



AFRL-RI-RS-TR-2022-097

CAUSAL HYPOTHESES FROM THE ANALYSIS OF OBSCURE SYSTEMS (CHAOS)

TWO SIX TECHNOLOGIES

JULY 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2022-097 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

CARL R. THOMAS
Work Unit Manager

/ S /

GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
JULY 2022		FINAL TECHNICAL REPORT		START DATE AUGUST 2017	END DATE JANUARY 2022
4. TITLE AND SUBTITLE CAUSAL HYPOTHESES FROM THE ANALYSIS OF OBSCURE SYSTEMS (CHAOS)					
5a. CONTRACT NUMBER FA8750-17-C-0231		5b. GRANT NUMBER N/A		5c. PROGRAM ELEMENT NUMBER 61101E	
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER R2CP	
6. AUTHOR(S) Singer, Jedediah, M, Rupley, Elissa, L, Eslami, Mohammed, A					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Two Six Technologies 901 Stuart Street Arlington VA 22203				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITA DARPA 525 Brooks Road 675 N. Randolph St. Rome NY 13441-4505 Arlington VA 22203			10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RI-RS-TR-2022-097
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT CHAOS is a novel and revolutionary approach for scientific discovery whereby causal and associative hypotheses are automatically generated from an analytics layer and models are refined using real-time experimental results. CHAOS' analytics automatically identify stable regions of system operation across temporal and physical scales.					
15. SUBJECT TERMS Scientific Disscovery, Feature Extraction, State-Space Prediction, Stability Analysis, Model Refinement, Model Reconfigurability, Autoencoders, Deep Neural Networks					
16. SECURITY CLASSIFICATION OF:				17. LIMITATION OF ABSTRACT	
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U		SAR	
				18. NUMBER OF PAGES 93	
19a. NAME OF RESPONSIBLE PERSON CARL R. THOMAS				19b. PHONE NUMBER (Include area code) N/A	

TABLE OF CONTENTS

FIGURES	ii
EQUATIONS	v
1 SUMMARY	1
2 INTRODUCTION.....	2
3 METHODS, ASSUMPTIONS, AND PROCEDURES.....	4
3.1 Model Layer - Feature Extraction	6
3.2 Model Layer - State-Space Prediction	8
3.3 Discovery Layer - Qualitative Relationship Discovery	9
3.4 Discovery Layer - Stability Analysis.....	10
3.5 Discovery Layer - Quantitative Relationship Discovery	12
3.6 Refinement Layer – Model Improvement and Reconfiguration	13
3.7 Computational Architecture	15
4 Results and Discussion	18
4.1 Versioned Dataset Repository	18
4.2 Predicting Promoter Strength	18
4.3 Escalation	19
4.4 Other Perovskites Tools.....	20
4.5 Protein Stability Modeling and Refinement	23
4.6 Protein/DNA Binding Prediction	26
4.7 Neuroscience Collaboration.....	26
4.8 Combinatorial Design Model	28
4.9 AutoGater	32
4.10 Test Harness	35
5 CONCLUSIONS.....	36
6 RECOMMENDATIONS.....	37
7 APPENDIX A - PUBLICATIONS.....	38
7.1 AutoGator: A weakly Supervised Neural Network Model to gate Cells to Flow Cytometry..	40
7.2 A Toolkit for Enhanced Reproducibility of RNASeq Analysis for Synthetic Biologists	53
7.3 Reconstruction of Neural Visual Scenes from Primary Visual Cortical Neural Activity	69
7.4 Comparison and Evaluation of data-driven protein stability prediction models	81
8 APPENDIX B - REFERENCES.....	85
9 ACRONYMS	86

FIGURES

Figure 1 Concept Image.....	1
Figure 2 Concept Image.....	2
Figure 3 CHAOS revolutionizes scientific discovery through a three-layer architecture integrated with the TA4 data hub.....	5
Figure 4 CHAOS will use rule-based autoencoders to incorporate formal models. An autoencoder is trained with (left) and without (center) rules. A tuning term, p , provides a mechanism to balance the influence of formal models on accuracy and explainability (right)	8
Figure 5 Qualitative relationships are discovered with a two-step process: (a) features are grouped by their edge-wise relationships in the autoencoder and DNN; then, (b) discretized variations of features are fed through the models. The effects of feature variations provide qualitative relationships and associative hypotheses.	10
Figure 6 Adaptive flow of the integration of EDMD and Ulam’s method for coarse- and fine-tuned dynamical systems analysis. Regions of attraction and their boundaries are highlighted to mitigate experimental surprise.	11
Figure 7 System failure data is passed through autoencoder to uncover state transition model; unstable state is identified as system shifts to an unexpected stable state.	12
Figure 8. Multi-layered knowledge representations across systems with deep lifelong learning. 14	
Figure 9 CHAOS’ Adaptive Parallelization, Validation, and Refinement Architecture	16
Figure 10 Parallel execution of stability analysis workflow for state perturbations that identify stable regions of operation	17
Figure 11 Model performance predicting promoter strength for a library composed of combinations of discrete promoter components.	19
Figure 12 Example excerpt from an Escalation web page.....	20
Figure 13 Visualization from tool to reconcile between different XRD machines	21
Figure 14 Performance of our neural network embedding model against expert-selected molecular features.....	22
Figure 15 User interfaces from perovskite analysis tools.....	23
Figure 16 Evaluation of models and datasets for protein stability prediction	23
Figure 17 Successful stabilization of synthetic proteins.....	24
Figure 18 Accelerating the discovery of stable proteins though machine-learning-guided search	25
Figure 19 AlphaFold2 predictions of our stabilized binder (red) overlaid on the predicted structure for the known binder (cyan).....	26
Figure 20 Comparison of baseline BLAST nearest neighbor model, two recently published models, and our gradient boosting model.....	26

Figure 21 Examples of lowest-quality reconstructions from model.....	28
Figure 22 Combinatorial Design Model	29
Figure 23(Left) Novel chassis NAND 2.0 Omics fluorescence predictive error measured through earth movers distance, and (right) yeast titration data.	29
Figure 24 Fluorescent predictions for a particular condition between E. Coli and Yeast	29
Figure 25 Predictions on impact of engineering a combination of gates for E. Coli and a combination of inducers for Bacillus.....	30
Figure 26 Architecture for Live/Dead Analysis.....	30
Figure 27 Plot showing the impact on the population of dead predicted cells over time	31
Figure 28 Threshold, Condition, and Cluster Definitions for Live/Dead	32
Figure 29 Using AutoGater to automatically select viable cells.....	33
Figure 30 Comparison of the traditional stain-based method versus AutoGater.....	34
Figure 31 Events that were changed from Live to Dead by the CFU objective	34
Figure 32 Test Harness Toolkit.....	35
Figure A-1 Autogator Methodology	41
Figure A-2 Autogator Training	43
Figure A-3 Autogator Results	46
Figure A-4 Autigator Comparison Matrix	48
Figure A-5 Comparison of dead matrix	50
Figure A-6 RNA-Seq workflow	57
Figure A-7 OMIC Tools Analysis	59
Figure A-8 Pearson's Correlations EColi	61
Figure A-9 Pearson's Coorelations FPKM	62
Figure A-10 Comparisons of Experimental Runs	63
Figure A-11 Comparisons of Experimental Runs	70
Figure A-12 Reconstructed Images	71
Figure A-13 Precision of Models	73
Figure A-14 Image Comparisons	73
Figure A-15 Model Precision Recall	73
Figure A-16 Preferred model	73
Figure A-17 Model Precision	74
Figure A-18 Histogram of Realism	74

Figure A-19 Reconstructed Images	75
Figure A-20 Reconstructed Images	75
Figure A-21 Reconstructed Images	76
Figure A-22 Reconstructed Images	79
Figure A-23 Reconstructed Images	79
Figure A-24 Reconstructed Images	80

Equations

Equation 1 Modality Imbalances 7

Equation 2 Lifelong Learning Techniques 14

1 SUMMARY

CHAOS will revolutionize and automate current models of complex systems via the effective utilization of data generated from high-throughput experiments to accelerate scientific discovery, generate hypotheses, and mitigate experimental surprise. CHAOS is built around four main interacting modules:

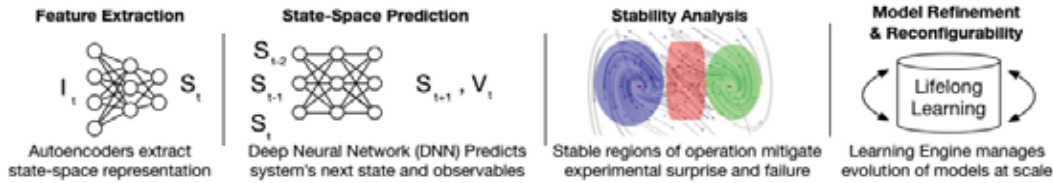


Figure 1 Concept Image

Autoencoders capable of automatically learning the mapping between observed variables and a *compressed feature representation* that continuously encodes the *state of the system*.

Deep Neural Networks (DNNs) that *predict future states* by capturing the *nonlinear dynamics* of the system.

Stable Regions of Operation extracted with the Perron–Frobenius operator that *reduce experimental surprise* and *enable discovery* of root failure causes.

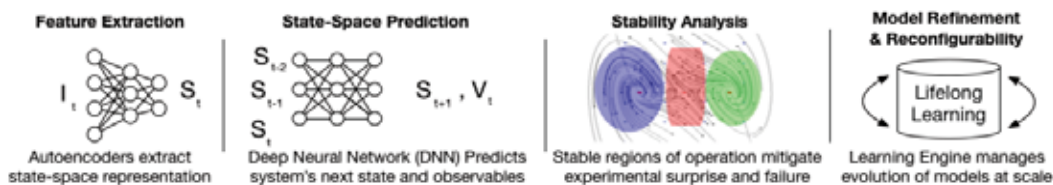
CHAOS will provide a novel data-driven modeling framework to significantly expedite the scientific discovery process in domains that lack complete models. This in turn will enhance a researcher's understanding of complex dynamical systems.

2 INTRODUCTION

Current scientific models in synthetic biology, materials, and neuroscience failed to represent all salient features of a complex system and its evolution over time. Scientists traditionally constructed models for complex systems using abductive reasoning and a limited number of observed variables. By designing experiments around plausible explanations for observed phenomena, they can miss the interconnection of attributes. Furthermore, scientists often experiment in a data silo, focusing on singular aspects of a system without considering second- and third-order effects. These methods resulted in experimental surprise and lack of reproducibility, with little or no insight into the root cause of failures.

To foundationally transform the design and discovery of robust models in domains that lack complete models, Two Six Technology, Netrias, Raytheon BBN Technologies (BBN), the University of Pennsylvania (Penn), and Professor James Yorke developed work for four years on the *Causal Hypotheses from Analysis of Obscure Systems* (CHAOS) effort.

CHAOS will revolutionize and automate current models of complex systems via the effective utilization of data generated from high-throughput experiments to accelerate scientific discovery, generate hypotheses, and mitigate experimental surprise. CHAOS is built around four main interacting modules:



- Feature Extraction: **Autoencoders** that are capable of automatically learning the mapping between observed variables and creating a *compressed feature representation* that continuously encodes the *state of the system*.
- State-Space Prediction: A **Deep Neural Network (DNN)** that *predicts future states* by capturing the *nonlinear dynamics* of the system.
- Stability Analysis: **Stable Regions of Operation** extracted with dynamical systems analyses that *reduce experimental surprise* and *enable discovery* of root failure causes.
- Model Refinement: A **Lifelong Learning Engine** that *efficiently re-configures and updates* deep networks, enabling cross-domain knowledge transfer and data rates of one terabyte (TB) per day.

CHAOS is a novel and revolutionary approach for scientific discovery whereby causal and associative hypotheses are automatically generated from an analytics layer and models are refined using real-time experimental results. CHAOS' analytics automatically identify stable regions of system operation across temporal and physical scales. The result is a robust set of design rules for TA2 performers. Accomplishing this challenge was completed by a multidisciplinary, collaborative team including Two Six Technologies (prime), Netrias, Professor James York, Raytheon BBN, and Dr. Eric Eaton

CHAOS enables a fundamental shift in the method for causal understanding of dynamical systems, like those in synthetic biology, from a fragile, abductive approach to a robust, principled one, through a domain-agnostic, scalable computational workflow that leverages proposed innovations in artificial deep neural network architectures, dynamical systems analysis, and lifelong learning. Together with other components of the SD2 system, the CHAOS framework reduces the time needed to make publishable scientific discoveries by exploiting high-volume, high-velocity experimental data.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

CHAOS instantiates a novel deep learning architecture informed by formal models and simulates dynamics of the system. The architecture can extract qualitative and quantitative relationships among heterogeneous, low and high-level features, each at its own time scale (e.g, gene expression $\rightarrow$ protein expression $\rightarrow$ cell growth). This approach is superior to other deep learning topologies used for heterogeneous features such as stacked autoencoders and Long Short-Term Memory (LSTM) networks because CHAOS works with heterogeneous sets of data and validates feature relationships through the incorporation of lower- and higher-level features at different temporal scales.

CHAOS performs beyond current state of the art simulation of dynamical systems, by using DNN models derived from massive, heterogeneous data to conduct stability analysis of experimental designs. Current methods for dynamical systems analysis of deep neural network models operate on homogeneous sets of data, overlook sensitivity analyses across multiple levels of the system, and do not diagnose root cause of failure.

CHAOS is unique, as the first large scale online model validation system capable of analyzing complex systems in real time. Central to the CHAOS approach is the ability to *construct and iteratively train a variety of machine learning models* utilizing petabytes of input data across millions of experimental variables. While open-source frameworks for single-machine and single-GPU training are widely available, current distributed and multi-GPU applications of these frameworks require extensive configuration and suffer from inflexibility to changing input scales.

CHAOS is a computational workflow designed to expedite scientific discovery (Figure 3). Its Model Layer takes a data-centric approach to maintain a nonlinear dynamical model of a system over time. Its Discovery Layer uses this model to generate associative and causal hypotheses by extracting qualitative and quantitative relationships between engineering variables, respectively, as well as identify design constraints through stability analysis. Finally, its Refinement Layer uses lifelong learning techniques to initialize and re-configure the model. A summary of each layer, their inputs and outputs, and their integration with one another is summarized below.

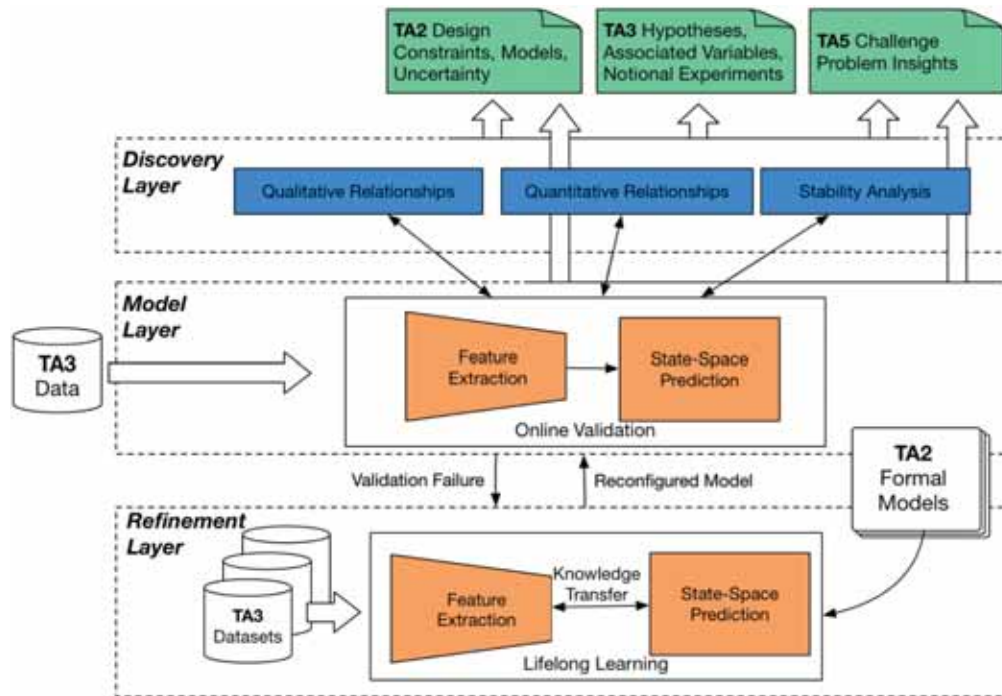


Figure 3 CHAOS revolutionizes scientific discovery through a three-layer architecture integrated with the TA4 data hub.

Model Layer: Ingests the stream of data from TA3 experiments and continually validates a nonlinear dynamical model of the system under study. Input data are compressed and translated into a state-space representation that is provided to a state-space prediction component. An online validation process continually ensures the compressed feature representation and state-space prediction correspond with the streamed data. The model is passed to TA2 for experimental design and sent to the *Discovery Layer* for analysis. In the case where either of these validation checks fail, the *Refinement Layer* will update the model. The Model Layer's components include:

- *Feature Extraction* - An autoencoder extracts salient features from the input data, forming a compressed state-space representation that is used by the CHAOS state-space predictor. This model is trained using experimental data and formal models (prior knowledge).
- *State-Space Prediction* - A deep neural network (DNN) provides a dynamical model of the system by learning the nonlinear function that maps past states to future states. This model is trained using experimental data and formal models.

Discovery Layer: Analyzes the autoencoder and DNN received from the *Model Layer* to generate semantically meaningful rules, constraints, and hypotheses from the system. Hypotheses are experimentally tested by TA3 and their results instantaneously contribute to CHAOS' model accuracy. Its components include:

- *Qualitative Relationship Discovery* - Extracts associations among experimental variables by analyzing the topology and evaluating discrete input vector outputs from the

autoencoder and DNN. Simple and complex associative hypotheses for SMEs and TA3 are generated here.

- *Stability Analysis* - Extracts root cause of failure and mitigates experimental surprise by perturbation analysis on engineering variables through simulations of the models. Design rules and constraints for TA2 are generated here.
- *Quantitative Relationship Discovery* - Extracts quantitative associations between engineering variables over time by performing regression analysis on local stable regions of the system. Simple and complex causal hypotheses for SMEs and TA3 are generated here.

Refinement Layer: In the case where validation checks from the model layer fail, the models will be reconfigured using lifelong learning. This allows CHAOS to massively scale model validation and refinement driven by TB/day data ingest.

- *Model Improvement and Reconfiguration* - Selectively transfer and reuse knowledge from previously learned models to accelerate learning and improve performance through lifelong learning. Learned models are refined to account for variation and salient features in new data.

Computational Architecture: All three layers are supported by a computational architecture that provides adaptive parallelization and scaling of model generation, refinement, and simulation. This architecture supports all three layers of CHAOS and will be closely integrated with TA4 APIs and compute/storage infrastructure.

- *Adaptive parallelization techniques* - Automatically parameterize and distribute multiple types of models from the *Model and Discovery Layers* using model and data parallel techniques.

CHAOS is designed to generalize across many kinds of datasets and domains. To demonstrate, we have included a concrete application of synthetic biology throughout the technical plan in the form of a challenge problem: *Does the strong expression of enzyme E inhibit the population growth of cell type O?*

3.1 Model Layer - Feature Extraction

Successful SD2 solutions will compress high-dimensional, multimodal, time-series data from scientific experiments into a concise, computationally practical yet expressive, representation of system state.

Dynamic models for complex systems face three main challenges in extracting relevant feature representations:

1. Since the computational requirements for SD2 will scale relative to the size of a compressed feature vector, the high-dimensional input data will need to be efficiently compressed.

2. Any compression of the input data will need to preserve the nonlinear relationships between dynamical system features for downstream exploitation and analysis.
3. Extracted features must preserve semantic relationships among the variables to enable the generation of meaningful hypotheses and experimental designs.

Natively, autoencoders offer many attractive properties to address these challenges, including their ability to compress an input vector to an arbitrary number of features, capture nonlinear relationships among input vectors [9], and perform unsupervised training on unlabeled data. While deep architectures are normally treated as black-boxes, recent uses of autoencoders show they contain semantically meaningful features in corpuses of biological datasets. The topology of these networks have been shown to represent structures of the biological systems. CHAOS will advance the state-of-the-art through incorporation of formal models to preserve interpretability among features and constrain the topology of the model to scale to high-dimensional spaces.

CHAOS will incorporate formal models of the biological system by extending state-of-the-art rule-based autoencoders. Traditionally, autoencoders minimize $L(y, y')$, where L is a loss function between y , the input, and y' , the reconstructed output. For rule-based optimization, however, the network minimizes a combined loss function: $L(y, y') + L(y, u')$, where u' is a prediction in the subspace limited by a set of rules or constraints on the network [12]. CHAOS expands the rule-based optimization function, where L is extended to account for imbalances between measurement modalities (λ):

$$\underset{y', u', p}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^N (1-p)L(y_n, y'_n; \lambda) + pL(y_n, u'_n; \lambda) \quad \text{Eq. 1}$$

Loss function Multi-scale parameters
 Average over N training samples Weight of formal models Prediction in rule based subspace

This extension will be needed, for instance, if one measurement is taken at one frequency and another measurement is captured at another frequency.

This approach allows CHAOS to examine the effect of a rule-constrained subspace and determine an effective point of operation where formal models are included within the overall system. This is illustrated in Figure 4. Example rules could restrict the number of edges for a node, the connections between nodes, or the composition of features. SMEs can use rule sets to preserve the explainability of a network by constraining its growth.

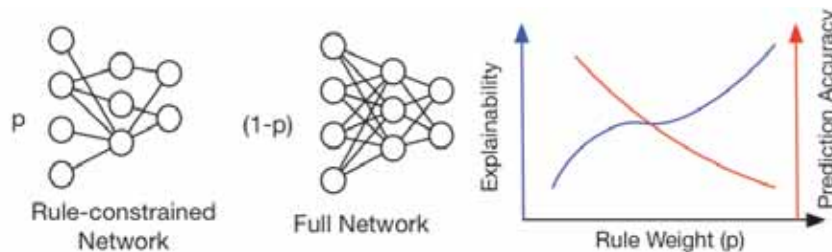


Figure 4: CHAOS will use rule-based autoencoders to incorporate formal models. An Autoencoder is trained with (left) and without (center) rules. A tuning term, p , provides a mechanism to balance the influence of formal models on accuracy and explainability (right)

3.2 Model Layer - State-Space Prediction

SD2 solutions will need to model how the states of a system change over time at multiple scales and modalities. CHAOS will use deep neural networks (DNNs) to simultaneously predict the next state in the system given a history of previous states and predict the values of higher-level features of the system.

Dynamical models in complex scientific domains face four main challenges to effectively predict the evolution of a system:

1. Data from experiments are captured at multiple system and temporal scales which requires lower-level features to be related to higher-level features, where higher-level features could be measured at different time scales (e.g. gene expression measured every minute → protein expression measured every hour → 4-hour circadian behavior patterns).
2. The relationships between features are nonlinear across time.
3. Outputs from a predictive model need to be validated with multimodal experimental data.
4. Semantic meaning among features needs to be preserved in the prediction of future states.

Natively, DNNs offer many attractive properties to address these problems, most notably, their ability to capture nonlinear relationships across time. Current methods for time-based prediction are recurrent neural networks (RNNs) and long short-term memory networks (LSTM). These methods have shown success in effectively handling multivariate time series data at different temporal scales. However, these methods contain feedback mechanisms that obfuscate semantic meaning of features. Furthermore, they have only been used for datasets of *single-level systems* (pictures from a video, or signals for clinical monitors) and have not been used to preserve semantic meaning among the features over time [3][4][5]. CHAOS will advance the state-of-the-art by training a DNN with data from multiple levels of the system and incorporate formal models to constrain the topology of the network while preserving semantic meaning. CHAOS will use a novel, rule-constrained, multimodal DNN to model the evolution of the states of the system over time. In many complex systems, higher-order features are derived from a history of lower-order features. For example, the expression of a protein can be a function of gene expression over time.

CHAOS' heterogeneous predictions can be validated with streaming, multimodal experimental data. This is possible because the DNN predicts two components given a history of state: the next state (S) and a set of higher-order system properties (V). The value of V is dependent on a history of states. In the case of a biological system, S could represent gene expression and V could capture protein expression. CHAOS can continually check its predictions against streaming data and know when its current models are failing to capture the incoming dynamics. This structure means the DNN will capture the temporal relationships between state features and the relationships that generate higher-level characteristics of the system. The validation of high-level features only needs to occur when

corresponding measurements are acquired, allowing CHAOS to operate even when measurements are at irregular intervals.

