Hudson-Edwards. Tackling mine wastes. *Science*, 352(6283):288–290, 2016.

[56] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. *Machine Learning and Knowledge Discovery in Databases*, (Chapter 40):507–523, 2011.

[57] Aarshay Jain. Complete Guide to Parameter Tuning in XGBoost (with codes in Python).

[58] Lars Järup. Hazards of heavy metal contamination. *British medical bulletin*, 68:167–82, 2003.

[59] Andrej Karpathy. Convolutional Neural Networks for Visual Recognition. *GitHub*.

[60] Hyun Ju Kim, Haeyoung Jeong, and Sang Jun Lee. Synthetic biology for microbial heavy metal biosensors. *Analytical and Bioanalytical Chemistry*, 410(4):1191–1203, 2 2018.

[61] Minseok Kim, Ji Won Lim, Hyun Ju Kim, Sung Kuk Lee, Sang Jun Lee, and Taesung Kim. Chemostat-like microfluidic platform for highly sensitive detection of heavy metal ions using microbial biosensors. *Biosensors and Bioelectronics*, 65:257–264, 3 2015.

[62] Hiroaki Kitano and Hiroaki Kitano. Systems Biology : A Brief Overview. *Science*, 295(5560):1662–1664, 2017.

[63] Will Knight. The Dark Secret at the Heart of AI, 2017.

[64] R Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, 2(0):1137–1143, 1995.

[65] Kosmas Kosmidis and Marc Thorsten Hütt. The E. coli transcriptional regulatory network and its spatial embedding. *European Physical Journal E*, 42(3), 2019.

[66] Martin Krasser. Bayesian optimization.

[67] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. pages 1378–1387, 6 2015.

[68] Joseph J. LaViola. Double exponential smoothing, 2004.

[69] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[70] Ovidiu Lipan and Wing H Wong. The use of oscillatory signals in the study of genetic networks. *PNAS*, 2005.

[71] Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal Thresholding of Classifiers to Maximize F1 Measure. *Machine learning and knowledge discovery in databases : European Conference, ECML PKDD ... : proceedings. ECML PKDD (Conference)*, 8725:225–239, 2014.

[72] A. S. Lubansky, Y. Leong Yeow, Yee Kwong Leong, S. Ranil Wickramasinghe, and Binbing Han. A general method of computing the derivative of experimental data. *AIChE Journal*, 52(1):323–332, 2006.

[73] George Luka, Ali Ahmadi, Homayoun Najjaran, Evangelyn Alocilja, Maria DeRosa, Kirsten Wolthers, Ahmed Malki, Hassan Aziz, Asmaa Althani, Mina Hoorfar, George Luka, Ali Ahmadi, Homayoun Najjaran, Evangelyn Alocilja, Maria DeRosa, Kirsten Wolthers, Ahmed Malki, Hassan Aziz, Asmaa Althani, and Mina Hoorfar. Microfluidics Integrated Biosensors: A Leading Technology towards Lab-on-a-Chip and Sensing Applications. *Sensors*, 15(12):30011–30031, 12 2015.

[74] Scott Lundberg. shap.

[75] Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. (Section 2):1–10, 2017.

[76] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. Explainable AI for Trees: From Local Explanations to Global Understanding. *arXiv preprint arXiv:1905.04610*, 2019.

[77] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-wai Low, Shu-fang Newman, Jerry Kim, and Su-in Lee. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(October):749760, 2018.

[78] Jianzhu Ma, Michael Ku Yu, Samson Fong, Keiichiro Ono, Eric Sage, Barry Demchak, Roded Sharan, and Trey Ideker. Using deep learning to model the hierarchical structure and function of a cell. *Nature Methods*, 15(4):290–298, 2018.

[79] Zhen Ma, Faith E Jacobsen, and David P Giedroc. Coordination chemistry of bacterial metal transport and sensing. *Chemical reviews*, 109(10):4644–81, 10 2009.

[80] D. Mandic and J. Chambers. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Wiley, 2001.

[81] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[82] J. Mockus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer Berlin Heidelberg, 1975.

[83] Jonas Mockus, William Eddy, and Gintaras Reklaitis. *Bayesian Heuristic approach to discrete and global optimization: Algorithms, visualization, software, and applications*, volume 17. Springer Science & Business Media, 2013.

[84] Christopher Olah. Understanding LSTM Networks, 2015.

[85] Lyn Patrick. Toxic metals and antioxidants: Part II. The role of antioxidants in arsenic and cadmium toxicity. *Alternative medicine review : a journal of clinical therapeutic*, 8(2):106–28, 5 2003.

[86] Michael B. Pell and Joshua Schneyer. Reuters finds lead levels higher than Flint's in thousands of locales., 2016.

[87] Arthur Prindle, Phillip Samayoa, Ivan Razinkov, Tal Danino, Lev S Tsimring, and Jeff Hasty. A sensing array of radically coupled genetic 'biopixels'. *Nature*, 481(7379):39–44, 1 2012.

[88] Philipp Probst, Bernd Bischl, and Anne-Laure Boulesteix. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. 20:1–32, 2018.

[89] Teresa M. Przytycka, Mona Singh, and Donna K. Slonim. Toward the dynamic interactome: It's about time. *Briefings in Bioinformatics*, 11(1):15–29, 2010.