CHAOS' DNN will also need to incorporate formal models, preserve explainability, and minimize computational cost. For this, the DNN will need to incorporate rules into its optimization function, in a process similar to the one used for autoencoders. While traditional DNNs minimize the error between predicted values and actual values $E(y, y')$, our approach will need to minimize the difference between prediction vectors based on the observables $E([S, V], [S', V'])$ along with a term that encodes formal rules expressed by subject matter experts $E([S', V'], [S, V]) + \mathbf{R}$. These rules will be encoded in the state-space representation for the DNN and can include constraints on how states are combined for prediction.

3.3 Discovery Layer - Qualitative Relationship Discovery

SD2 solutions will need to extract qualitative relationships among input variables. In CHAOS, these relationships will be extracted from the topologies of the autoencoder and the DNN models, and through the evaluation of discrete perturbations to their input features. These relationships will form simple and complex associative hypotheses that will be provided to TA3.

Complex dynamical systems have a large number of variables that interact over time and ultimately result in the final state of the system. Models of these complex systems face two main challenges:

1. Relationships among variables are at varying timescales.
2. Simple hypotheses are difficult to extract from complex models.

State-of-the-art methods for qualitative relationship discovery are either topology-based or simulation-based. Current research efforts that use autoencoders on biological data show that the topology of the network encodes relationships among the input variables. For example and find that transcription factors and gene networks are encoded in the topology of autoencoders trained to compress gene expression data. Another work uses a qualitative simulator to discretize the feature space and evaluate the effects of qualitative inputs on the network's output. CHAOS will integrate these approaches in a novel two-stage process that extracts meaningful qualitative relationships and generates simple and complex associative hypotheses.

First, CHAOS will analyze the topology of the neural networks (autoencoder and DNN) to extract groups of features that appear strongly related to one another, where relationship strengths are encoded in the edge-weights of a neural network. Relationships can become obfuscated by neural network depth; thus, the most semantically meaningful relationships will likely need to be inferred from the first or second layers. These groups of features can be evaluated independently.

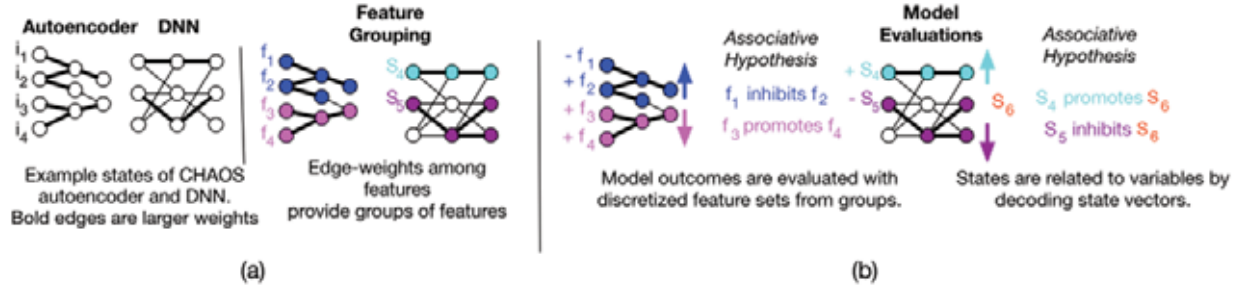


Figure 5 Qualitative relationships are discovered with a two-step process: (a) features are grouped by their edge-wise relationships in the autoencoder and DNN; then, (b) discretized variations of features are fed through the models. The effects of feature variations provide qualitative relationships and associative hypotheses.

Next, CHAOS will discretize the range of each feature into a small set of bins (e.g. bins of [low, medium, and high], or quantiles) to identify the extreme associations between the variables. A *variable* will be perturbed through its bins and the *effect* of the perturbations will be measured. Simple and complex associative hypotheses are formed by the concatenation of *variable-and-effect* measurements. The hypotheses can be experimentally tested by TA3, assuming that changes to given features are available. Once experimentally validated, a discovered relationship can be encoded as a design constraint for TA2. CHAOS will ensure this approach scales by only applying it to variables within a feature group.

3.4 Discovery Layer - Stability Analysis

SD2 solutions will need to analyze the state transition model of the system under study to mitigate experimental surprise and identify root cause of failure. CHAOS will apply a novel, scalable, two-phase approach, where a coarse-grain analysis first extracts *high-level* regions of stability, followed by a fine-grain analysis of those regions that mitigates experimental surprise. These regions will form design constraints provided to TA2.

A dynamical systems analysis of complex models requires the examination of a representative set of potential state transitions. This poses a set of unique challenges for complex dynamical systems including:

1. Computational methods scale exponentially with the dimensionality of the state representations [16].
2. Regions of stability and instability can be mixed within the state space.
3. Evaluation of all possible states' trajectories is infeasible.

Two well-known candidates for the extraction of stable regions of operation are the

Koopman and Perron-Frobenius operators (PFO). The Koopman operator limits itself to the evolution of observables [17] while PFO captures the probability distribution of state-space evolution. Therefore, PFO is a better fit for scientific discovery when observations themselves have uncertainty and the dynamical models are over *states* rather than *observations*. PFO identifies the states that are indicative of experimental surprise, i.e. when a slight change in the input will lead to an unexpected output. However, any approximation of PFO typically requires short simulations for a large number of different initial conditions. Without prior knowledge

about the system, approximation of regions of stability scales exponentially – a phenomenon attributable to *the curse of dimensionality*.

CHAOS will conduct stability analysis using a novel two-stage methodology to overcome the curse of dimensionality and approximate stable regions of operation from a TB of data per day. First, Extended Dynamic Mode Decomposition (EDMD) will be used to approximate flow transitions and highlight coarse regions of stability. Second, a simulation-based, fine-grain analysis will only be applied to those regions. This approach ensures that large-scale simulation is only used in select regions of the state space, limiting the required computational resources.

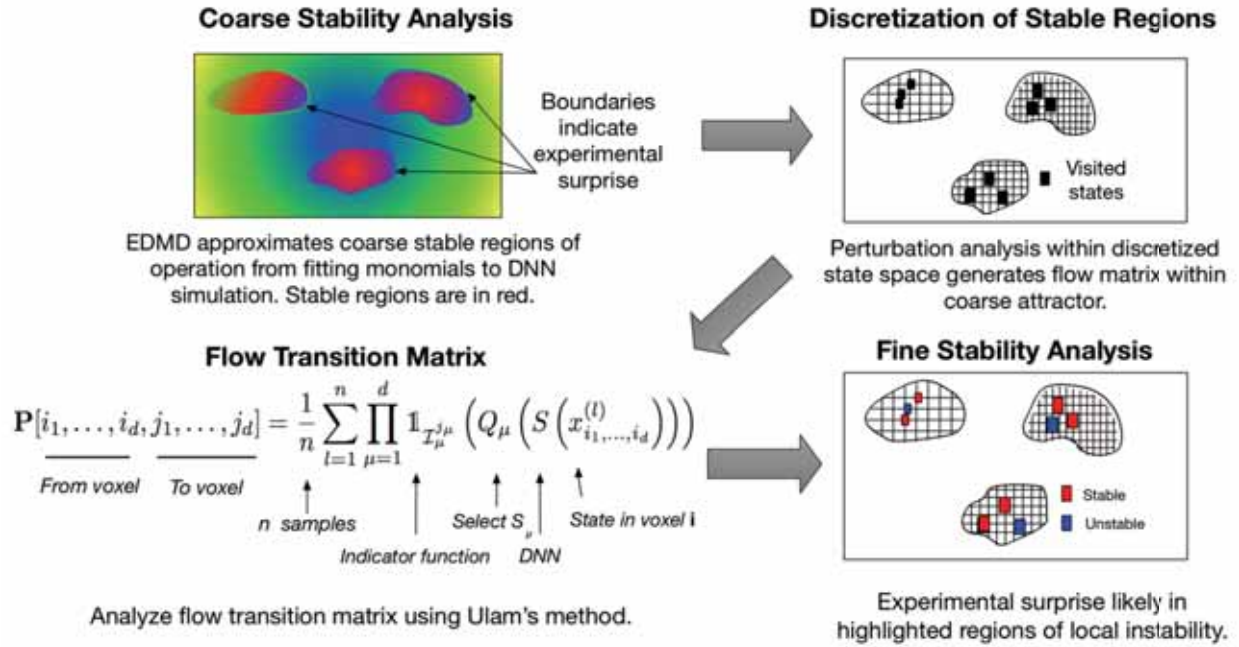


Figure 6 Adaptive flow of the integration of EDMD and Ulam's method for coarse- and fine-tuned dynamical systems analysis. Regions of attraction and their boundaries are highlighted to mitigate experimental surprise.

Coarse analysis will be applied with EDMD, a method that analyzes the coefficients of the continuous basis functions that are found from a regression analysis on simulated trajectories of the system. The dominant eigenvectors of the coefficient matrix highlight the most *frequently visited* regions that make up the *basin of attraction*, also known as the *region of stability*. Conversely, outside of the boundaries of the basin are regions of experimental surprise, or regions of instability.

Second, these areas are then discretized to perform a more fine-tuned analysis using Ulam's method to mitigate experimental surprise. Let us define a voxel as a d -dimensional grid point that covers a region of state space, and k as a uniform discretization of each dimension. To discretize the entire state space would take k^d voxels, which is not feasible as d gets large. Discretization of the attractors found by EDMD significantly reduces the number of voxels needed to the total volume of the manifold that covers the attractors (ideally, $\ll k^d$). A transition flow matrix is then constructed from the simulation of the model in this reduced discretized space. The flow matrix has entries p_{ij} that captures probable flow transitions from voxel i to voxel j .

CHAOS will use the low-rank tensor formulation of this large transition matrix presented in [18] that applies a canonical numbering of the boxes to transform its d -dimensional approximation of indicator functions to a product of d one-dimensional indicator functions. This low-rank approximation reduces the memory required for the approximation from $O(k^d)$ to $O(kd)$. The eigenfunctions of the largest eigenvalue of the tensor are extracted to identify local regions of stability (these are regions that are most frequently visited by the system). While computing a full eigen-decomposition of tensors is NP-hard, [18] provides a power iteration method to extract the dominant eigenvectors of the system. These eigenvectors highlight regions of the attractor that are visited less, and thereby unstable.

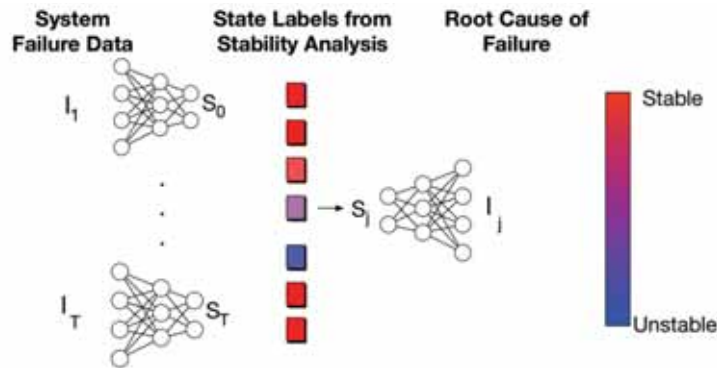


Figure 7 System failure data is passed through autoencoder to uncover state transition model; unstable state is identified as system shifts to an unexpected stable state.

As a system evolves over time there will likely be instances where it will fail. In these instances, we will need to determine root cause of failure (see Figure 7). The system will produce a set of time-series data $\{I_1, \dots, I_T\}$ that can be passed through the autoencoder to determine its state transitions. The autoencoder will provide the state-space encoding of the data from the time the system was monitored to failure $\{S_1, \dots, S_T\}$. Upon comparison of $S_1, \dots, S_T$ to the states extracted as *unstable* states from CHAOS' stability analysis, we determine when and which state was unstable. That state is fed back to the decoder to reconstruct the engineering variables that were most likely responsible for the system's failure. These engineering variables and the associated time are passed to TA2 for further analysis.

3.5 Discovery Layer - Quantitative Relationship Discovery

SD2 systems need to build causal hypotheses by identifying the quantitative relationships among variables for solutions to inform the TA2 designers, engineers, and scientists. CHAOS will use "shallow nets," or simple regression models, to approximate the quantitative relationships between the variables in localized regions of stability. The principal challenges with the discovery of quantitative relationships are that:

1. Simple quantitative relationships can be nonlinear.
2. There is a large space of potential relationships.

Current research uses simple machine learning models on isolated regions of feature space to identify quantitative relationships [19], where linear regression expresses the relationships between features for classification. However, this system requires a set of linear relationships within a neighborhood of the feature space. CHAOS will extend this method to work for its models and to include more complicated, nonlinear relationships.

CHAOS will extract quantitative relationships and generate causal hypotheses from stable regions of operation within the state space. By selecting conditions that lead to stable orbits, we increase the likelihood that we will discover simple relationships that exist between features in our models. Then, CHAOS will train shallow nets for these orbits. Initially, a 2-level neural network will perform linear regression on the input features, and those operations encode quantitative relationships.

CHAOS' use of shallow networks allows tuning of the complexity extracted from relationships. This mechanism enables the system to iteratively increase the complexity of the shallow net to construct a set of rules that generate hypotheses about the system. Complexity can be added by increasing the number of network layers. A simple hypothesis from a two-layer network could be "affecting feature I_a doubles the expression of feature I_b with a five-minute delay." A more complex hypothesis from a three-layer network may be, "affecting I_c exponentially diminishes I_d . These hypotheses can be described in rules for TA3 to test and, when verified, provided as design constraints to TA2.

3.6 Refinement Layer – Model Improvement and Reconfiguration

Successful SD2 solutions will require complex models to be refined when new data is received and generalized to share knowledge across domains. CHAOS will use lifelong learning to transfer knowledge between its components when models are refined. Models will need to be refined when they do not fit incoming experimental data; they will need to be reconfigured when they need to be applied to new processes or a new system. This produces a set of challenges:

1. Methods are needed to scale to TB/day streams of data that will reconfigure models rather than completely retrain them.
2. Knowledge must be transferred between models of feature extraction and state-space prediction as they are updated.

Lifelong learning techniques can be used effectively in classification, regression, and reinforcement learning domains. These methods provide state-of-the-art performance in learning multiple tasks online across numerous domains. These methods optimize the performance of each individual task model while maximizing the amount of knowledge $\mathbf{L}$ shared between tasks $(t^{(0)}, t^{(1)}, \dots)$, mapping knowledge autonomously across domains via projections Ψ :

$$\arg \min_{\mathbf{L}, \mathbf{S}, \Psi} \underbrace{\Omega(\mathbf{L}, \Psi)}_{\text{Relationships between tasks and knowledge}} + \underbrace{\sum_{t^{(i)} \in \mathcal{T}} \mathcal{L}(h(X^{(i)} | \boldsymbol{\theta}^{(i)}), t^{(i)})}_{\text{Aggregate Model fit to data on task } t^{(i)}} + \underbrace{\mu \|\mathbf{s}^{(i)}\|_1}_{\text{Sparsity}}, \quad \text{Eq. 2}$$

$$\text{where } \boldsymbol{\theta}^{(i)} = \underbrace{\Psi_{\mathcal{D}(\mathbf{a}_i)}(\mathbf{L})}_{\text{Knowledge in } \mathbf{L} \text{ projected by } \Psi \text{ to the domain } \mathcal{D}(\mathbf{a}_i) \text{ of task } t^{(i)}} \mathbf{s}^{(i)}.$$

Notably, it has been shown that with a two-level hierarchical factorization of the learned knowledge, lifelong learning can transfer knowledge effectively between task domains with different feature and output dimensions. This work has been shown to speed up model reconfigurability 1000x [20]. Although effective, work in lifelong learning has been limited to relatively simple linear models with two-layer factorized knowledge representations. Furthermore, these approaches are based on the idea of factorizing each model's parameters into a sparse linear combination over a set of shared basis functions to facilitate transfer between models. CHAOS will extend this lifelong learning framework to more sophisticated models, namely deep models and multi-layered knowledge representations, enabling the lifelong learning process to capture significantly more complex system models (Figure 8)

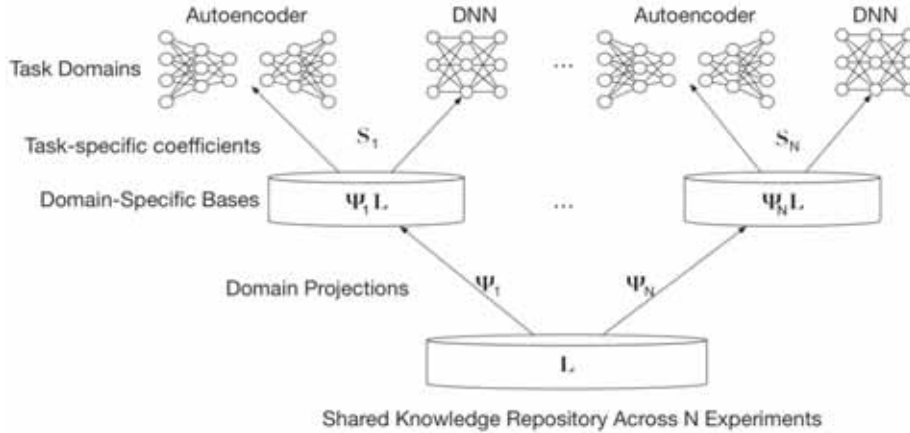


Figure 8. Multi-layered knowledge representations across systems with deep lifelong learning.

CHAOS will employ a deep hierarchical representation of the shared knowledge, enabling it to be autonomously transferred between different tasks at the **appropriate level of granularity**. The top layer of the trained deep network is the sparse function, with the other layers of the network representing the basis features at different granularities. Given a new task, CHAOS will reuse these lower-level features by retraining the top layer. This same notion can be extended to the lower layers, capturing multiple levels of abstract, shared knowledge. Our deep lifelong learning approach will dynamically select among the different multi-level deep features to maximize knowledge transfer between tasks, using stochastic gradient descent to refine each layer of features as needed. Critically, unlike model updates in traditional machine learning methods, updates to the autoencoder will not require the retraining of the DNN, allowing the model to adapt rapidly to handle new learning objectives at scale.

As the shared knowledge base is refined, it will form an increasingly stable basis for the learned models, thereby ensuring that the models are robust to varying conditions. Moreover, because models will be built upon shared knowledge, subsequent refinements of particular chunks of knowledge can improve the performance of previous models through the process of reverse transfer. This same computational workflow will be able to map changes between models, even if they apply to different biological systems. Both mechanisms are accomplished by maintaining an annotated history of models and their interactions.

3.7 Computational Architecture

SD2 algorithms need to scale to large amounts of data streaming into the system, building into petabytes of data by program end. CHAOS will utilize adaptive parallelization and refinement strategies to support scalable model generation, validation, reconfiguration, and simulation. This approach addresses the following challenges:

1. Scaling over streaming data to ensure that the autoencoders and DNNs efficiently and accurately track the state of the system over time.
2. Scaling over data at rest to automatically parameterize and distribute our machine learning algorithms across clusters of machines.
3. Simultaneous execution of many models with small input perturbations to determine stable regions of operation and extract root cause of failure.

Current approaches for machine learning model parallelization require extensive manual configuration to both the model and the execution environment. Even when parallelized, model reconfigurations that require full re-training can be expensive and do not take into account previously learned versions of the model. CHAOS solves both problems by combining online model refinement and adaptive parallelization strategies with incremental stream-based processing to support 1 TB/day processing rates. An architecture summarizing this combination is seen below:

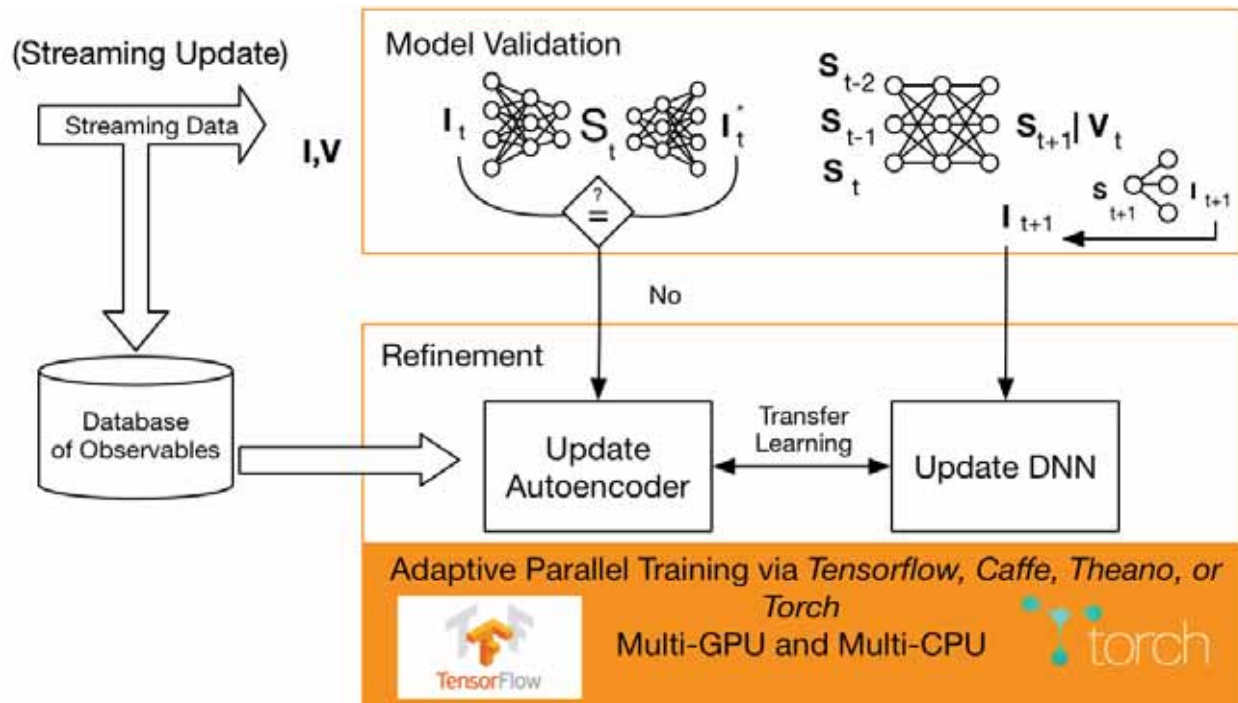


Figure 9 CHAOS' Adaptive Parallelization, Validation, and Refinement Architecture

To handle high-volume data streams generated by TA3 providers, CHAOS will inform the selection and use of a fast and scalable streaming framework from TA4, such as Apache Kafka, Flink, or Samza. CHAOS will ensure that its prior models track new streaming data, and quickly and automatically refine models when points of failure are detected. For example:

- Autoencoder failure: decoding of state vector S does not equal input data. This can be accomplished via a tunable error rate over the autoencoder's error loss function.
- DNN failure: state prediction S_{t+1} diverges from next state-represented observation. Flexibility over failure conditions can be provided by the use of a sliding error window over the predicted time.
- DNN failure: higher-level feature V is not accurately predicted at time t . Flexibility over failure conditions can be provided by use of a sliding error window over the predicted time.

CHAOS uses lifelong learning to reconfigure and update the components that failed, but initially the autoencoders and DNNs will need to be trained. CHAOS will employ adaptive, distributed model and data parallel training techniques to batch-train models with millions of parameters, illustrated in Figure 9. Using metadata such as the number of model input parameters, the computational cost of the model's updates, and the available network bandwidth, CHAOS will select an appropriate strategy and parameters for a model, vastly reducing configuration and development complexity. CHAOS will work closely with TA4 to automatically distribute and execute our models across available compute infrastructure.

The dominating parameter in selecting a model or data parallel strategy is whether the model fits into memory – the larger the model, the more we benefit from parallelism – while small models suffer from network overhead to communicate updates. This approach is generalizable across a variety of deep learning frameworks, such as TensorFlow and Torch, and can be applied to both multi-GPU and multi- CPU compute architectures, scaling to deep networks with billions of parameters. This reduces the dependency risk of relying on a specific TA4 compute provider. This improvement can be shared with all TA performers, shortening model configuration, training, and testing cycles, and scaling every module of SD2 beyond single machine, baseline performance. An application of these strategies to the CHAOS’ modules is shown in Figure 9.

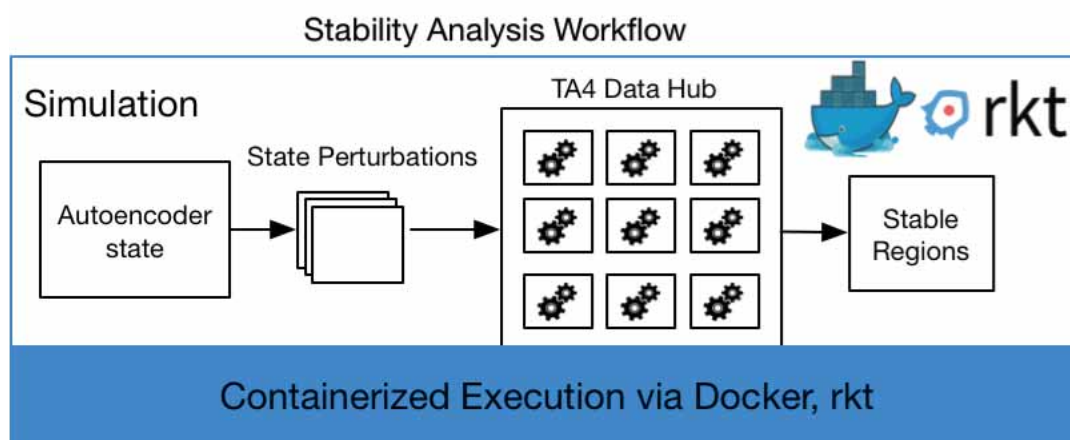


Figure 10 Parallel execution of stability analysis workflow for state perturbations that identify stable regions of operation

Finally, stability analysis (Figure 10) will require the scalable simulation of many input variations on our state space. We will parallelize the execution of these model perturbations using containerized execution, e.g. Docker or Rocket, to allow a large number of simulations to be distributed, launched, and managed in a structured, scalable way across a TA4 compute cluster. The expression of this execution is well suited to a directed-acyclic-graph-based workflow language that describes the model permutations, their ordering constraints, and how inputs and outputs flow between different simulation instances. As with streaming updates to our autoencoders and DNNs, model simulations only need to be re-run in the case of failure conditions.

4 Results and Discussion

4.1 Versioned Dataset Repository

At the start of SD2 (and even now), scientific collaborators often had difficulty organizing and sharing data and analytical results. Often people resorted to emailing CSV files or sharing them via Slack. The resulting lack of traceability and standards led to several cases of confusion and disagreement, when different research groups unknowingly operated on different versions of data files. In other cases, people were unsure of where data files came from, or whether postprocessing had been applied, or why the values in one version of a file didn't match those in another version.

To solve these problems, we built the Versioned Dataset Repository. This software uses Git to ensure that the following requirements are met:

- Whenever a data file is updated, it contains the right columns, the right number of rows, and reasonable data.
- Updates are associated with a user and a date and must be accompanied by an explanatory message detailing why they exist and what they accomplish.
- Users of the data can always easily ensure that they are always working with the latest version of a data set.
- All previous versions of each data set are maintained for posterity.

This software was deployed to TACC, where it served the Protein Design and Perovskites challenge problems over the course of the program, as an important part of each challenge problem's data and analysis pipeline. It also supported data/analysis sharing, versioning, and management for several of the summer collaborations in which SD2 analysts were matched with data providers from other DARPA programs (including the neuroscience collaboration with the Schnitzer lab, which has continued through the end of the program). Lessons learned from its development motivated TACC to adopt the Frictionless data standard going forward, a recently-released set of tools and data standards that accomplish much of what we built into the Versioned Data Repository in a more general way.

4.2 Predicting Promoter Strength

In the past, machine learning models have been developed to predict a bacterial promoter's transcriptional initiation rate using expert-labeled sequence elements across a defined set of DNA building blocks. The generalizability of these methods is limited by the necessary labeling of the components studied. If generalizable models existed, they could greatly facilitate the design of synthetic genetic circuits with well-controlled transcription rates in bacteria.

To address these limitations, we built a convolutional neural network to predict a promoter's transcriptional initiation rate directly from its DNA nucleotide sequence. We first evaluated the model on a previously published promoter component dataset. Trained using only the sequence as input, our model matched the performance of published models that fit expert-labeled sequence elements. For SD2, the Salis lab produced a new promoter strength dataset including

non-repetitive promoters with high sequence variation, not limited to combinations of discrete expert-labeled components. Our CNN trained on this more diverse dataset fits held-out promoter strength with $R^2 = 0.61$. Previously published models cannot even be applied to a dataset like this with highly diverse input. This work was published in 2020 and has already had an influence on the field.

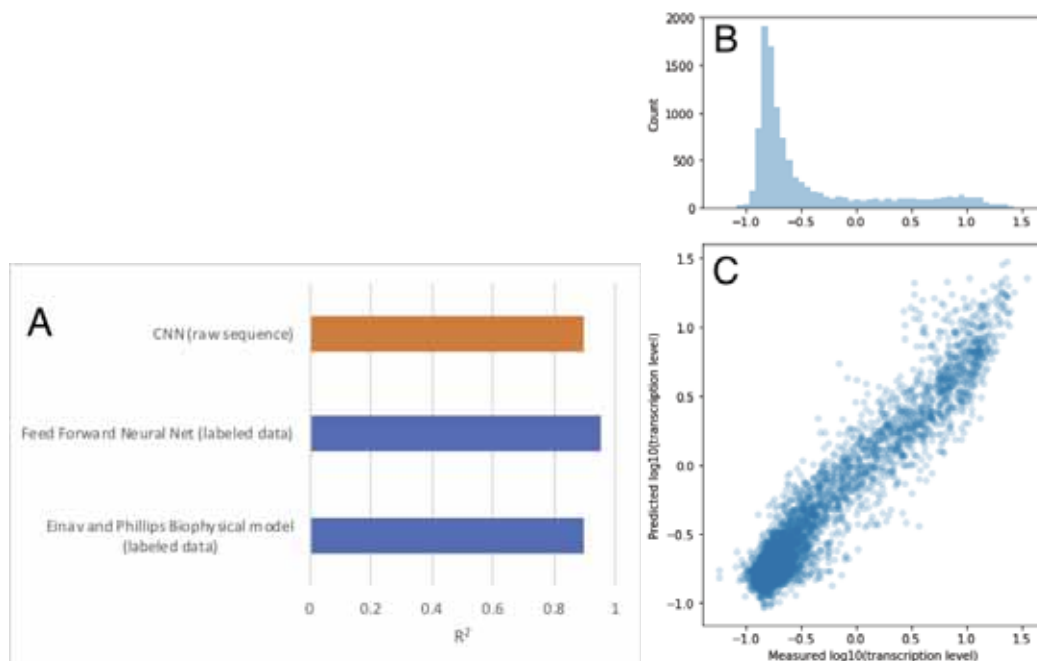


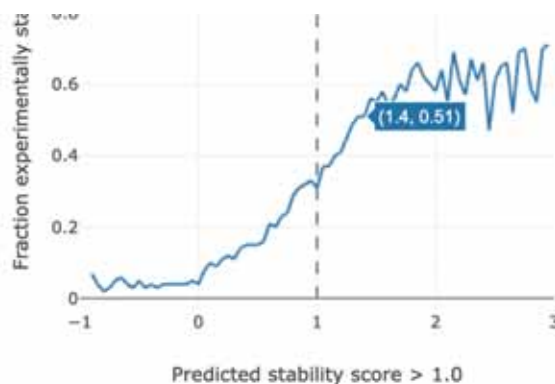
Figure 11 Model performance predicting promoter strength for a library composed of combinations of discrete promoter components.

Model performance predicting promoter strength for a library composed of combinations of discrete promoter components. A) R^2 for the sequence-based CNN model compared to published models fitting expert-labeled sequence components. B) Distribution of promoter transcription rate (log10-transformed) in this dataset. C) CNN model predictions for held-out test data versus the observed transcription rate.

4.3 Escalation

We created the Escalation visualization dashboard, which compiles and presents experimental results, automatically runs ML models and analysis code on experimental data uploads, assesses and compares the quality of ML model predictions on past data, and suggests the next set of experiments to be run based on ML model performance in an iterative “bake off” of prediction outcomes. This allows for a closed loop of automated experimentation. Unlike existing tools, Escalation is geared towards scientists and is open source. It stores user-uploaded CSVs or data from a variety of other sources in an automatically maintained database and enables the painless creation of interactive graphs for sharing and display within a browser. Within SD2, Escalation was used as a portal for our protein stabilization paper, to view ongoing experiments for the yeast states challenge problem, and to track and visualize data and the results of (Escalation-run) analytical models for the perovskites challenge problem.

Escalation offers support for word clouds, linked controls of visualizations (e.g. multiple graphs that all respond together to a single changed filter), performing data aggregations on the back end rather than the front end, improved password security, and Elasticsearch fully unconstrained by the Python Elasticsearch API. An example of a small part of an Escalation dashboard that we deployed for the protein stabilization work is seen below.



Stability Scores: Predicted versus Observed

Figure 2. (C): Predicted versus observed stability scores for the library of new designs.

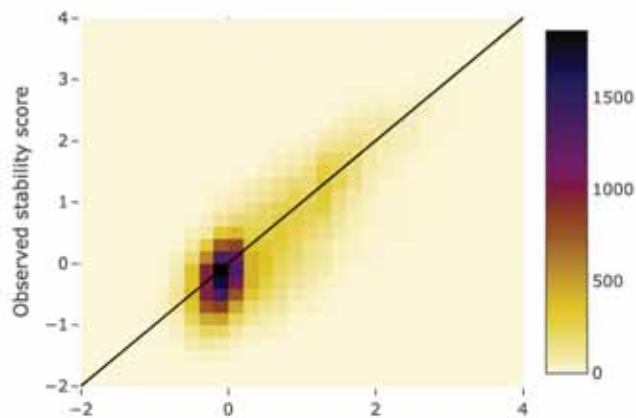


Figure 12 Example excerpt from an Escalation web page.

4.4 Other Perovskites Tools

On a data-science and -engineering level, we developed techniques for mapping between XRD data generated by different machines, which was a boon both for our own modeling efforts and for others in the program.

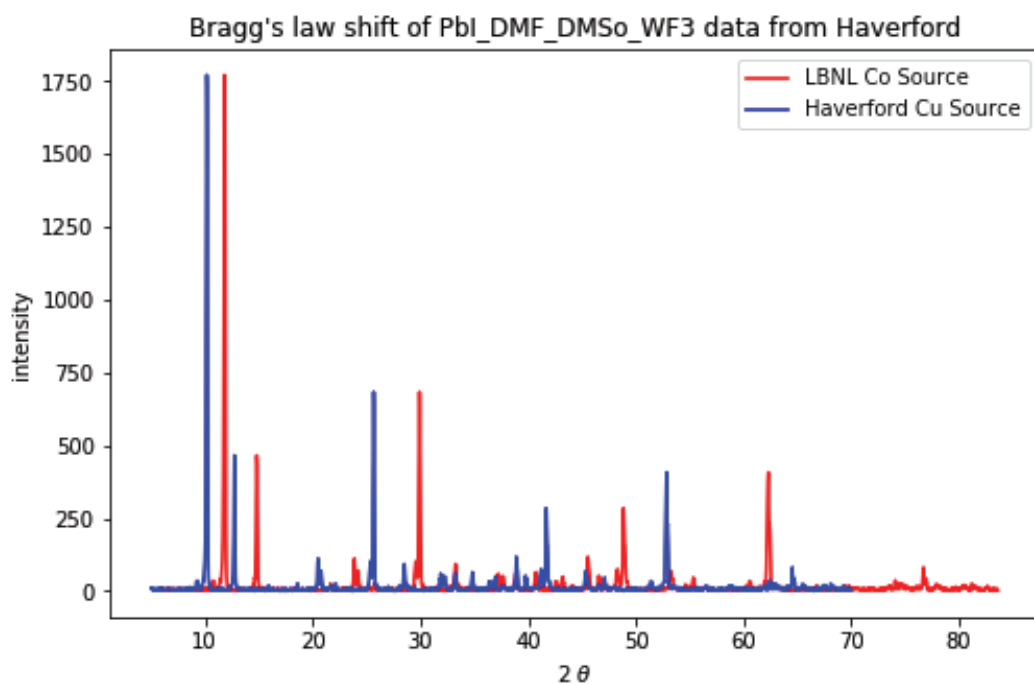


Figure 13 Visualization from tool to reconcile between different XRD machines

We built a quasi-autoencoder to generate embedding vectors for amines used in perovskite reactions. It is only “quasi” because it maps between ECFP and SMILES, which are two different ways of representing molecules. We applied consistency constraints on the embedding space itself. While the results are not an obvious improvement over previous work, they are comparable to results seen with a set of molecular features:

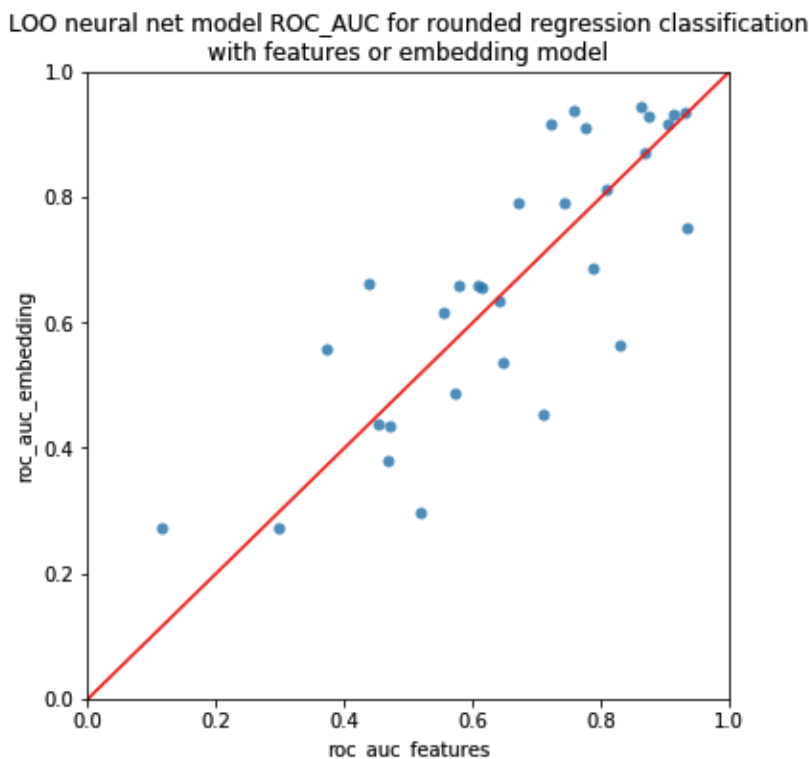


Figure 14 Performance of our neural network embedding model against expert-selected molecular features

We used jackknife statistics to discover individual experimental data points that were responsible for inordinately large changes in overall data sets and suggested them for replication to ensure that a few errors weren't throwing off entire data sets. This metric was also highly correlated with another, the number of bad crystals contained in convex hull of good crystals.

We also built tools that supported experimentalists in their primary data analysis. This reduced data storage needs by three orders of magnitude and expert annotation time by two orders of magnitude. Below are user interfaces from two tools. The first, on the left, supports analysis of differential laser refraction videos to measure diffusion coefficients. The second (center and right) precisely tracks solvent volumes and crystal size.

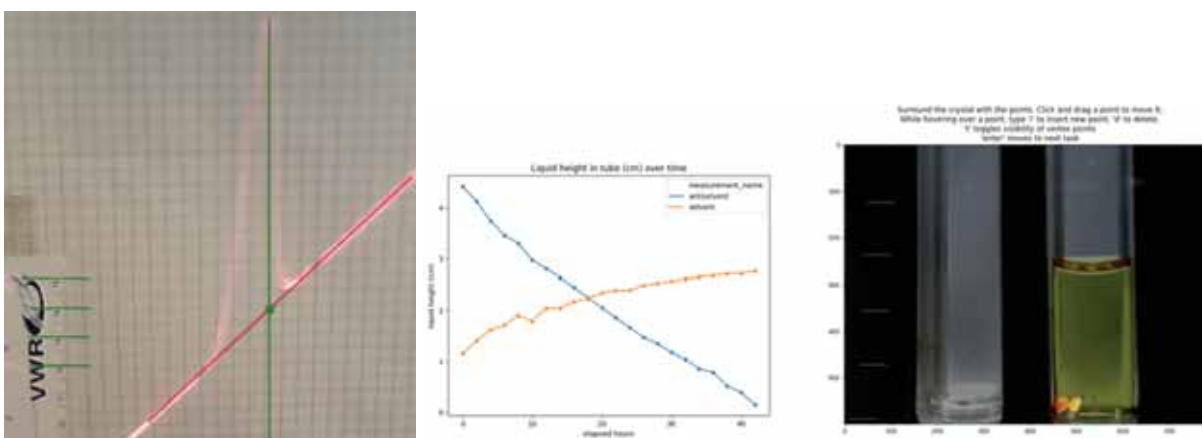


Figure 15 User interfaces from perovskite analysis tools

4.5 Protein Stability Modeling and Refinement

Stability is a near prerequisite for engineered proteins; unstable designs are likely to be quickly degraded and are unlikely to perform the function conferred by their intended shape. Over the course of SD2, we dramatically improved our ability to design, predict, and refine stable proteins, and to build and evaluate those designs. We developed efficient and high-throughput design and analysis tools and used those to create large data sets. These data sets demonstrated that libraries of 100k designs could yield data of similar quality to experimental runs using only 15k designs, the standard prior to SD2. We developed data-driven (ML) design methodologies that can reliably create novel stable proteins, which had never been done at the start of SD2. We built machine-learning models that can predict a design's stability using only sequence information. These models perform ~20% better than the state of the art at the beginning of the program, on a data set ~30x more complex, without the need for structural information about the proteins being evaluated. We also contributed a majority of the designs to a public data set of ~300k proteins with evaluated stability metrics (up from ~20k available at program start).

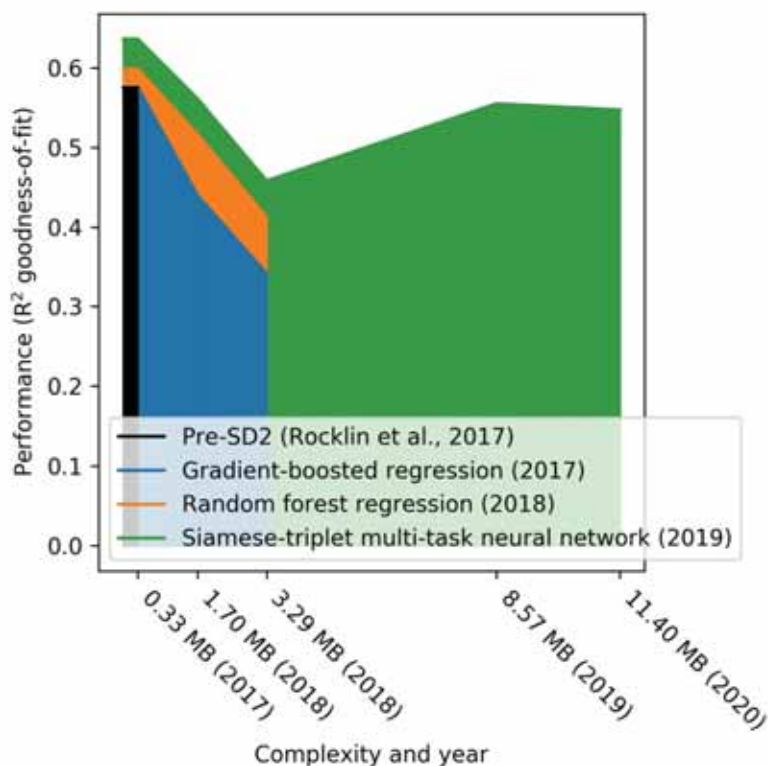


Figure 16 Evaluation of models and datasets for protein stability prediction

We showed that our stability prediction models can refine the stability of designs ten-fold, under a wide variety of constraints, making possible the automated creation of enormous libraries of stable designs with other desirable properties. This could enable large-scale screening or directed evolution approaches. This work yielded a publication, currently under review. See below for an evaluation of our stabilization method compared against random mutations.

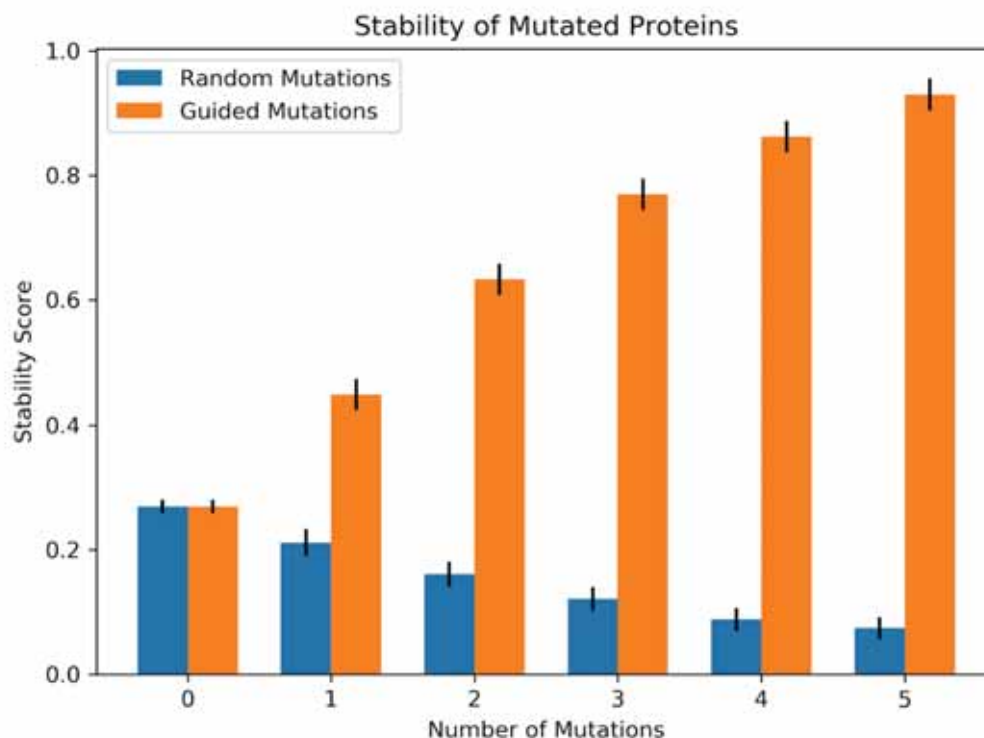


Figure 17 Successful stabilization of synthetic proteins

These components, as well as additional software for pre- and post-processing that we built, formed part of an automated pipeline capable of executing closed design-build-test-learn loops for protein stability. We demonstrated through virtual experiments that this loop could discover stable designs 150% more quickly than random exploration. See below for results; guided search was not only able to reach a given threshold 150% more quickly, but the results obtained at plateau were about 12% more stable. While this complete pipeline was never physically executed, due to stability per se having been adequately “solved”, we believe it could prove useful for design and refinement of more complex proteins under more substantial constraints. The computational components of the pipeline did see use, and cut the time needed for analysis from several weeks to less than a day.

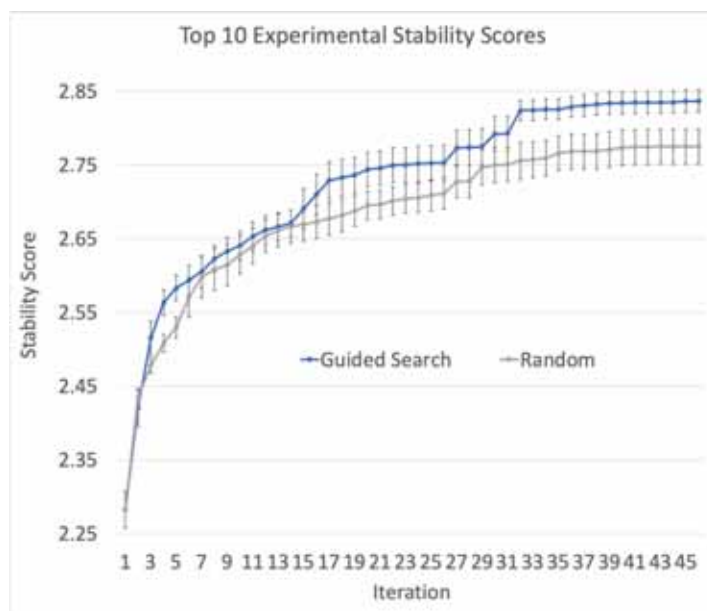


Figure 18 Accelerating the discovery of stable proteins through machine-learning-guided search

To demonstrate the utility of our stabilization pipeline in functioning proteins, which will make transition more appealing to potential partners or customers, we have two experiments currently running. One, being executed by the Baker Lab at UW, examines our ability to stabilize synthetic binders developed by the Baker Lab for use against cancer and covid. The other, being executed by Ellen Goldman's lab at the Naval Research Lab, combines our stabilization pipeline with structural predictions from AlphaFold2. The goal is to optimize the stability of a known binder to the coronavirus nucleocapsid protein, without damaging the structure of the protein's binding region. Below are AlphaFold2 predictions of our stabilized binder (red) overlaid on the predicted structure for the known binder (cyan).