[90] Kanwal Rehman, Fiza Fatima, Iqra Waheed, and Muhammad Sajid Hamid Akash. Prevalence of exposure of heavy metals and their impact on health consequences. *Journal of Cellular Biochemistry*, 119(1):157–184, 2018.

[91] Alex Rogozhnikov. Gradient Boosting explained [demonstration].

[92] Perri Zeitz Ruckart, Adrienne S. Ettinger, Mona Hanna-Attisha, Nicole Jones, Stephanie I. Davis, and Patrick N. Breysse. The Flint Water Crisis: A Coordinated Public Health Emergency Response and Recovery Initiative. *Journal of Public Health Management and Practice*, 25(January 2016):S84–S90, 2019.

[93] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[94] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[95] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[96] David L. Shis, Matthew R. Bennett, and Oleg A. Igoshin. Dynamics of Bacterial Gene Regulatory Networks. *Annual Review of Biophysics*, 47(1):447–467, 2018.

[97] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. 2017.

[98] Sayantani Sikdar and Madhusree Kundu. A Review on Detection and Abatement of Heavy Metals. *ChemBioEng Reviews*, 5(1):18–29, 2018.

[99] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 10 2017.

[100] Christina G. Siontorou, Spyridoula Bratakou, Stephanos Karapetis, and Nikolaos Tzamtzis. Biosensors Based on Microfluidic Devices Lab-on-a-Chip and Microfluidic Technology. *Nanotechnology and Biosensors*, pages 375–394, 1 2018.

[101] David H Smith. R factors mediate resistance to mercury, nickel, and cobalt. *Science*, 156(3778):1114–1116, 1967.

[102] Mitch Smith, Julie Bosman, and Monica Davey. Flints Water Crisis Started 5 Years Ago. Its Not Over., 4 2019.

[103] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in neural information processing systems*, pages 2951–2959, 2012.

[104] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 7 2009.

[105] Zachary D. Stephens, Skylar Y. Lee, Faraz Faghri, Roy H. Campbell, Chengxiang Zhai, Miles J. Efron, Ravishankar Iyer, Michael C. Schatz, Saurabh Sinha, and Gene E. Robinson. Big Data: Astronomical or Genomical? *PLOS Biology*, 13(7):e1002195, 7 2015.

[106] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147, 1974.

[107] Paul B Tchounwou, Clement G Yedjou, Anita K Patlolla, and Dwayne J Sutton. Molecular, clinical and environmental toxicology: v.2: Clinical toxicology. *Choice Reviews Online*, 47(10):47–5683, 2013.

[108] Max Tegmark. *Life 3.0: Being Human in the Age of Artificial Intelligence*. Knopf Publishing Group, 2017.

[109] The Associated Press. EPA seeks dismissal of Gold King Mine spill lawsuit, 7 2018.

[110] Jonathan Thompson and Jeremy Wade Shockley. Silvertons Gold King reckoning, 5 2016.

[111] Chen Tianqi and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[112] Oswaldo Trelles, Pjotr Prins, Marc Snir, and Ritsert C. Jansen. Big data, but are we ready? *Nature Reviews Genetics*, 12(3):224–224, 3 2011.

[113] United States Environmental Protection Agency. Basic information about lead in drinking water, 2011.

[114] Rick van Hattem. Chapter 6: Generators and Coroutines - Infinity, One Step at a Time. In *Mastering Python*, chapter 6, pages 141–166. Packt Publishing, Birmingham, 2016.

[115] Francesca Volpetti, Ekaterina Petrova, and Sebastian J Maerkl. A Microfluidic Biodisplay. *ACS Synthetic Biology*, 6(11):1979–1987, 2017.

[116] Kevin J. Waldron and Nigel J. Robinson. How do bacterial cells ensure that metalloproteins get the correct metal? *Nature Reviews Microbiology*, 7(1):25–35, 2009.

[117] Eric W. Weisstein. Metric.

[118] World Health Organization. Adverse Health Effects of Heavy Metals in Children. *Children's Health and the Environment WHO Training Package for the Health Sector World*, pages 1–77, 2011.

[119] Jason H. Yang, Sarah N. Wright, Meagan Hamblin, Douglas McCloskey, Miguel A. Alcantar, Lars Schrübbers, Allison J. Lopatkin, Sangeeta Satish, Amir Nili, Bernhard O. Palsson, Graham C. Walker, and James J. Collins. A White-Box Machine Learning Approach for Revealing Antibiotic Mechanisms of Action. *Cell*, pages 1–13, 5 2019.

[120] Alon Zaslaver, Anat Bren, Michal Ronen, Shalev Itzkovitz, Ilya Kikoin, Seagull Shavit, Wolfram Liebermeister, Michael G Surette, and Uri Alon. A comprehensive library of fluorescent transcriptional reporters for Escherichia coli. *Nature Methods*, 3(8):623–628, 2006.

[121] Ce Zhang, Hsiung Lin Tu, Gengjie Jia, Tanzila Mukhtar, Verdon Taylor, Andrey Rzhetsky, and Savas Tay. Ultra-multiplexed analysis of single-cell dynamics reveals logic rules in differentiation. *Science Advances*, 5(4):1–11, 2019.

[122] Jian Zhou and Olga G. Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, 12(10):931–934, 2015.