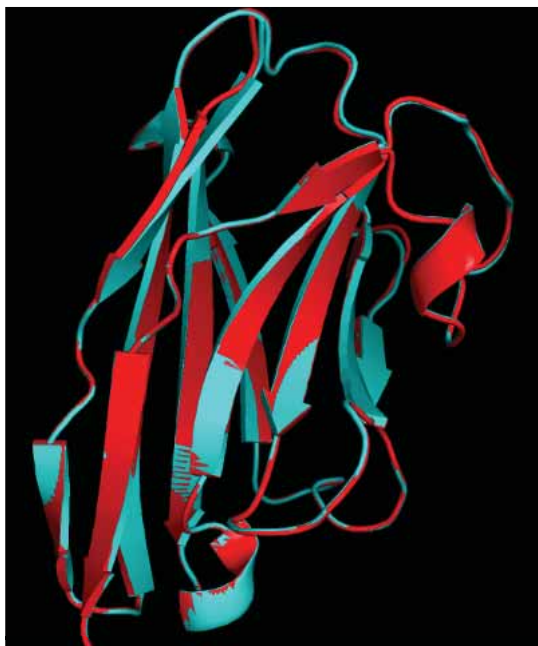


Figure 19 AlphaFold2 predictions of our stabilized binder (red) overlaid on the predicted structure for the known binder (cyan)

4.6 Protein/DNA Binding Prediction

We pursued an investigation into whether proteins would bind to DNA. Prior to SD2, most such studies relied on a data set that suffered from substantial flaws (most notably that 80% of the test data were also present in the training data). We curated and published a new data set that remedied these flaws, as well as providing many more data points (~362k vs. ~1.2k). We showed that, on this improved data set, recently published models performed much worse than a simple BLAST-based nearest neighbor model (Matthews Correlation Coefficients of 0.32 and 0.54 vs. 0.85), despite those models' computational complexity. We also showed that an efficient model using published embedding tools and gradient boosting performed slightly better than the nearest neighbor model (MCC of 0.88), suggesting a more fruitful avenue for future development (see below). This work has been published.

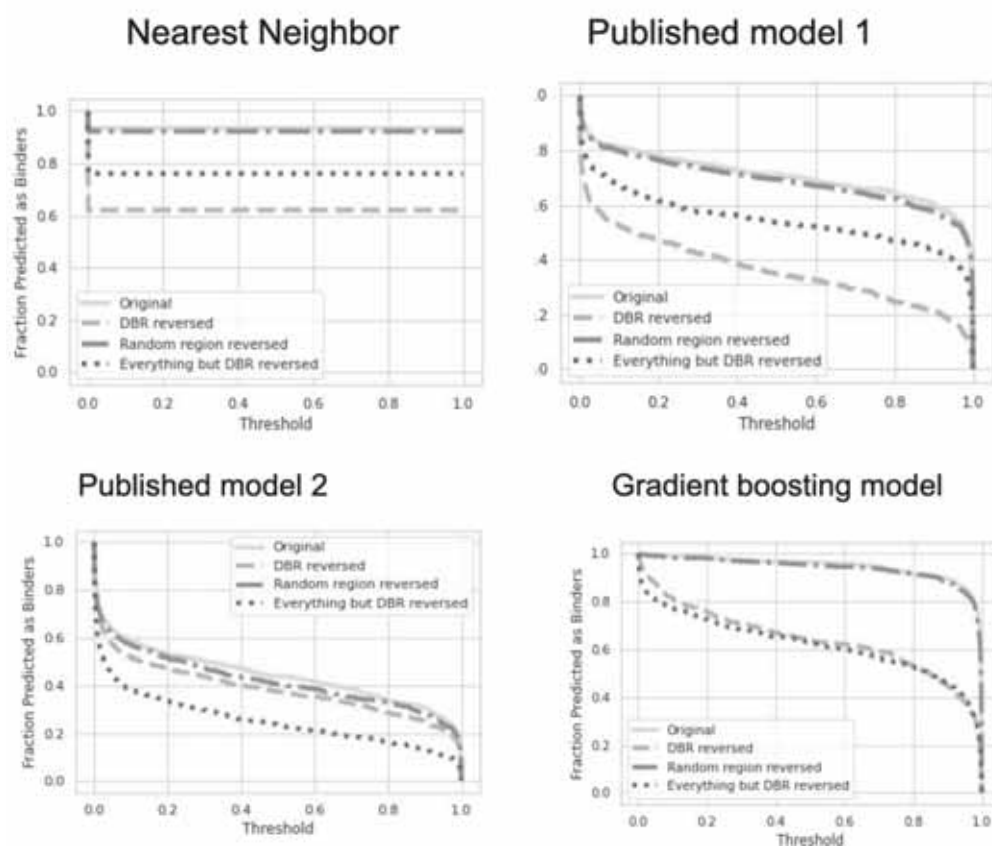


Figure 20 Comparison of baseline BLAST nearest neighbor model, two recently published models, and our gradient boosting model

4.7 Neuroscience Collaboration

We initially began working with the Schnitzer Lab during the summer, 2019 data-sharing exercise. The collaboration was fruitful enough, however, that we maintained it through the remainder of the program. They have developed a technique for recording from hundreds of neurons in mouse cortex simultaneously. Our initial analyses focused on neurons in the hippocampus, essential for navigation and memory. We were able to predict the mouse's location based on its neuronal activity, slightly more accurately and two to six orders of magnitude more quickly than existing state of the art decoding. This allowed for real-time decoding, which inspired us to turn our attention to processing of visual information.

Our goal has been to both better understand neural visual processing and (explicitly and implicitly) take advantage of neural processing to aid computer vision of video. We used early pilot data in which a mouse viewed static images, while incorporating a large library of natural scene images from which we have extracted image statistics. We built a model that combines three loss functions (one pixel-based, one using a generative adversarial model to encourage images to be realistic, and one using a feature loss to encourage them to contain the right "things"). This reduces the blurriness of the predictions and, using the natural image statistics, can "guess" what the likely features of the presented image were even when the mouse's visual system was not acute enough to represent them. Below we show examples of the lowest-quality reconstructions from this model (the image shown to the mouse is on the left in each case, and the image reconstructed from the mouse's brain is on the right).

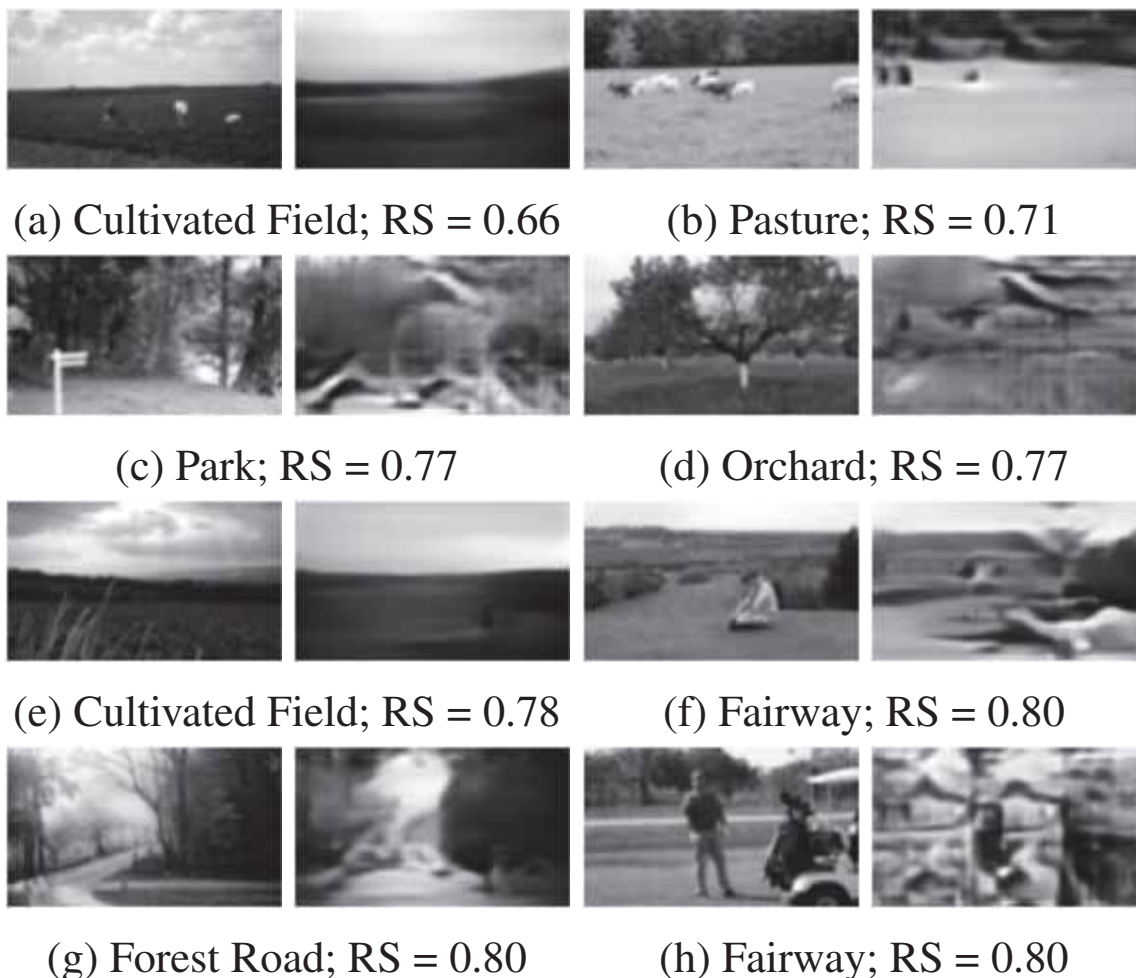


Figure 21 Examples of lowest-quality reconstructions from model

4.8 Combinatorial Design Model

The Combinatorial Design Model (CDM) consists of two independent models – one flow cytometry based to predict fluorescent output of circuits, and the second RNASeq based to predict the response of the host through differential gene expression prediction. The two branches are shown in the figure below:

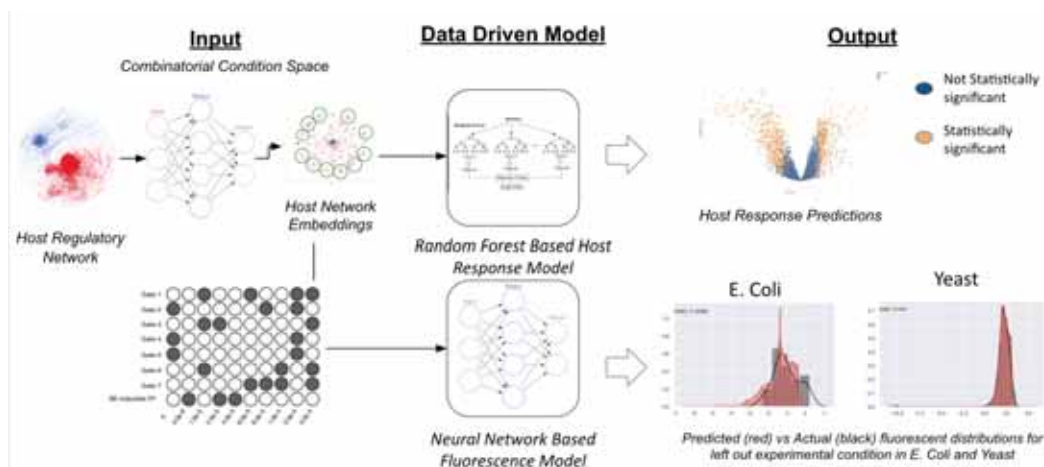


Figure 22 Combinatorial Design Model

The models were tested in their predictions of host response and circuit fluorescence for E. Coli using the NAND 2.0 Omics data, host response for Bacillus using the RNASeq data available after application of inducers, and finally circuit fluorescence model of titrating gates of yeast. In all cases, we were able to show that the circuit fluorescence model achieves >90% of its performance using only 25% of the condition space.

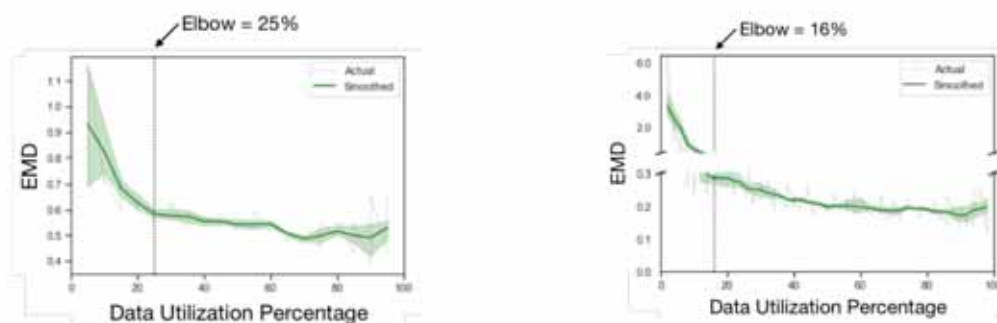


Figure 23(Left) Novel chassis NAND 2.0 Omics fluorescence predictive error measured through earth movers distance, and (right) yeast titration data.

Fluorescent predictions for a particular condition between E. Coli and Yeast are shown below.

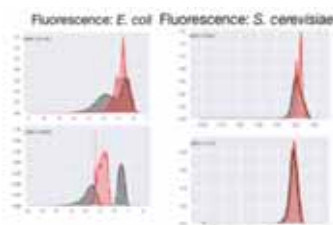


Figure 24 Fluorescent predictions for a particular condition between E. Coli and Yeast

Using the same training framework and network embedding technology developed in previous quarters, we trained a model to predict the differential expression of each gene of a host. The network that was embedded was a regulatory network that was published for each organism, in

this case E. Coli and Bacillus. The model was trained to predict impact of engineering a combination of gates for E. Coli and a combination of inducers for Bacillus. The predictions are shown below:

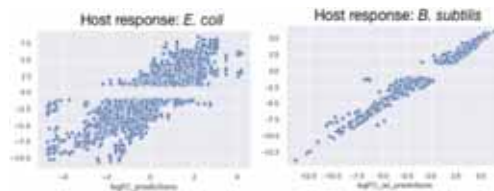


Figure 25 Predictions on impact of engineering a combination of gates for E. Coli and a combination of inducers for Bacillus

For the live/dead analyses, we generalized our model and presented how different definitions of dead can have an impact on the performance of the model. The current architecture is depicted below:

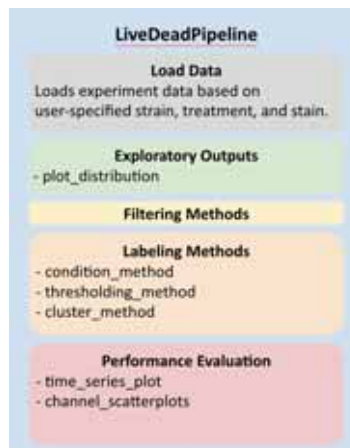


Figure 26 Architecture for Live/Dead Analysis

We tested three forms of dead:

1. Thresholding of a sytox stain
2. Condition based label of live and dead where a condition is a tuple of (ethanol concentration, time point).
3. A clustering algorithm that was used to label live/dead based

The plots below show the impact on the population of dead predicted cells over time given some definition:

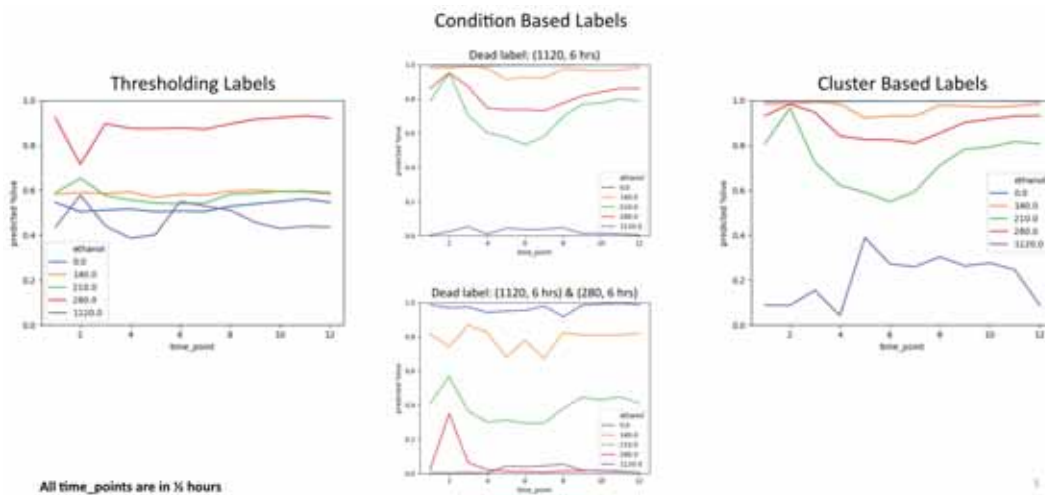
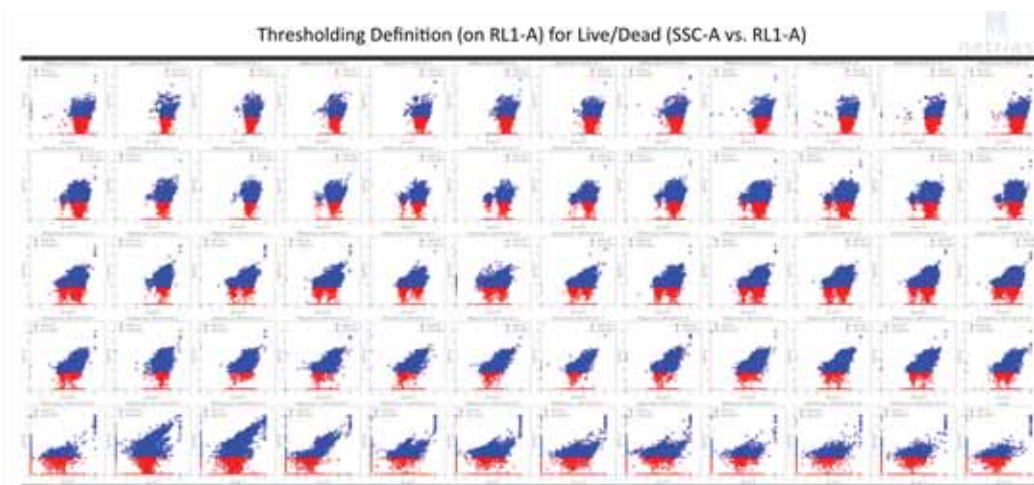


Figure 27 Plot showing the impact on the population of dead predicted cells over time

Evaluating which definition is more *accurate* becomes difficult without a notion of ground truth. We thus use two forms of expert driven validation:

1. As time increases, do we see more dead cells for higher concentrations of ethanol.
2. An expert reviews the scatter plots of the channels and provides feedback on whether the labels meet expectations. The scatter plots for the three definitions above are shown below.



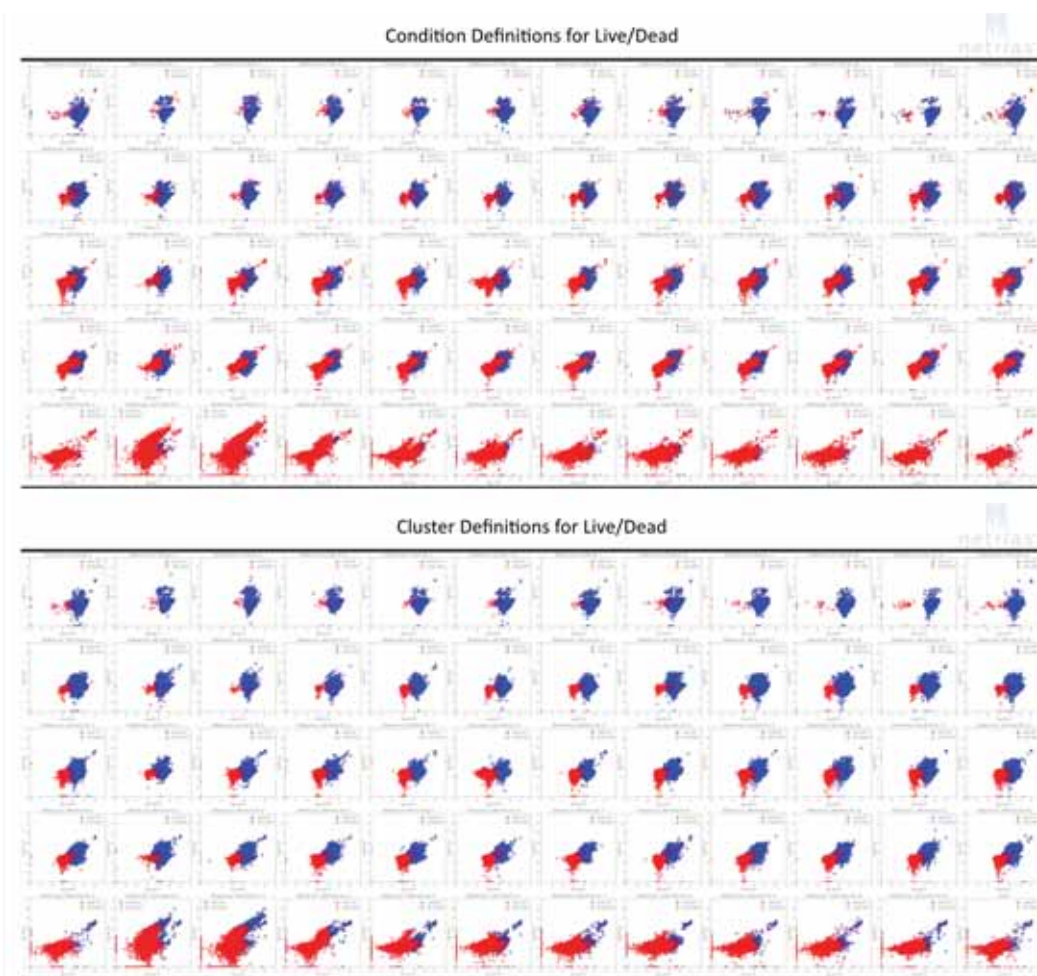


Figure 28 Threshold, Condition, and Cluster Definitions for Live/Dead

Upon reviewing these plots with an expert (Steve Haase), we derived a method to better objectively compare condition-based definitions and cluster-based definitions as they both qualitatively aligned with an expert understanding of dead. The method uses a complementary method of measuring cell population death using colony forming units (CFUs) and select the best set of conditions or definition of death that align with CFU measures.

4.9 AutoGater

AutoGater's predictions are shown in Figure 29: A and B show the events that AutoGater predicted to be healthy cells, while C and D show the events that it predicted to be dead, dying, or debris. In B and D. In B, AutoGater's predictions of healthy cells are overwhelmingly (~93%) on the left side of the boundary line, even though the AutoGater model has no information about the Sytox stain. In 3D, we note that AutoGater not only seems to eliminate dead cells that have high Sytox values, but also many dying or dead cells that are not captured by the stain-based method.

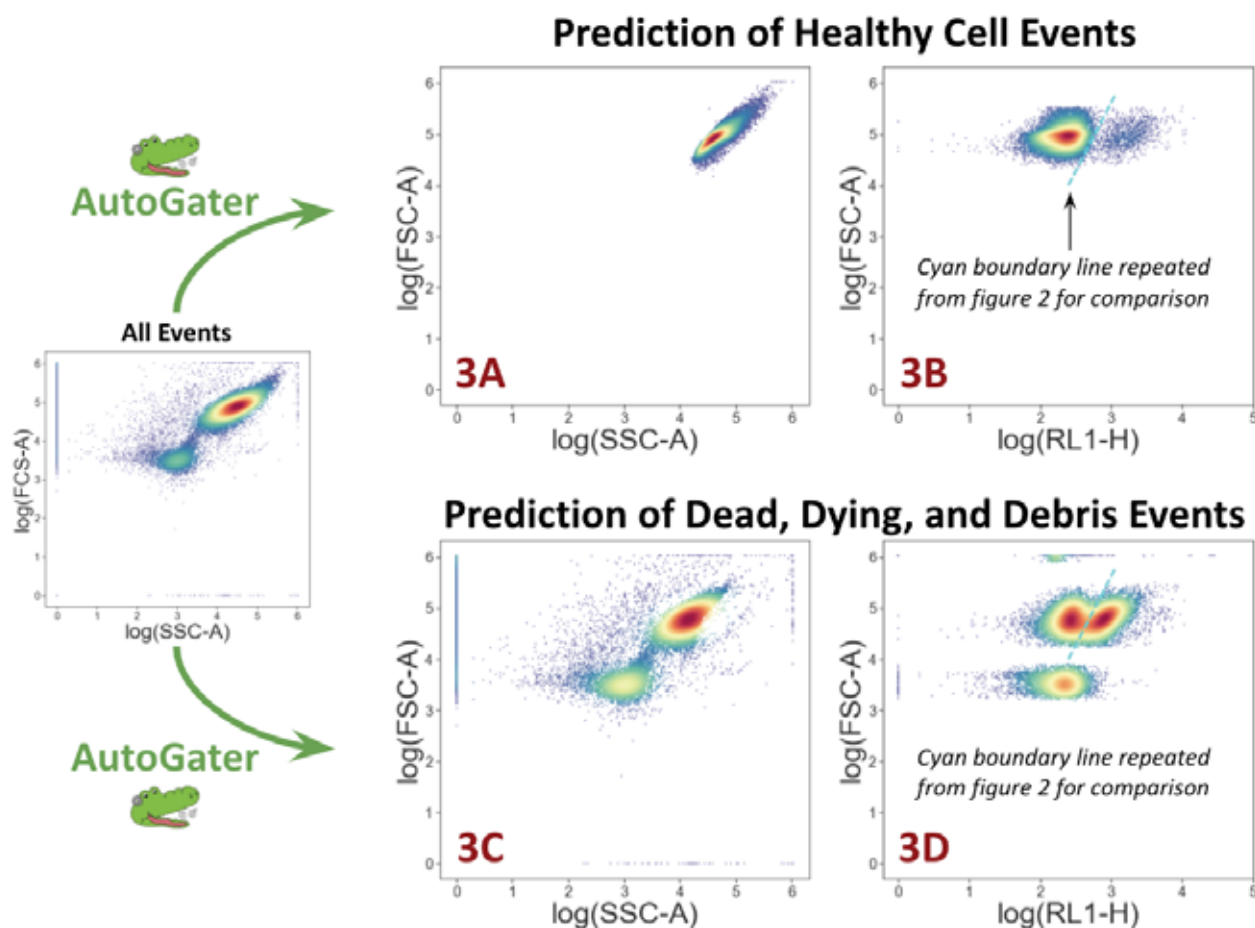


Figure 29 Using AutoGater to automatically select viable cells

The above is using AutoGater to automatically select viable cells. **A:** Events that AutoGater predicted to be healthy on side-scatter and forward-scatter axes. **B:** Events that AutoGater predicted to be healthy on stain (Sytox height) and forward-scatter axes. **C:** Events that AutoGater predicted to be dead, dying, or debris on side-scatter and forward-scatter axes. **D:** Events that AutoGater predicted to be dead, dying, or debris on stain (Sytox height) and forward-scatter axes.

The figure below compares percent-live values derived from AutoGater predictions, stain-based predictions, and CFUs. AutoGater predictions align better with CFU measures and account for their additional context not captured by stains. Overall, the results show AutoGater's ability to capture debris and different kinds of dead or dying cells that would otherwise be counted as live cells by a traditional stain-based method.

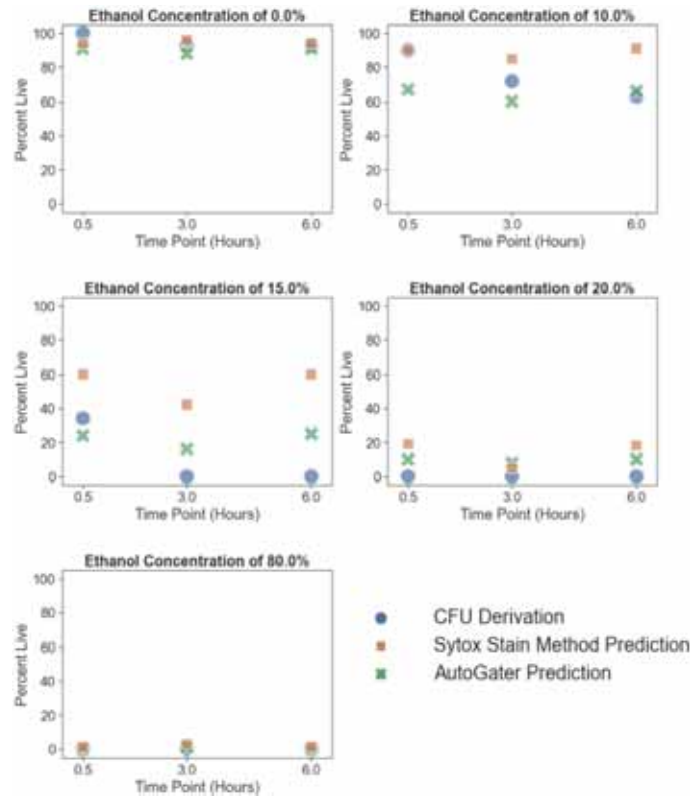


Figure 30 Comparison of the traditional stain-based method versus AutoGater

Comparison of the traditional stain-based method versus AutoGater at the population level, measured by predicted percentage of healthy cells at different experimental concentrations of ethanol and time-points. Results from CFU experiments are also shown to display alignment.

We further investigated the cells that the model selected to update from live to dead and vice versa when incorporating the CFU measurements. The cells that the model was least confident about (those with $p \sim 0.5$) are the ones that had their labels changed by incorporation of CFUs.

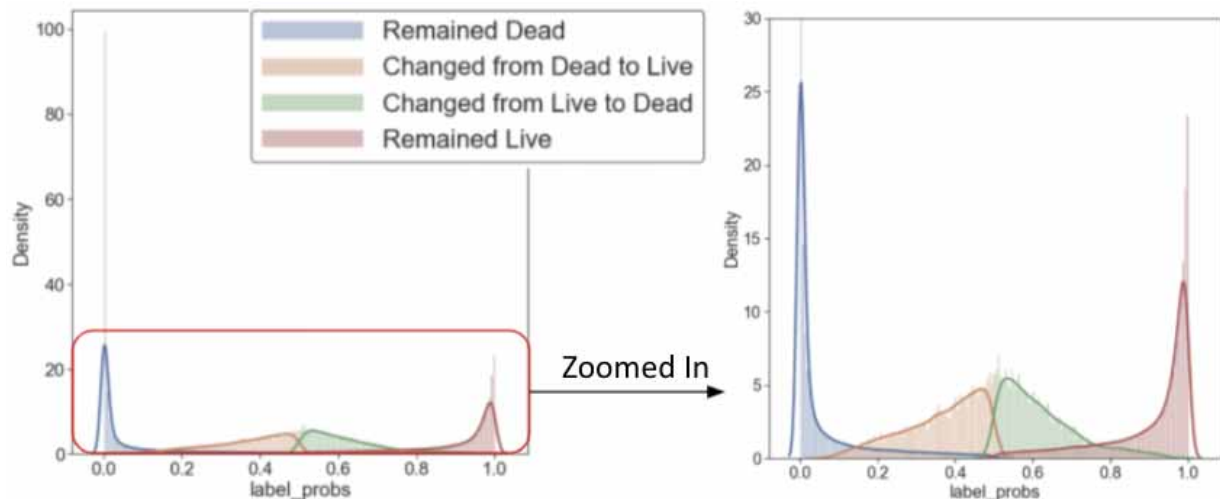


Figure 31 Events that were changed from Live to Dead by the CFU objective

Events that were changed from Live to Dead (or Dead to Live) by the CFU objective with the model were overwhelmingly points that the model without the CFU objective was uncertain about ($p \sim 0.5$).

Our success demonstrates the ability to filter out dead cells from virtually any flow cytometry experiment without needing to make use of additional experimental methods that are costly and time consuming.

4.10 Test Harness

Early in the program, with data generation and dissemination in a disorganized state, and model building and evaluation lacking universally agreed-upon standards, it was very difficult to determine which models should actually be used to drive successive rounds of iteration. We built a Test Harness that allowed users to upload models; it would automatically evaluate those models according to data sets and splits common across the program, and publish the results on a web page. In the early days of the program, this allowed rapid evolution of models by allowing a focus on those that were truly succeeding, rather than those that had been run on different (or buggy) data sets or split.

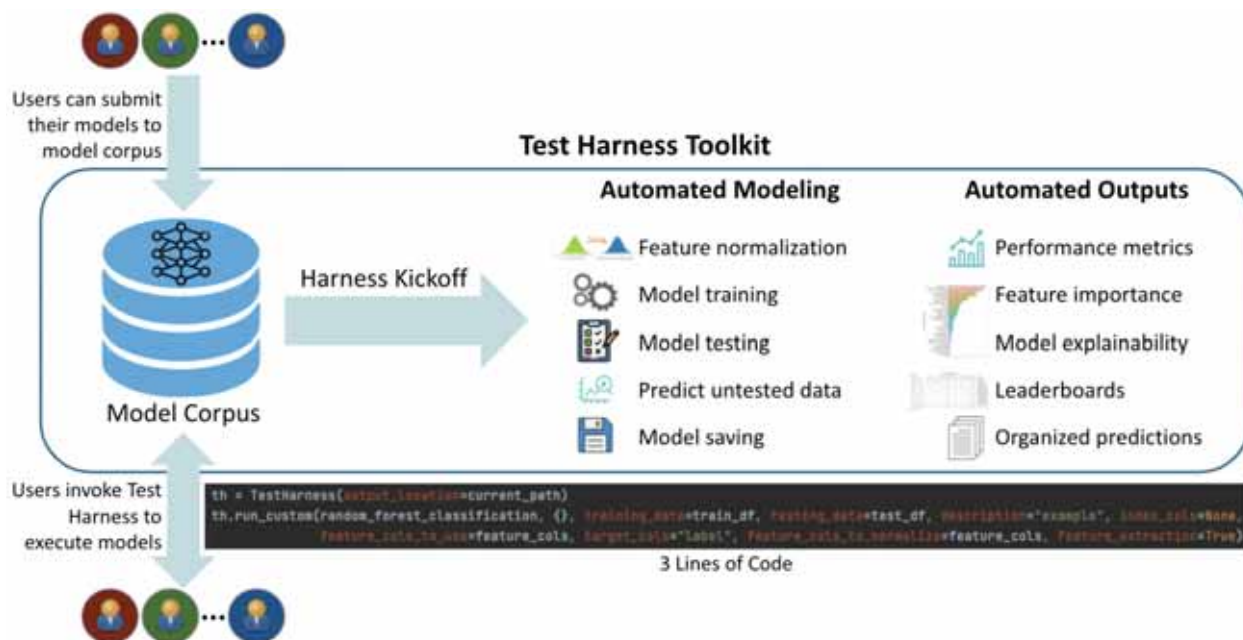


Figure 32 Test Harness Toolkit

5 CONCLUSIONS

We built analytical tools that supported all four challenge problems and deployed them to TACC and to other end-users. They increased the rate of scientific discovery, and in many cases, themselves yielded scientific discoveries.

These include tools and models to enhance design in cell-based systems. Our models overcame shortcomings of previous approaches by focusing on making predictions in large experimental design spaces where making measurements was infeasible. Experts then were able to use our predictions to either build new genetic circuits in *E. coli*, *B. subtilis*, and yeast, or to adjust experimental design conditions (such as inducer concentrations) based on our predictions. We have published these tools in a variety of journals and conferences. Discussions to enhance our platforms with future collaborations are underway.

These tools also include models to predict protein stability and protein binding. We used them to help drive selection of TA2 designs, to refine TA2 designs, and to refine our own designs. We also published dramatically larger databases of protein stability and protein binding, enabling future scientists to build on our work. Experiments to test broader applications are underway in other laboratories, and publications are released or forthcoming.

A third domain for our analytical solutions began with the data sharing exercise in Summer 2019 and continued through the end of the program. Our neuroscience models demonstrated highly efficient and successful translation between mouse brains and visual stimuli or location information. These models may lead to a better understanding of brain function, more effective brain-machine interfaces, and better computer vision systems. A paper is forthcoming.

We also, particularly earlier in the program when data to drive our TA1 analyses were less available, contributed important support tools to other areas of the program. We helped prototype and/or build data management pipelines and systems, tools to empower experimentalists, and frameworks to support the operations of designers. These tools reduced error, enabled collaboration, increased the rates of data collection, and improved design success.

6 RECOMMENDATIONS

We would like to call attention to two outstanding opportunities made possible by our work in SD2.

The first area where we see the potential for big wins with additional work is in protein stabilization. We have already shown the ability to dramatically improve the stability of proteins through guided mutation. Stabilization can improve the efficacy of protein-based therapeutics and biologics, improve their shelf life and robustness to environmental extremes, and make their manufacture cheaper. Stabilization without constraint, however, has the potential to interfere with a protein's actual function. For this reason, we are tremendously excited about our experiment being run by Ellen Goldman's lab at the Naval Research Laboratory. Here we combined stabilization with structural constraint (guided by AlphaFold2, which is revolutionizing biology by accurately predicting protein structure); because structure is the key determinant of function, we hope to see that our stabilized mutants maintain their coronavirus-binding activity while exhibiting greater stability. This is an early pilot experiment in what has the potential to be an enormously fruitful area of research.

Second, our neuroscience work with Mark Schnitzer's lab at Stanford still holds tremendous untapped potential. We have had great success combining decoding with prediction to drive image reconstruction, based on early and rudimentary data. On the neuroscience side, we think that the really exciting opportunities come with better controlled data collection combined with richer (video) stimuli. If we can build models for video decoding that are as good as those for image decoding (or better, given higher temporal resolution and eye tracking, from a larger region of brain, in response to video stimuli), we can replicate or implicitly reverse-engineer the brain's real-time visual processing system. This could lead to powerful video compression, breakthroughs in computer vision, and better use of information from remote sensing platforms (e.g. for autonomous vehicles).

8 APPENDIX A – PUBLICATIONS

We have contributed to the following publications that have already been published or pre-published:

1. Eslami, Mohammed, et al. “Prediction of whole-cell transcriptional response with machine learning.” *Bioinformatics* 38.2 (2022): 404-409.
2. Estrada Pabón, Jan D., et al. “The Role of Configurational Entropy in Miniprotein Stability.” *The Journal of Physical Chemistry B* 125.12 (2021): 3057-3065.
3. Freeman, Jared, et al. “Selecting and assessing challenge problems.” *Theoretical Issues in Ergonomics Science* 20.1 (2019): 27-38.
4. Leiby, Nicholas, Ayaan Hossain, and Howard M. Salis. “EPiC Series in Computing.” *Proceedings of the 12th International Conference*. Vol. 70. 2020.
5. Singer, Jedediah M., et al. “Large-scale design and refinement of stable proteins using sequence-only models.” *bioRxiv* (2021).
6. Zaitzeff, Alexander, et al. “Improved datasets and evaluation methods for the automatic prediction of DNA-binding proteins.” *Bioinformatics* 38.1 (2022): 44-51.

The following manuscripts, attached to this report, have not yet been published or pre-published:

1. AutoGater: A Weakly Supervised Neural Network Model to Gate Cells in Flow Cytometry
2. A Toolkit for Enhanced Reproducibility of RNASeq Analysis for Synthetic Biologists
3. Reconstruction of Natural Visual Scenes from Primary Visual Cortical Neural Activity Using Adversarial Learning
4. Comparison and evaluation of data-driven protein stability prediction models

AutoGater: A Weakly Supervised Neural Network Model to Gate Cells in Flow Cytometry

Abstract

Flow cytometry is a popular method for the rapid characterization of a cell population based on the optical and fluorescence properties of individual cells. Ideally, the cell population would consist of only viable cells, however this is most often not the case. Separating out healthy cells from dying and dead cells, and any potential debris, is an important first step in analysis of flow cytometry data. While gating of debris can be conducted using measured optical properties, identifying dead and dying cells often requires utilizing fluorescent stains (e.g. Sytox, a nucleic acid stain that stains cells with compromised cell membrane such as dead cells) to identify cells that should be excluded from downstream analyses. These stains prolong the experimental preparation process and use a flow cytometer's fluorescence channels that could otherwise be used to measure additional fluorescent sources within the cells (e.g. reporter proteins). We aim to identify viable cells for downstream processing by gating cells that are dying or dead without using stains. AutoGater is a weakly supervised deep learning model that can separate healthy populations from unhealthy and dead populations using only light-scatter channels. In addition, AutoGater harmonizes different measurements of dead cells such as Sytox and CFUs.

Introduction

Flow-cytometers are instruments that can provide single-cell based measurements of components that are fluorescently tagged. They are critical instruments for a range of applications from synthetic biology [CITE], gene therapy [CITE], and biomarker discovery [CITE]. Flow cytometers provide an opportunity for downstream experiments or analyses to focus on events of interest, such as live and healthy cells for applications in synthetic biology. This process is known as gating. There are a variety of tools that support the gating process from manual, expert driven selection of events [CITE], to data-driven based clustering methods [CITE]. Recently, there has been a considerable amount of research in using neural network models to identify cell populations of interest [CyTOF, CellCNN]. While these new models are trained with the abundance of data present in flow cytometry, they do not directly encode information about each event for the model. Instead, they rely on sample information, such as disease not-disease, and show that filters learned can correlate with known phenotypic signatures (such as CD8+ cells).

This problem is further complicated when the phenotype of interest does not have a clear, consolidated definition, such as live and dead cells. Current methods of differentiation between live and dead yeast cells mainly rely on adding stains to cells prior to usage of a Flow Cytometer. There are two stain types that are commonly used, both of which rely on the fact that cells with broken membranes usually die ¹. The first type of stain can only penetrate into cells that have broken membranes, allowing for dead or damaged cells with ruptured membranes to be identified ¹. These kinds of stains include Sytox ², propidium iodide, ethidium bromide, trypan blue, or erythrosine B ¹. The second type of stain can penetrate live cells as well, but are able to be pumped out or reduced by living cells while dead cells cannot do the same ¹. Examples of these stains include phloxine B or methylene blue ¹. Staining methods allow researchers to distinguish between live and dead cells at an individual cell level, but they can be time consuming, expensive, and add complexity to the experiment. In addition, usage of stains requires one of the Flow Cytometer's fluorescence channels to be dedicated to measuring stain presence, thus making that channel unavailable for other usages. A final limitation of supplemental staining is that it can only identify dead cells that have ruptured membranes and not other types of dead or dying cells.

Another category of live/dead differentiating methods is based on growing yeast cells on solid or liquid medium ¹. Perhaps the most common of these methods is Colony Forming Units (CFUs) ¹. While using CFUs is considered to be a “gold standard”, there are two main limitations to this method. First, CFUs do not measure cell death directly, but rather provide a growth inhibition measure that can be used as a proxy to measure cell death. In doing so, they fail to capture some viable yeast cells such as those who remain alive but are unable to divide ¹. The second limitation to CFUs is that they can only provide a **population-based** estimate of the percentage of live and dead cells, and cannot determine if any **individual** cell is live or dead ¹.

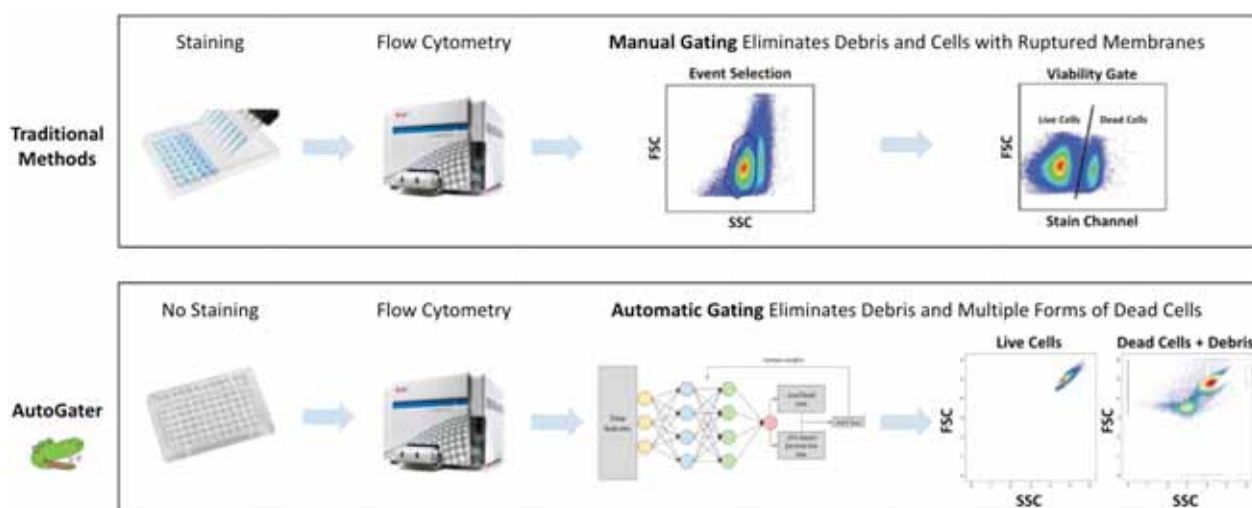


Figure A-1. Overview of the AutoGater methodology and its benefits. Manual Gating figures are from ³.

In this paper, we present a weakly-supervised neural network model trained with a novel objective function called AutoGater to support identification of dead cells without the use of stains (Figure 1). AutoGater's objective harmonizes event-based and population based notions of death into a single model. Experiments were performed to weakly label events from a cytometer as alive or dead. A two stage-model was then trained to label events from all other conditions as alive or dead. The model was validated from experiments that used two different modes of killing, ethanol and heat, and across time course measurements. The model's predictions were compared to both event based methods using stains as well as population based measurements in the form of CFUs.

Methods

Experiments to Generate Flow Cytometry and CFU Training Data

All experiments used the S288C wild-type strain of *Saccharomyces cerevisiae*. Before being treated with various ethanol concentrations or temperatures, yeast cells were grown in rich YEP medium (1% yeast extract, 2% peptone, 0.012% adenine, 0.006% uracil) containing 2% dextrose (YEPD) at a temperature of 30°C until a cell density of $\sim 3 \times 10^6$ cells/mL was reached. Upon reaching the required cell density, cells were either treated with various ethanol concentrations (0%, 5%, 10%, 12%, 15%, 20%, and 80%) or temperatures (25°C, 30°C, 35°C, 40°C, 45°C, 50°C, 55°C, and 65°C). Ethanol treated cells were kept at a temperature of 30°C. The sampling regime for ethanol treated cells was 0, 0.5, 3 and 6 hours, while the sampling regime for temperature treated cells was 0, 0.5, 1, 2, 3, 4, 5, and 6 hours. The time course and gradient of treatments resulted in 28 and 64 different conditions for ethanol and temperature, respectively. Effectively, this created a gradient consisting of 28 and 64 different levels of cell death, which we believe should capture most stages of cell death in yeast cells. Each sample was split into 3 samples: one for inoculating a plate with ~ 300 cells for subsequent CFU counting, one for staining with SYTOX Orange Nucleic Acid Stain (Invitrogen) and the last kept untreated with SYTOX. The last two samples, one with SYTOX Orange added and one without it added, were then run through an Attune NXT Flow Cytometer (ThermoFisher Scientific) with 488 nm and 561 nm lasers and fluorescent protein filter kit. The YL1 detector with the 585/16 band pass filter was used in the acquisition of SYTOX Orange fluorescence.

Two Stage Training of Neural Network with Custom Objective Function

The neural network model to gate cells goes through two stages of training. The first stage learns weights from a set of weak labels on events, while the second stage updates the weights based on information from CFU assay (Figure 2B). Another description of the two stages is that the first stage provides a set of preliminary labels to events that are then modified based on information from the additional assay.

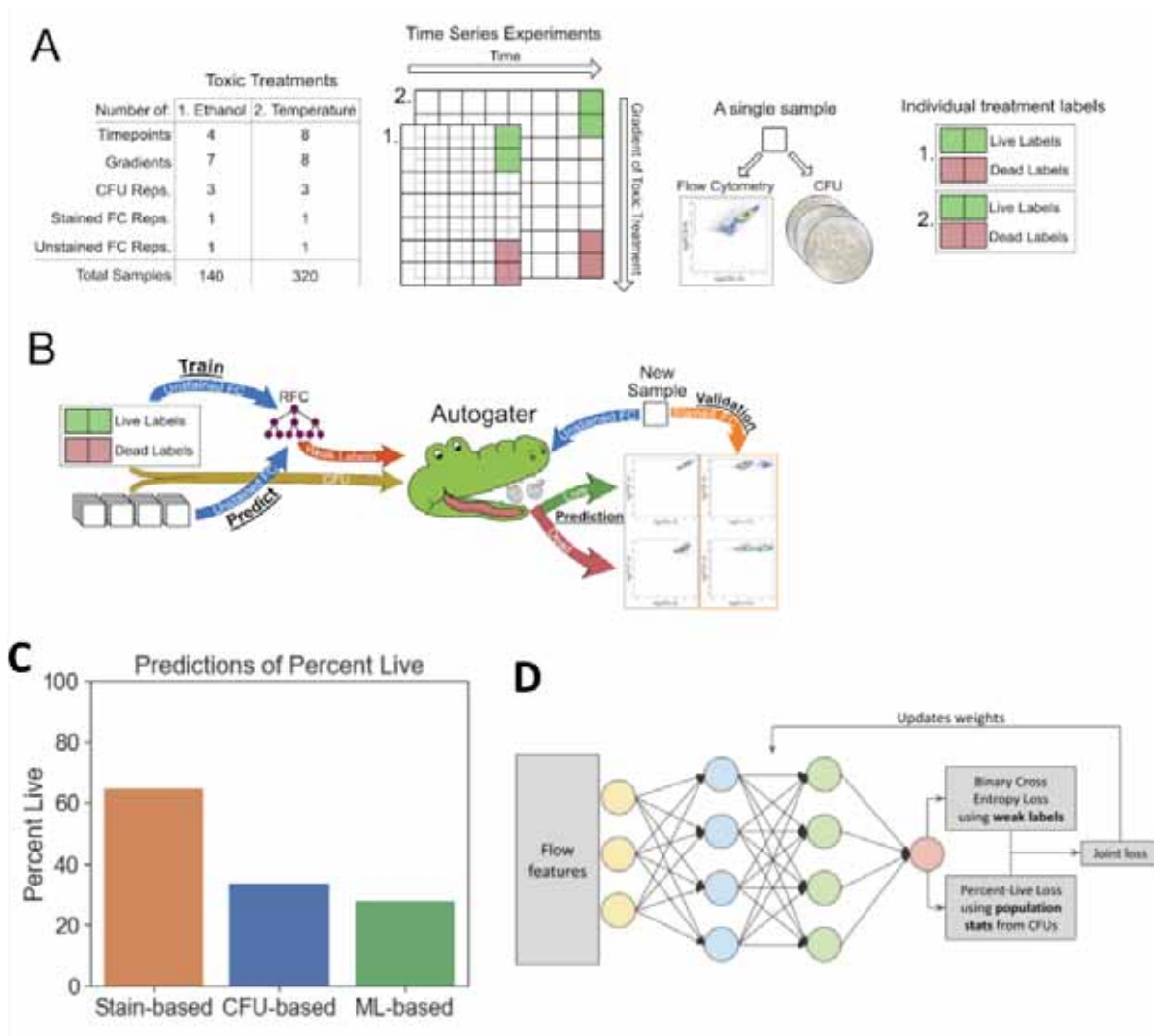


Figure A-2

Figure 2: A) Gradients of two modes of killing (heat and ethanol) were used generate data for training. Four conditions were used for training and all remaining were used for test. B) Autogater's two stage framework first predicts the set of held out conditions based on the weak labels, and then adjusts those predictions based on the CFU data at that condition. C) The three different methods provide three very different assessments of live cells for the same sample. Methods are needed to harmonize across these different assessments. D) Architecture of a deep learning model that takes non-color channels from a flow cytometer as input (FSC, SSC) and is trained to jointly optimize cell-based and population based notions of death.

Stage 1: Weakly Supervised Model for Annotating Dead Cells

The first stage of training relies on a set of weak labels attached to every event. In the case of the live/dead phenotype, we select a subset of experimental conditions to serve as "live" and "dead" conditions. Most of the 28 and 64 different conditions for ethanol and temperature contained a mixture of live and dead cells making this labeling effort by hand impossible.

However, a subset of the conditions did contain mostly, if not all, live or dead cells. The conditions containing all live cells were the 6-hour timepoints of 25°C, 30°C and 35°C for the temperature treatments and the 0-, 0.5-, 3-, and 6-hour timepoints of the 0% ethanol treatment. These are typically permissive conditions for yeast cell growth and the CFUs for these conditions support this notion. The conditions containing mostly dead cells were the 6-hour timepoints of 55°C and 65°C for the temperature treatments and the 6-hour timepoints of the 20% and 80% ethanol treatments. The selection of these conditions for dead labels was based on CFU support and on the idea that enough time had passed for cells to be completely dead. Both the 20% and 80% ethanol conditions were used as dead labels as we wanted to include an extreme condition of death (80%) along with a milder condition of death (20%).

The final time point was selected for stability and the specific gradients were selected as live and dead controls. The labels are called weak labels because the underlying hypothesis of the training regimen is that the majority of events will be labeled correctly, but it does not require all the events to be labeled correctly. In other words, we select control conditions that will have the expected majority of cells to be alive and dead. The reason we select the two highest gradients of treatment is to provide the model context with different forms of dead - ones that either change or keep the cell's morphology intact. A model is then trained with six channels as input (FSC-A, FSC-H, FSC-W, SSC-A, SSC-H, SSC-W) and is applied to all other conditions to provide all events with a preliminary set of labels. This model can be a neural network or any other machine learning model that can provide a set of preliminary labels to data from all other conditions. We chose a random forest model for this first stage as it was significantly faster for training and inference. This model can be compared to stain based approaches of identifying live and dead cells.

Stage 2: Training Neural Network with Custom CFU-Based Objective

The second stage of training uses a fully-connected neural network model to update the labels based on a CFU measurement per condition (Figure 2D). Specifically, the objective function of the model can be divided into two parts:

$$L(y, y'; \text{CFU}, \text{condition}) = \text{BCE}(y, y') + |\text{sum}_i(y_i') / \text{number_of_events} - \text{CFU}|$$

The first term seeks to reproduce the labels that were learned by the first stage model while the second term updates the weights based on context from the CFU from each condition. Given the objective selects a step per batch, the batch size for this model needs to be larger than what one would usually use (we used 2048 versus common batch sizes of 32 or 64). This ensures that each batch has a sufficient number of events per condition. The model can now be used on any other data to annotate live and dead events.

Evaluation of AutoGater with Held out Experiments

Given two stages of training and the abundance of data generated from a large corpus of experiments, we evaluated the model in two ways:

1. **Event based comparison with stained data:** Event based comparison of model to manually using a Sytox stain. In this test, the model was trained with data that was stained, but the color channel that measured the stain was not provided as a feature to the model for training. In this way, we could identify the events that the model predicts as live and dead and correlate those events with the stain color channel.
2. **Population based comparison with unstained data:** Another goal of the modeling framework was to harmonize the different methods in which cells are considered alive and dead. In this case, we trained a model with unstained data and compared percent live and dead to the same experimental conditions with stained and CFU data. In this way, we expect to see AutoGater's predictions fall in between the stained and CFU percentages as the loss function jointly optimizes both terms.

As a reminder, the modeling framework presented selects a subset of conditions to label as *live* and *dead*. A model is trained with those conditions and then evaluated on all held out conditions. The CFU measurements at each condition then updates the labels to harmonize the definition of live/viable cells across both stain based and CFU measurements. Conditions that measured CFU were never held out and are only used during training of the model. We should make clear that while the experiments were conducted over time for both heat and ethanol treated cells, time was never a feature that was input to the model. Thus, any temporal trends discovered by the model are an indirect artifact of the FSC and SSC features, as well as the CFU measurement at the point in time.

Results

Autogater Successfully Identifies Live/Dead Regions

For the event based comparison of cells, our goal was to identify if the model would be able to identify events that would be gated out by an expert as debris or dead cells. As a reminder, in this instance, the model was trained with data from cells that were stained with Sytox, but the color channel that measured Sytox was not used as a feature for the model. Figure 3A shows how an expert would select live and dead events using the Sytox stain and how that compares with what the model selects as live and dead. We should note that the boundary line is not produced in the predictions, it is simply reproduced from the manually selected set of events for comparison. Over 90% of Autogater's predictions of live events are on the left of the manually selected set of events. Interestingly, ~60% of Autogater's predictions of dead events are also to the left of the line. Recall, that Autogater looks to balance both event based labels as well as CFUs for which we already saw a significant discrepancy in Figure 2C.

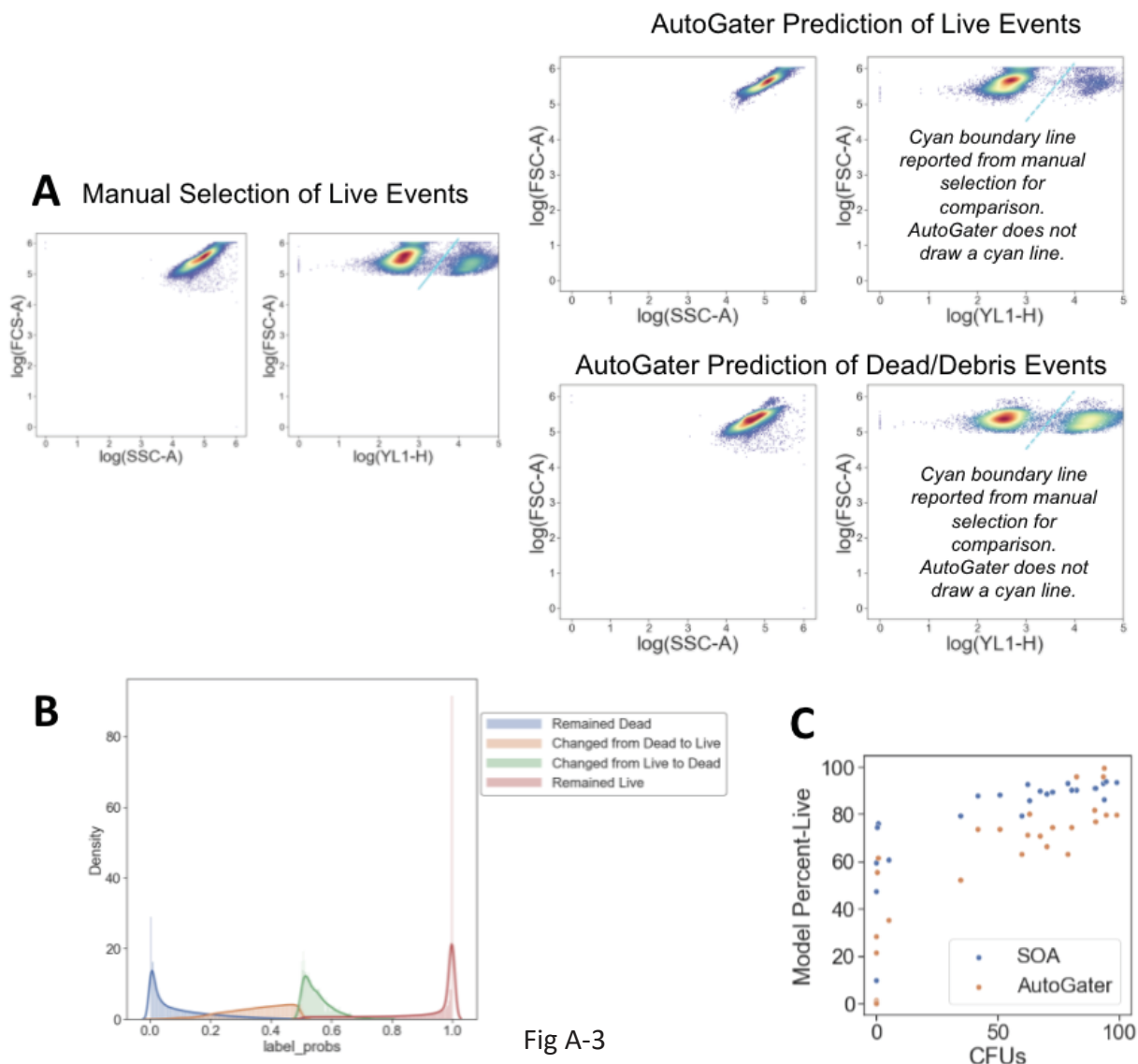


Fig A-3

Figure 3: (A) Testing the model on a sample it was not trained on shows Auto-gater is able to reconstruct the population of healthy, viable cells versus events that should be gated. Note: While and show the color channel, this channel was not provided as input to the model for its prediction. (B) Events that the model selected to update from live to dead and vice versa when incorporating the CFU measurements were ones that a model that did not use CFU information was least confident about (those with $p \sim 0.5$). (C) Using CFU as the ground-truth (x-axis), we compared the stain-based method (SOA) and AutoGater predictions to the CFUs and see that without using stains, AutoGater is more restrictive than stain-based methods to classify a cell as a live-cell. AutoGater's predictions align better with CFUs at the lower and medium values of CFUs, but it's restrictive nature gates out more events than needed at higher-CFUs.

Given the number of events classified as dead to the left of the line, we wanted to see which events did the model think are dead but the stain would indicate they are alive. Recall, the second stage of the training framework includes the CFU data to refine the labels that come from the random forest model trained in Stage 1. Therefore, it is not surprising that there are events to the left of the line that model would consider dead. Figure 3B shows that most of AutoGater's changes (the orange and green curves) are events in which the first stage of AutoGater, namely the ones that did not use information from CFUs, was unsure (probability of ~0.5). We conclude from this analysis that using the CFUs to update the labels uses events that are the most unsure while it preserves the confident predictions.

To further measure the correlation between the manual selection (SOA), AutoGater, and CFUs, Figure 3C shows that AutoGater better accounts for CFUs in the mid to large range. The R^2 between CFUs and SOA vs AutoGater is 0.05 versus 0.55, respectively. This shows a significant impact on CFU information in AutoGater's ability to identify dead events and harmonize event based and population based measurements of death. While not shown in the plot, we also measured the correlation between CFU and AutoGater Stage 1 and Stage 2. Stage 1 only uses the weak labels while Stage 2 accounts for the CFU data. In this case, we observed an R^2 of 0.4 and 0.55, a 37.5% increase in performance. The impact of the CFU data for training is noticeably evident and enables AutoGater to harmonize the two measurements by only using forward and side scatter channels.

A focus on Ethanol Based Killing of Cells

For all methods of screening live/dead cells in ethanol treated populations, a trend of increasing death occurs as ethanol concentrations are increased as well as treatment time increases (Figure 4 Panels A-F). Additionally, as the ethanol concentrations increase, the rate at which cells die increases.

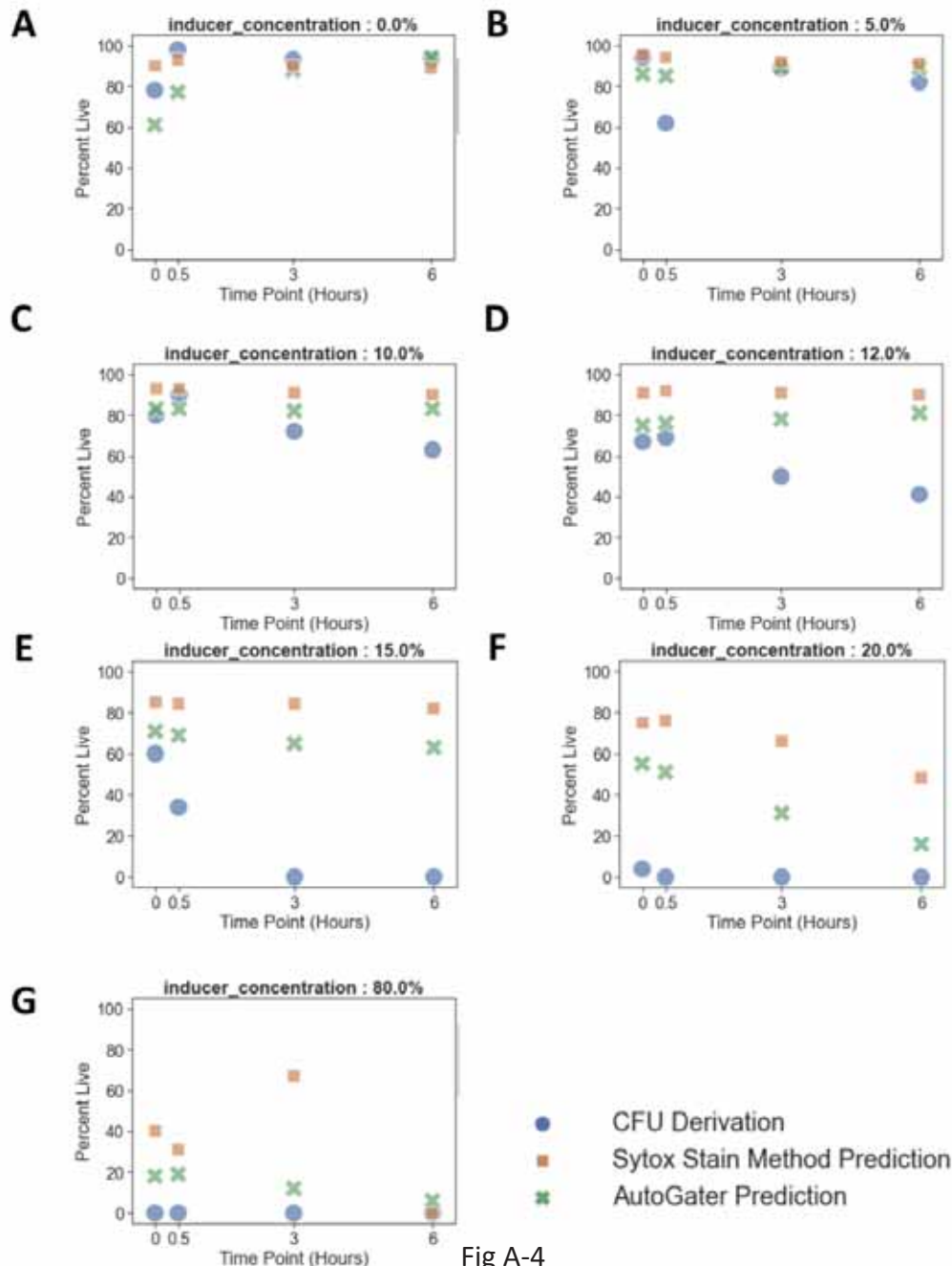


Fig A-4

Figure 4: Comparison of various methods (colors and style) to measure dead over time (x/y axes) and ethanol concentrations (panels A-F). CFU derivations are population based measures of death, Sytox stained are cell based proxy measurements of death, and AutoGater are predictions from the neural network that use a weakly supervised approach to refine condition labels of death with those of CFUs. In this case, the live label was the final time point at 0.0% ethanol, while the death was the final time point at (20% and 80%) ethanol. AutoGater is able to learn a method to tag dead cells that combines the cell based measurements and population based measurements. This is best observed in panel (F).

However, the actual rate of death between methods does differ as the percent of cells determined to be alive by each method mostly differ for each sample. As ethanol concentrations increase, the CFU method is typically the first to respond, followed by AutoGater and then STYOX. This phenomenon is most notable in ethanol concentrations 15% and 20%. The fact that AutoGater can capture the trends is interesting as AutoGater does not take in time as a feature and therefore has no concept of time. Additionally, AutoGater's predictions on what cells are alive mostly fall in between SYTOX's and CFU's estimates of live cells. SYTOX is a single cell perspective on cell death and CFU is a cell population perspective on cell death. AutoGater was designed to use both perspectives, single cell and cell population, to assess cells as live or dead. The fact that AutoGater's predictions on what cells are alive fall mostly between STYOX's and CFU's estimates on live cells suggest that AutoGater is indeed incorporating both perspectives in its predictions.

A Focus on Heat Based Killing of Cells

All three methods for screening live/dead cells in temperature treated populations displayed similar trends in their estimates of live cells as their estimates of live cells in ethanol treated cells. The trend being an increase in cell death as temperatures increased and as time increased. This trend is perhaps even more apparent in the temperature treated experiments as they are in the ethanol experiments (Figure 5).

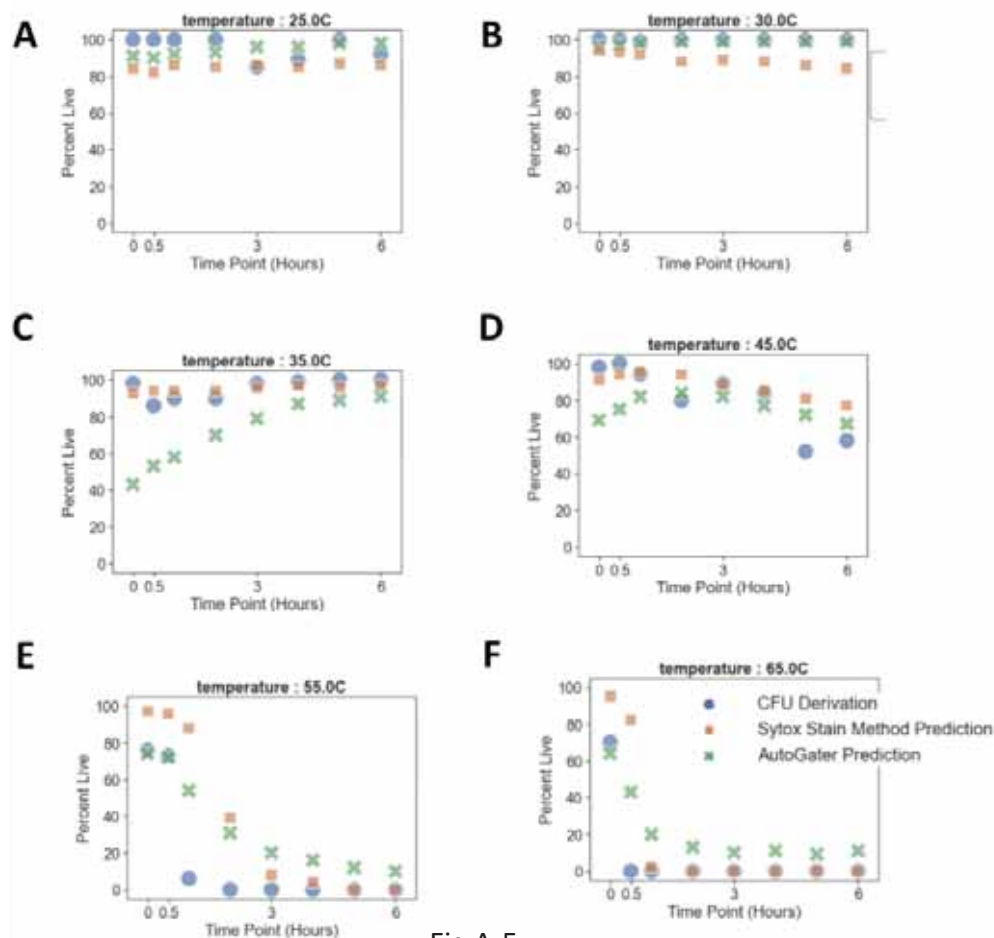


Fig A-5

Figure 5: Comparison of various methods to measure dead over time and temperature. CFU derivations are population based measures of death, Sytox stained are cell based proxy measurements of death, and AutoGater are predictions from the neural network that use a weakly supervised approach to refine condition labels of death with those of CFUs. In this case, the live label was the final time point at 25C, 30C, 35C ethanol, while the death was the final time point at 55C and 65C. AutoGater is able to learn a method to tag dead cells that combines the cell based measurements and population based measurements. This is best observed in panel (F).

Additionally, AutoGater's percent live predictions mostly fall in between STYOX's and CFU's percent live estimates as they did for the ethanol experiments. This suggests again that AutoGater is considering both the single cell and cell population perspectives of cell death. The samples treated at 40°C and 50°C seem to be an exception to this as CFU and SYTOX show an increase in cell death over time, however, AutoGater predicts an almost constant percentage of cells remaining alive over the course of both experiments. AutoGater learns and predicts live and dead cells using the FSC and SSC channels. Upon inspection of the data from these channels across all temperature treated samples, it was found that the samples from 40°C and 50°C had

significantly large populations of events at the saturation point on the FSC channel. This observation led to filtering out all events at this saturation point and retraining AutoGater for subsequent predictions. AutoGater predictions for these two temperatures were not improved suggesting some other phenomena could be at play.

Discussion

AutoGater provides two primary advantages for synthetic biologists that are interested in using flow cytometry to measure fluorescence of a reporter channel. First, it harmonizes two complementary definitions of death into a single model. This allows healthy viable cells to be gated and used for downstream analyses thereby providing a better set of data to characterize fluorescence. Second, and more importantly, once trained, AutoGater does not need a fluorescent feature to identify healthy, viable cells. This frees up a color channel in the cytometer, which at times could only have three channels, to be used for other purposes.

AutoGater's two-stage training framework, namely the weakly supervised neural network approach, is also not limited to identifying healthy viable phenotypes. We envision this training framework to be extended to any application that needs single-cell based predictions for a higher level phenotype. The weak labels provide the model the ability to learn the patterns across channels that make up the phenotype, a technique often reserved for experts to gate based on fluorescence and scatter channels. The joint objective function can also be extended to incorporate additional population based measures for the phenotype.

Finally, we should note that given the fact that AutoGater only uses forward and side scatter channels, it will not transfer well across organisms (Supplementary Figure 1). We tested the first stage of training for AutoGater across multiple single-cell organisms (*Escherichia coli* MG1655, *Bacillus subtilis* Marburg 168, and *Saccharomyces cerevisiae*) and found the performance to have 50% accuracy, or random chance, across organisms. Furthermore, the transfer across prokaryotes and eukaryotes was noticeably worse.

References

These need citations

1. <https://www.biorxiv.org/content/10.1101/680975v1>
2. https://www.researchgate.net/publication/340984084_Classification_of_dead_and_living_microalgae_Chlorella_vulgaris_by_bioimage_informatics_and_machine_learning
3. <https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=6964&context=etd>
 - a. Someone's PhD Dissertation on manual live/dead gating versus automated via ML. See where we can fit this.

Bibliography

1. Kwolek-Mirek, M. & Zadrag-Tecza, R. Comparison of methods used for assessing the viability and vitality of yeast cells. *FEMS Yeast Res.* **14**, 1068–1079 (2014).
2. SYTOX™ Red Dead Cell Stain, for 633 or 635 nm excitation. *ThermoFisher Scientific* at <<https://www.thermofisher.com/order/catalog/product/S34859#/S34859>>
3. Checking Vital Signs: Don't Let Dead Cells Mislead You. *ThermoFisher Scientific* at <<https://www.thermofisher.com/us/en/home/references/newsletters-and-journals/bioprobes-journal-of-cell-biology-applications/bioprob69/cell-health-assays-flow-cytometry.html>>

A Toolkit for Enhanced Reproducibility of RNASeq Analysis for Synthetic Biologists

Benjamin J. Garcia¹, Joshua Urrutia², George Zheng³, Diveena Becker⁴, Carolyn Corbet⁴, Paul Maschhoff⁴, Alexander Cristofaro^{1,5}, Uma Saxena¹, Yi-Pei Chen⁴, D. Benjamin Gordon¹, SD2 Infrastructure Group, Mohammed Eslami³

¹Synthetic Biology Center, Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, United States

²Texas Advanced Computing Center, University of Texas at Austin, Austin, Texas, United States

³Netrias LLC, Annapolis, Maryland, United States

⁴Ginkgo Bioworks, Boston, Massachusetts, United States

⁵TScan Therapeutics, Waltham, Massachusetts, United States

Abstract

Motivation: Sequencing technologies, in particular RNASeq, have become critical tools in the design, build, test, learn cycle for synthetic biology. They provide a better understanding of synthetic designs and they help identify ways to improve and select designs. While this data is beneficial to design, its collection and analysis is a complex, multi-step process that has implications both on discovery and reproducibility of experiments. Additionally, tool parameters, experimental metadata, and normalization of data and standardization of file formats present challenges that are computationally intensive. This calls for high-throughput pipelines expressly designed to handle the combinatorial and longitudinal nature of synthetic biology. In this paper, we present a pipeline to maximize analytical reproducibility of RNASeq for synthetic biologists. We also explore the impact of reproducibility on the validation of machine learning models.

Results: We present the design of a pipeline that combines traditional RNASeq data processing tools with structured metadata tracking to allow for exploration of combinatorial design in a high-throughput and reproducible manner. We then demonstrate utility via two different experiments: a control comparison experiment and a machine learning model experiment. The first experiment compares datasets collected from identical biological controls across multiple days for two different organisms. It shows that a reproducible experimental protocol for one organism does not guarantee reproducibility in another. The second experiment quantifies the differences in experimental runs from multiple perspectives. It shows that lack of reproducibility from these different perspectives can place an upper bound on validation of machine learning models trained on RNASeq data.

Availability and Implementation: The secondary analysis pipeline can be found at <https://github.com/SD2E/rna-seq-pipeline> and the configurable differential expression analysis tools and tutorials are available at https://github.com/SD2E/omics_tools.

Introduction

Synthetic design is often an iterative process, where components are improved over time. However, despite the improvement of components, controls can often stay the same across multiple iterations (unless alterations are made to the base strain in such a way that the altered strain becomes the new controls). As such, it is important to know whether or not changes are the result of design choices, experimental variability, or other unintended perturbations due to inherent variability in the pipeline. Furthermore, researchers conducting experiments in synthetic biology often have combinatorial designs that test all conditions in replicates, resulting in sample sizes in the 1000s or greater. It is not always feasible to run all of these samples at the same time, so it is important to consider how experiments change across batches (ie. differences in execution of the pipeline, differences in experiment conditions, different technicians running the pipeline, dilutions, culturing, etc). It then becomes important to ensure that variability is low and that controls are repeatable across different batches to ensure that experiments can be aggregated and compared.

RNASeq has become a ubiquitous tool in the design, build, test, learn cycle for synthetic biology (Costa-Silva, Domingues and Lopes 2017) (Finotello and Di Camillo 2015) (Gorochowski *et al.* 2017) (Hazen *et al.* 2015) (Abbas-Aghababazadeh, Li and Fridley 2018) (Sastry *et al.* 2019). Applications include debugging design failures (Gorochowski *et al.* 2017; Park 2019; Espah Borujeni *et al.* 2020) and optimization of genetic designs (Naseri and Koffas 2020). Transcriptional analysis provides an opportunity to measure chassis response holistically and identify differentially expressed genes (Gorochowski *et al.* 2017), infer transcriptional networks or co-regulated genes (Langfelder and Horvath 2008; Taylor and Singhal 2009; Kc *et al.* 2019; Nelson *et al.* 2019), identify read through and unexpected expression in synthetic parts (Gorochowski *et al.* 2017; Park 2019; Espah Borujeni *et al.* 2020), and also infer impacts on mechanistic pathways encompassing multitude of biomolecular entities (transcripts, proteins, metabolites) (Petzold *et al.* 2015; Intosalmi *et al.* 2016; Reimand *et al.* 2019; Garrido-Rodriguez *et al.* 2021).

Collection and processing of RNASeq data, however, is a complex, multi-step process (Finotello and Di Camillo 2015; Conesa *et al.* 2016). Processing pipelines are highly parameterized, computationally intensive, and are often not comparable across experiments without significant effort in normalization and harmonization of the output measurements (Costa-Silva, Domingues and Lopes 2017; Abbas-Aghababazadeh, Li and Fridley 2018). Furthermore, these tools often require a significant amount of additional biological context to explain how and why the dataset was generated, which is often disconnected from the data itself (Pinoli *et al.* 2018; Gonçalves and Musen 2019; Wilson *et al.* 2021). Collection and processing is one aspect of reproducibility, which can be more broadly delineated into three categories: 1) biological, 2) experimental and 3) analytical (Jessop-Fabre and Sonnenschein 2019; Tiwari *et al.* 2021).

Biological variability results from different responses to similar stimuli, either as a mechanism for survival, due to responses not being strictly regulated, or due to experimental conditions being variable enough to elicit different biological responses. Experimental variability is the result of

imperfect tool precision, slight alterations in handling during cell growth, lysis, RNA extraction, library construction, or data analysis. Experimental variability can also impact biological reproducibility. To minimize experimental variability, consistency of conditions, reagents, instruments, and protocols used to grow organisms, extract RNA, and process the sample are required. Finally, analytical variability arises from either an inconsistency in the use or a lack of information about processing pipelines, tools, and parameterization used to analyze the data and extract conclusions (Li *et al.* 2011; Lazic 2016; Chavez, Ho and Tan 2017). While there are a number of consortiums that have provided guidance on methods to assess the reproducibility of RNASeq, a toolkit to enable that reproducibility for synthetic biology is seldom found (Yu 2020) (Łabaj and Kreil 2016). The toolkit provided in this paper helps mitigate analytical variability by providing a clear and consistent set of methodologies that can be applied across different RNASeq experiments. Given the pipeline provides a clear set of reproducible analytical methods, we are then able to explore biological and experimental reproducibility that can then impact analytical results.

Here, we seek to present the minimal set of information required from both the experimental conditions as well as the data processing pipelines to maximize analytical reproducibility. We first present an overview of the software infrastructure required to make analyses reproducible across RNASeq experiments. The tool outputs a set of consistently formatted datasets for downstream analyses and can be used by researchers that seek to analyze RNASeq at scale (100s to 1000s of sequencing runs). We then detail a configuration-driven pipeline that enables reproducible analysis of data that was utilized to analyze data from a large experimental condition space (e.g. many genetic designs and inducers) (Eslami *et al.* 2021). We use these tools to measure the variability of controls in multiple experimental runs with perturbing controls of two popular model organisms: *Escheria coli* MG1655 (*E. coli*) and *Bacillus subtilis* Marburg 168 (*B. subtilis*). After quantifying the variability in controls, we examine the impact of the variability on machine learning models built using two *B. subtilis* datasets collected using nearly identical protocols surveying a large space of growth conditions .

Methods

Infrastructure to Automate and Track Secondary Analysis Stages

Secondary analysis of the sequencing data involves a set of hierarchical steps that depend on the experimental design, conditions, and reference files to generate counts and normalized counts data. Given the scale of experimental conditions (e.g. genetic designs, organisms, induction conditions), software infrastructure is required to ensure analytical reproducibility of an RNASeq pipeline to be parameterized and automated to process RNASeq data at scale. The software infrastructure we have developed has three main components: 1) Applications, 2) Databases, and 3) Actors.

Applications allow the software infrastructure to capture version, any configuration parameters used during the alignment and quality control process, and input/output data identifiers. They ensure that the specific processing or analysis pipeline that was run by the application can be

run again exactly the same way, in the same environment, on any system. Tools such as Docker containers provide an opportunity for researchers to ensure their code is self-contained, shareable, and reproducible (Jackman *et al.* 2019; Perkel 2019). Applications are containerized versions for each stage in the pipeline. The main motivation behind modularization was to provide us the flexibility to re-run specific stages, if needed, without re-running the entire pipeline. Furthermore, the modularity has the added benefit of updating specific components and tracking versions of software and configurable parameters for every execution.

Databases allow the software infrastructure to store locations of raw data and processed data, metadata, and full parameterizations and provenance of processing pipelines that link the data together in a queryable manner. We had two databases used in our infrastructure. The first was a data catalog that stored links to the raw data, universal identifiers, and the experimental conditions. The second was a database specifically designed to store genetic parts known as SynBio Hub. Some experimental conditions, such as strain, parts, modifications, and media ingredients, stored links to SynBioHub (McLaughlin *et al.* 2018) instead of complete information. The links provided more context if the user was not familiar with media composition, and gate names.

Actors allow the software infrastructure to respond to events (e.g. file upload, completion of an intermediate stage) to automate the processing of data at scale. In our system, reads/writes to the data-catalog and job submissions are managed by Abaco Actors (Brookes and Stubbs 2019). Actors are lightweight, containerized scripts that are triggered in response to events. Actors do not perform computationally intensive tasks but instead oversee and coordinate job processing throughout the pipeline. Computationally intensive tasks (trimming, alignment, aggregation) are performed by Tapis Applications, a framework that provides a web-based API to manage computational workloads developed by Texas Advanced Computer Center (Cleveland *et al.* 2020). Application jobs are triggered by actors and have a unique job identification number.

Given software for each stage of the secondary analysis pipeline (pre-processing, alignment, post-processing, and quality control (QC)), our emphasis was to use the software infrastructure to ensure the pipeline is manageable and reproducible for the large datasets made available over time (Figure 1).

RNA-seq Workflow and Linkages

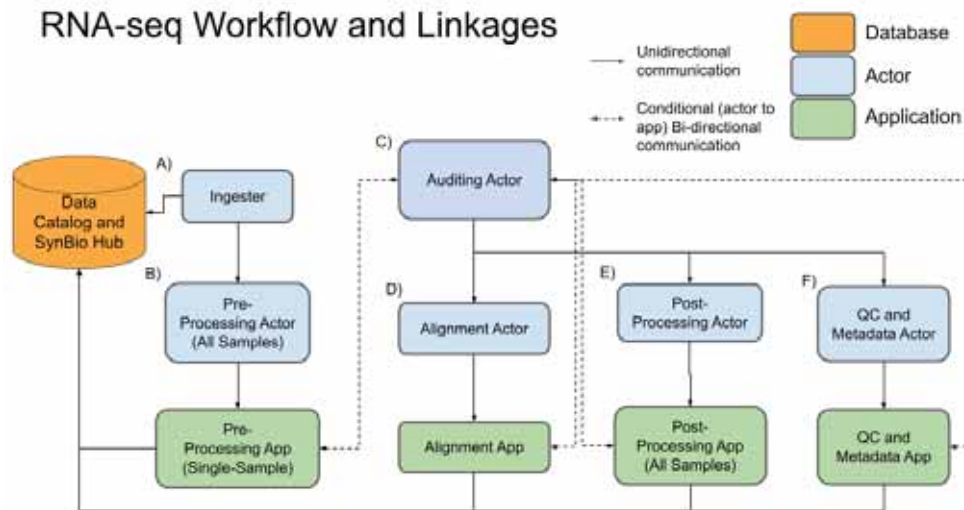


Fig A-6

Figure 1: Diagram of the RNASeq processing pipeline. A) The Ingestor monitors a metadata file uploaded to the Data Catalog to see if the experiment includes RNASeq data to trigger the processing pipeline. B) The Pre-Processing Actor triggers and stores the job ID and version of the Pre-Processing App to the Data Catalog. C) The Auditing Actor receives notifications from applications to validate the outputs and triggers the next actor. The Auditing Actor will re-submit applications up to three times to handle stochastic failures. D) The Alignment Actor queries the data catalog to determine the reference genome that is used for each preprocessed sample and triggers the Alignment Application. E) The Post-Processing Application annotates the alignments and aggregates samples into counts (raw, FPKM, TPM) dataframes. F) The QC and Metadata Actor uses metadata stored in the data catalog, and information from the logs/outputs of each job to add experimentally relevant metadata and quality control flags to the counts dataframes.

The applications used in our RNASeq pipeline consist of trimming and quality filtering raw RNA-seq data with trimmomatic (v0.36) (USADLLAB.org - Trimmomatic: A flexible read trimming tool for Illumina NGS data), and FastQC is used to generate reports for paired, trimmed reads. Preprocessed reads are aligned to an indexed reference genome with bwa (v0.7.17) (Burrows-Wheeler Aligner). For alignment there is the prerequisite that the relevant reference genome has been identified, a fasta and gff has been provided, and the linkage between sample and reference genome (strain) is enumerated in the provided metadata. After alignment, the resulting SAM files are sorted by PICARD tools (v2.18.15) function SortSam (Picard Tools - By Broad Institute), and then AddOrReplaceGroups is run on the output of SortSam. Gene-level quantification of counts is performed using the featureCounts function of Rsubread (v1.34.4) (Bioconductor - Rsubread). Annotations are performed using the GFF provided. For RNA-sequencing, we identify any CDS feature type from the GFF and annotate these in the dataframe using the "Name=" attribute. All annotated samples are aggregated into a dataframe and outputs include both raw counts and rudimentary normalization functions: TPM and FPKM. Two boolean metrics were used to measure sample quality:

1. Number of mapped reads $\geq 500K$
2. Replicate correlation of TPM values of a condition >0.8

If any of these metrics did not pass, the sample would be flagged as a low quality sample and not used for downstream analysis.

These applications can then be run individually on a high performance computing (HPC) system. For each job that is run in the software infrastructure, an actor queries a centralized database of information (the data catalog), to identify the required metadata for processing. For each job submission, there is also a record created in the data catalog enumerating the inputs, references, software versions, and outputs for a given job. Tapis applications stage assets (inputs, references, container images) to a temporary working directory, submit resource requirements to a Simple Linux Utility for Resource Management (SLURM) queue on a HPC system, archive outputs to a predefined location (aka storage system), and send a notification to an actor providing the exit status of a job (success/failure). By leveraging Tapis applications, Abaco Actors, and the data catalog, an automated pipeline is constructed that responds to events (raw data, metadata uploads), processes samples in parallel for trimming and alignment, and serially for aggregation and annotation, without the need for human intervention. This serves both to increase efficiency and throughput, and also reduce the probability for human error. Since the core components for processing are containerized, and the inputs/parameters for every job are stored in a centralized database, the results are reproducible across different machines.

Configurable Tertiary Analysis Pipeline

A challenge often faced in synthetic biology is that experimental designs are often combinatorial in nature (multiple inducers, designs, timepoints, etc); and therefore, the number of differential expression comparisons can become unwieldy and unreliable to set up and analyze in a manual fashion. Additionally, when an experiment has hundreds to thousands of different comparisons, manual processing of differential expression is often impossible, so automated solutions are necessary to analyze the data. To help facilitate the automated calculation of differential expression between the different factors, we developed a Python-based configurable toolkit, which we call *omics_tools* (Figure 2). Instead of having to write R scripts for every desired comparison, *omics_tools* automatically generates each R script, runs them, and then generates consistently labelled output. The automated process increases repeatability/reliability by standardizing output and by reducing the risk of producing manual errors, such as mislabelling comparisons, mislabelling groups, including the wrong samples, etc. Standardizing output across multiple different experiments also allows for development and use of tools that analyze differential expression results without having to generate formatting scripts for each different experiment, facilitating cross-experiment comparisons. The tool aggregates the outputs from the parallelized runs and combines all the data into a single unified dataset where each row represents a gene, its differential expression, statistical significance, and the condition for downstream machine learning. *Omics_tools* uses edgeR (Robinson, McCarthy and Smyth 2010) to conduct differential expression analysis (DEA) using TMM normalization across the set of design variables.

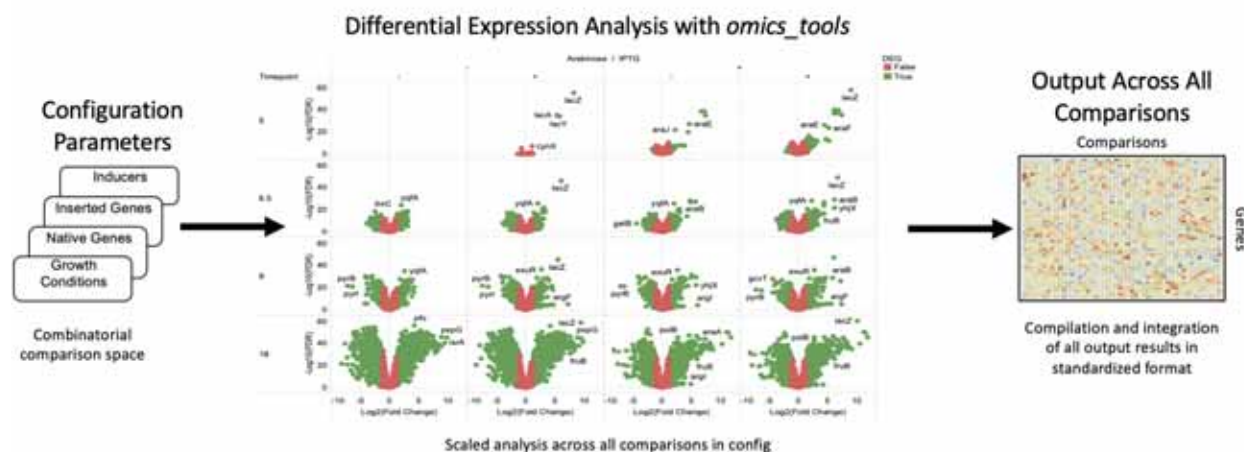


Fig A-7

Figure 2. *Omics_tools* facilitates the analysis of combinatorial design RNASeq by integrating metadata (inducers, growth conditions, genetic manipulations etc) to automatically calculate differential expression with edgeR across all conditions of interest. The tool kit allows for parallel processing of thousands of samples and comparisons in a high-throughput manner and utilizes a standardized schema to help facilitate reproducibility in analytical analysis.

Results and Discussion

Using *omics_tools* to Analyze Repeatability of Controls

To quantify the reproducibility of controls, we ran two experiments involving *E. coli* MG1655 in media at three different timepoints (5, 8, and 18 hours growth) on two different days. The timepoints were selected to represent common phases of growth in circuit characterization (Gorochowski *et al.* 2017). Four replicates were used for each timepoint on each day. Our processing pipeline and *omics_tools* were used to analyze the data for all comparisons across all timepoints. The results from this comparison are presented in Table 1.

Table A-1. Percent of the total 4097 genes assayed that were significantly differentially expressed across comparisons (sample_time, FDR < 0.05, log2FC > 2).

Timepoint	Day 1: 5 hrs	Day 1: 8 hrs	Day 1: 18 hrs	Day 2: 5 hrs	Day 2: 8 hrs	Day 2: 18 hrs
Day 1: 5 hrs	0.0%	3.8%	29.4%	0.3%	4.6%	32.5%
Day 1: 8 hrs	3.8%	0.0%	20.2%	2.4%	0.3%	24.2%
Day 1: 18 hrs	29.4%	20.2%	0.0%	27.2%	20.1%	0.2%
Day 2: 5 hrs	0.3%	2.4%	27.2%	0.0%	2.2%	29.6%

Day 2: 8 hrs	4.6%	0.3%	20.1%	2.2%	0.0%	22.6%
Day 2: 18 hrs	32.5%	24.2%	0.2%	29.6%	22.6%	0.0%

Samples from the same time point on different days were found to be most similar (average 0.3% of the genome differentially expressed), and the least similar were the 5 hr vs 18 hr comparison (avg 29.9% differential expression). The differences in 5 hr vs 8 hr, 5 hr vs 18 hr, and 8 hr vs 18 hr timepoints are as expected and can be largely attributed to differences in growth phase (log phase versus transition versus stationary), while the minor differences in the same hour comparison can be attributed to slight differences in the state of the bacteria and aggregate of error and variability across the pipeline.

We also compared different means of performing sample-to-sample normalization and identifying differentially expressed genes. The sample similarities can also be observed when using an FPKM normalization (Figure 3). We also utilized FPKMs to compute the differential expression of genes (with t-test and Benjamini-Hochberg multiple testing correction, $\log_2FC > 2$ and $FDR < 0.05$) (Supplemental Table 1) and compared those to *omics_tools*. Despite having a similar number of differentially expressed genes between the two methods, the resulting significant differential genes have a subset of genes that is not shared between the two. Unsurprisingly, the Jaccard coefficients (Supplemental Table 1) for the same time points across the two days averaged only 0.214 (range 0.083 to 0.308) due to the very low number of genes differentially expressed (10.8 on average with a range of 1 to 21). The different time point and day comparisons had much higher similarities in differential genes with average Jaccard of .598 (range 0.468 to 0.762) and an average percent of 82.6% (range 75.3% to 88.8%) of genes that were significant from both *omics_tools* and FPKM comparisons. The main difference between *omics_tools* and FPKM is from intersample normalization of TMM. While normalization and differential strategy does matter, the majority of genes that are significant are shared between the methods.

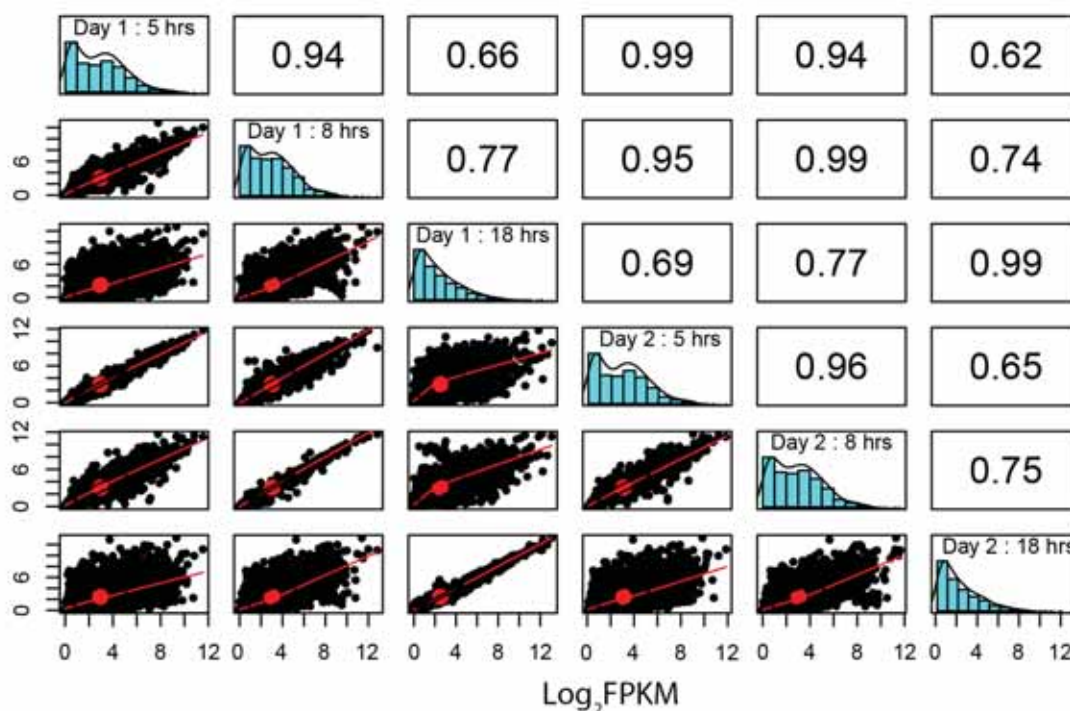


Fig A-8

Figure 3. Pearson's correlations (upper right numbers) between log2 FPKM of *E. coli* samples at each of the measured timepoints on both days. Samples taken on different days, at the same hour, are much more similar (avg .99) than at different hours. The 5 hour and 18 hour have high correlations (avg 0.95), suggesting they are in similar growth states, while the 5 hour and 18 hour have the least similar expression profiles (avg. 0.64) due to their differences in growth states. Scatterplots are gene-gene Log₂ FPKM comparisons, with a red linear regression line. Blue plots are histograms for frequencies of gene FPKMs.

To better understand how different species are affected by similar experimental conditions, we also examined data collected for *B. subtilis* controls at 5 hours on 3 different days. The data were processed through the same high throughput sequencing pipeline and analyzed with *omics_tools*. Unlike the *E. coli* strain, *B. subtilis* had much higher variability with an average of 4.87% significant differential expression (range 1.1% to 7.16%) (Supplemental Table 3) compared to 0.269% for the 5 hour *E. coli* timepoint. The FPKM correlations (Figure 4) have a similar pattern as the 5 to 8 hour *E. coli* comparison (Figure 3), suggesting that the growth rates may be different between the different days for *B. subtilis*. It is hypothesised that *B. subtilis* was more impacted by overnight growth and recovery in wells as compared to *E. coli*. The differences in dilution potentially impact growth and sporulation as well. Fundamentally, we showed that experimental conditions that produce reproducible results for one organism do not necessarily produce reproducible results for different organisms.

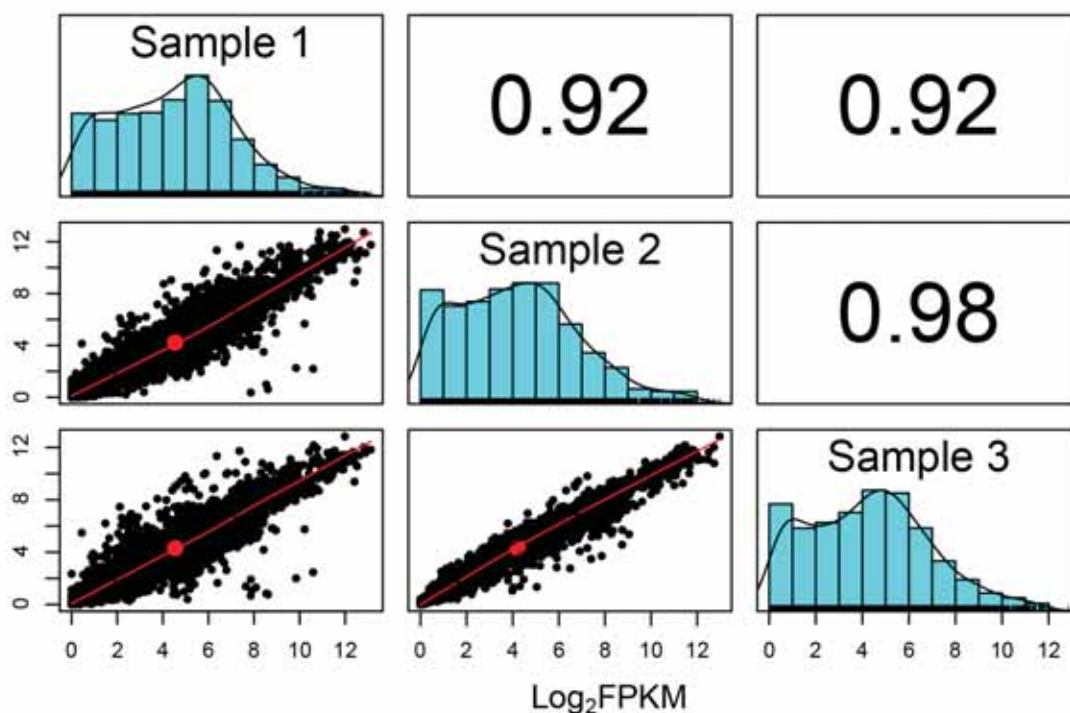


Fig A-9

Figure 4. Pearson's correlations (upper right numbers) between log2 FPKM of 5 hour time points for *B. subtilis* on each of the three different days. Samples 2 and 3 have much more similar patterns compared to 1-2 and 1-3. Additionally, these two comparisons have a lower correlation than the 5-8 hour *E. coli* comparisons, suggesting greater variability in the growth/biological conditions for *B. subtilis*. Scatterplots are gene-gene Log₂ FPKM comparisons, with a red linear regression line. Blue plots are histograms for frequencies of gene FPKMs.

The Impact of Reproducibility on Validation of Machine Learning Predictions

One challenge associated with validation of machine learning models for high-throughput experiments is regarding reproducibility of the training data. Specifically, if a model is built from a set of data that lacks or has underdeveloped biological and experimental context (ie metadata). Despite the immediate accuracy concerns, there is no guarantee that the model will generalize to future runs of experiments with similar experimental/biological context. Furthermore, extensibility of a given model has an upper bound of the experimental reproducibility of the set of conditions being modeled.

Here, we evaluate the implications of reproducibility on a machine learning model trained with response from single inducers to predict the transcriptional response to a combination of inducers. The test data was generated twice on different days to measure the impact reproducibility has on evaluation of the model. The only difference between the two experimental runs was the increase in volume of cell culture used for RNASeq from 500 uL to 900 uL to increase the amount of RNA available. We hypothesized the increase would improve the quality of the samples generated, which would result in data for conditions that would drop out. Interestingly, cell culture volume is a factor that is often not used in downstream

normalization, differential expression, and pathway analyses directly, but as we will show, does have implications for sample quality which in turn impacts all other analyses. Furthermore, the increase in volume did not unanimously increase the number of replicates/samples available for analysis either, which we found surprising. This comparison can be made at multiple levels:

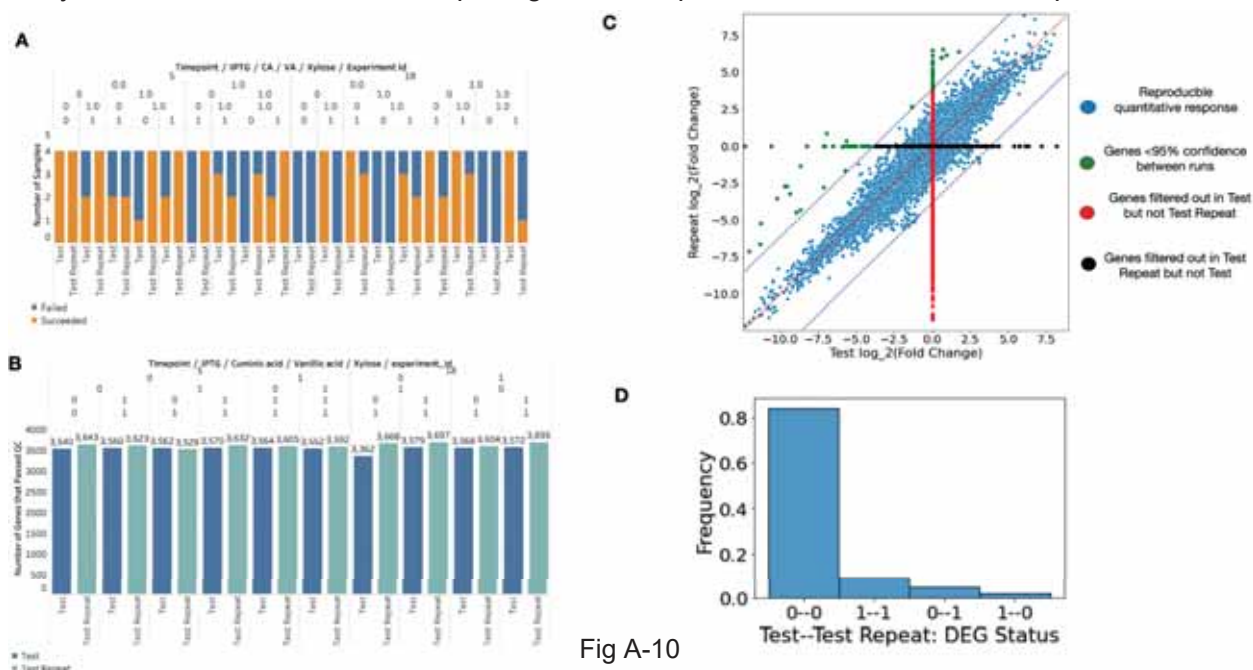


Fig A-10

Figure 5: Comparison of experimental conditions across Test and Test Repeat experimental runs. The 0/1 indicates absence/presence of inducers for panels A/B and differentially expressed status (0 not DEG and 1 is DEG for D). Each comparison will have a different impact on the experiment's ability to validate a machine learning model trained on single inducers (conditions not shown here). A: Different sample dropouts for the two runs means that if both experiments were not run, some predictions would not be validatable. B: Different genes being filtered will impact the set of genes from a condition that can be validated. C: Genes can have quantitatively different responses which can further add complications to validation and the gene dropouts (horizontal black and vertical red lines) across the two experimental runs means different genes will be validated. D: While the set of DEGs that are different between days are in the minority, these discrepancies can have mechanistic consequences on the inferences made.

Sample QC (Figure 5A) = Given the set of experimental parameters, QC can filter out a set of different samples per condition. For example, a sample induced with xylose and vanillic acid measured at 18 hours post induction has lost all four replicates in the repeated test condition. This implies that the condition will not generate the data necessary to identify how reproducible the model is at that condition. A researcher could look to see if the model predicted any impact on essential genes or pathways that can be linked to dropout, especially if the sample failed in both experimental runs. This could help the researcher determine if the dropout is biological in nature and not an experimental error. The more difficult scenario is when a set of samples pass QC in one run, but not the other. In total, there are 5/18 discrepancies across the two experimental runs, which means that the day-to-day variability could cause upwards of 27% difference depending on the experimental run that is used.

Gene QC (Figure 5B) = Tertiary analysis, such as differential expression analysis, uses expression levels across genes to identify outlying genes that do not fit a dispersion model. Genes that do not fit within a dispersion estimate are filtered out. Filtering out genes means that a model that is making gene level predictions will not have predictions of those genes validated quantitatively. However, if the model predicts the gene is not differentially expressed, then its expression likely falls below a noise threshold and that could also be why the dispersion model filtered it out. Thus, it is necessary to choose a dispersion model that accounts for expression level changes across experiments. Here we use a local regression dispersion method to determine if a gene is an outlier or not. Looking across the conditions, the impact of filtering out genes leads to a different set of genes being measured per condition. On average, there are 93 out of 4,267 genes that are different between each experimental run. Of the 93, there is an average of 67 unique genes, or 1.6%, of the transcriptome that are different between the experiments. Namely, these genes will have different responses given the day the experiment was run and thus place an upper bound on the repeatability of the model.

Gene quantitative response (Figure 5C) = The next question to address is if the response levels of the genes (differential expression) are consistent across experimental runs. Response is measured with a quasi-likelihood negative binomial generalized log-linear model which outputs a fold-change as compared to a control. As mentioned in (2), the black and red lines are the genes that were filtered out as outliers by the dispersion model for a single experimental run. Beyond the different set of genes, it is clear that the largest variability in response between the experimental runs are genes that have lower expression in both runs (those closer to the origin). Genes with low expression are inherently unreliable through traditional processing and analysis pipelines (Lamarre *et al.* 2018). These are genes that would not pass a threshold for subsequent analysis in either experimental run and so their responses are statistically insignificant. There are, however, 44 unique genes whose response falls above the noise threshold in both experiments but are outside the 95% confidence bounds of the experiment. Repeatability of a machine learning model with common metrics such as R^2 should not take these genes into account as they fall below a noise threshold.

If a gene is differentially expressed (Figure 5D) = In some applications, such as pathway analysis, the quantitative response of a gene to a perturbation is not as important as whether or not the gene was dysregulated. In such instances, genes are labeled as differentially expressed genes (DEGs) if they satisfy magnitude (often in terms of log fold change) and statistical significance (false discovery rate) thresholds. DEGs are fed to downstream enrichment and pathway analysis tools to identify mechanistic and functional changes. Thus, evaluating whether the set of DEGs remains consistent is also an important aspect of reproducibility. Classification methods that are trained to predict if a gene is differentially expressed should account for both class imbalance as well as the inconsistencies between runs. In our case, we see that the majority of genes across the experimental runs are consistently labeled as impacted (1) or not impacted (0). There are, on average, 140 genes, or 3% of the transcriptome that is labeled differently (labeled as 0-1 or 1-0 in the bar plot). If these genes are essential genes, the set of mechanistic insights that are predicted by the model can be very different based on the experimental run used for validation.

Conclusion

Reproducibility is a cornerstone for utilizing RNASeq data to analyze and improve synthetic designs. One important aspect of reproducible designs is computation pipelines that help standardize files, metadata, analysis techniques, and protocols in a high-throughput and easy to use environment. As such, we have developed and released both a secondary analysis pipeline and *omics_tools* to help facilitate analysis of synthetic biology RNASeq experiments. We have also utilized these two to better understand reproducibility in two different contexts: similarity of controls run on different days and the impact that reproducibility has on the test and training data utilized in machine learning techniques. We identified that the experimental protocols utilized for *E. coli* MG1655 produced highly reproducible results. However, when those same protocols were applied to *B. subtilis*, the reproducibility decreased in a manner that suggested growth states (or some other biological phenomena) were not the same on different days. Given that we standardized the analysis through the analytical pipeline, any differences we observe are guaranteed to be biological or experimental in nature.

Additionally, when exploring induction conditions we found that while the majority of induction conditions were reproducible across the different days/extraction volumes, some were different. The differences have an impact on the ability to validate a model to predict response to inducers if you only have one run to choose from (i.e., you were only able to run the experiment once). While the majority of genes were consistently present and consistently differentially expressed across the two days/volumes, some genes were not. Such differences limit the upper bound of accuracy of the model, as areas where the test data is inconsistent produces a response distribution versus a single response. A standardized analytical pipeline will enable researchers to identify areas of low reproducibility (either experimental or biological), and focus their experiments on reducing the variability of those response distributions. Ultimately, computation pipelines that help facilitate reproducibility will allow for more consistent analysis of the data.

Disclaimer

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Funding

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), the Department of Defense or the United States Government. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory under Contract No. FA8750-17-C-0231 (and related contracts by SD2 Publication Consortium Members).

Material Availability

The secondary analysis pipeline can be found at <https://github.com/SD2E/rna-seq-pipeline> and the configurable differential expression analysis tools and tutorials are available at https://github.com/SD2E/omics_tools.

Data availability

The data underlying this article and all subsequent analyses are publicly available on Figshare: https://figshare.com/projects/RNASeq_Reproducibility_for_Synthetic_Biology/125005

Conflict of Interest: none declared.

Bibliography

- Abbas-Aghababazadeh F, Li Q, Fridley BL. Comparison of normalization approaches for gene expression studies completed with high-throughput sequencing. *PLoS ONE* 2018;**13**:e0206312.
- Bioconductor - Rsubread.
- Brookes E, Stubbs J. Genapp, containers and abaco: technical paper. *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*. New York, NY, USA: ACM, 2019, 1–8.
- Burrows-Wheeler Aligner.
- Chavez M, Ho J, Tan C. Reproducibility of High-Throughput Plate-Reader Experiments in Synthetic Biology. *ACS Synth Biol* 2017;**6**:375–80.
- Cleveland SB, Jamthe A, Padhy S *et al*. Tapis API Development with Python: Best Practices In Scientific REST API Implementation: Experience implementing a distributed Stream API. *Practice and Experience in Advanced Research Computing*. New York, NY, USA: ACM, 2020, 181–7.
- Conesa A, Madrigal P, Tarazona S *et al*. A survey of best practices for RNA-seq data analysis. *Genome Biol* 2016;**17**:13.
- Costa-Silva J, Domingues D, Lopes FM. RNA-Seq differential expression analysis: An extended review and a software tool. *PLoS ONE* 2017;**12**:e0190152.
- Eslami M, Espah Borujeni A, Eramian H *et al*. Prediction of Whole-Cell Transcriptional Response with Machine Learning. *BioRxiv* 2021, DOI: 10.1101/2021.04.30.442142.
- Espah Borujeni A, Zhang J, Doosthosseini H *et al*. Genetic circuit characterization by inferring RNA polymerase movement and ribosome usage. *Nat Commun* 2020;**11**:5001.
- Finotello F, Di Camillo B. Measuring differential gene expression with RNA-seq: challenges and strategies for data analysis. *Brief Funct Genomics* 2015;**14**:130–42.
- Garrido-Rodriguez M, Lopez-Lopez D, Ortuno FM *et al*. A versatile workflow to integrate

- RNA-seq genomic and transcriptomic data into mechanistic models of signaling pathways. *PLoS Comput Biol* 2021;**17**:e1008748.
- Gonçalves RS, Musen MA. The variable quality of metadata about biological samples used in biomedical experiments. *Sci Data* 2019;**6**:190021.
- Gorochowski TE, Espah Borujeni A, Park Y *et al*. Genetic circuit characterization and debugging using RNA-seq. *Mol Syst Biol* 2017;**13**:952.
- Hazen TH, Daugherty SC, Shetty A *et al*. RNA-Seq analysis of isolate- and growth phase-specific differences in the global transcriptomes of enteropathogenic *Escherichia coli* prototype isolates. *Front Microbiol* 2015;**6**:569.
- Intosalmi J, Nousiainen K, Ahlfors H *et al*. Data-driven mechanistic analysis method to reveal dynamically evolving regulatory networks. *Bioinformatics* 2016;**32**:i288–96.
- Jackman SD, Mozgacheva T, Chen S *et al*. ORCA: a comprehensive bioinformatics container environment for education and research. *Bioinformatics* 2019;**35**:4448–50.
- Jessop-Fabre MM, Sonnenschein N. Improving reproducibility in synthetic biology. *Front Bioeng Biotechnol* 2019;**7**:18.
- Kc K, Li R, Cui F *et al*. GNE: a deep learning framework for gene network inference by aggregating biological information. *BMC Syst Biol* 2019;**13**:38.
- Łabaj PP, Kreil DP. Sensitivity, specificity, and reproducibility of RNA-Seq differential expression calls. *Biol Direct* 2016;**11**:66.
- Lamarre S, Frasse P, Zouine M *et al*. Optimization of an RNA-Seq Differential Gene Expression Analysis Depending on Biological Replicate Number and Library Size. *Front Plant Sci* 2018;**9**:108.
- Langfelder P, Horvath S. WGCNA: an R package for weighted correlation network analysis. *BMC Bioinformatics* 2008;**9**:559.
- Lazic SE. *Experimental Design for Laboratory Biologists: Maximising Information and Improving Reproducibility*. Cambridge University Press, 2016.
- Li Q, Brown JB, Huang H *et al*. Measuring reproducibility of high-throughput experiments. *Ann Appl Stat* 2011;**5**:1752–79.
- McLaughlin JA, Myers CJ, Zundel Z *et al*. SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. *ACS Synth Biol* 2018;**7**:682–8.
- Naseri G, Koffas MAG. Application of combinatorial optimization strategies in synthetic biology. *Nat Commun* 2020;**11**:2446.
- Nelson W, Zitnik M, Wang B *et al*. To embed or not: network embedding as a paradigm in computational biology. *Front Genet* 2019;**10**:381.
- Park Y PhD Massachusetts Institute of Technology. Design and debugging of ultrastable engineered genetic systems. 2019.
- Perkel JM. Make code accessible with these cloud services. *Nature* 2019;**575**:247–8.

- Petzold CJ, Chan LJG, Nhan M *et al.* Analytics for metabolic engineering. *Front Bioeng Biotechnol* 2015;**3**:135.
- Picard Tools - By Broad Institute.
- Pinoli P, Ceri S, Martinenghi D *et al.* Metadata management for scientific databases. *Inf Syst* 2018;**81**:1–20.
- Reimand J, Isserlin R, Voisin V *et al.* Pathway enrichment analysis and visualization of omics data using g:Profiler, GSEA, Cytoscape and EnrichmentMap. *Nat Protoc* 2019;**14**:482–517.
- Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 2010;**26**:139–40.
- Sastry AV, Gao Y, Szubin R *et al.* The Escherichia coli transcriptome mostly consists of independently regulated modules. *Nat Commun* 2019;**10**:5536.
- Taylor R, Singhal M. Biological Network Inference and analysis using SEBINI and CABIN. *Methods Mol Biol* 2009;**541**:551–76.
- Tiwari K, Kananathan S, Roberts MG *et al.* Reproducibility in systems biology modelling. *Mol Syst Biol* 2021;**17**:e9982.
- USADELLAB.org - Trimmomatic: A flexible read trimming tool for Illumina NGS data.
- Wilson SL, Way GP, Bittremieux W *et al.* Sharing biological data: why, when, and how. *FEBS Lett* 2021;**595**:847–63.
- Yu L. RNA-Seq Reproducibility Assessment of the Sequencing Quality Control Project. *Cancer Inform* 2020;**19**:1176935120922498.

Reconstruction of Natural Visual Scenes from Primary Visual Cortical Neural Activity Using Adversarial Learning

Anonymous Authors¹

Abstract

We build a model that reconstructs visual scenes shown to a mouse based on the patterns of neural activity that the scenes evoke in the mouse's visual cortex. We detail our optical recordings of several hundred visual cortical neurons at a time as a mouse views grayscale images. Next, we develop loss functions (a pixel loss, a feature loss, and an adversarial loss) that encourage convolutional neural networks to yield accurate and realistic reconstructions of what the mouse was viewing, and we train models with different relative contributions of the loss components. We also present a new framework for the evaluation of reconstructed images. Using this framework we select a preferred model and illustrate its ability to reconstruct images from neuronal data on a held-out test set.

1. Introduction

The mammalian visual system is a wonder of perceptual power and computational efficiency. One goal of computer vision is to replicate its power. Indeed, the first neural networks were inspired by how mammalian neurons work (McCulloch & Pitts, 1943). In this paper, we tackle the reconstruction of natural image scenes from mouse neuronal response with a newly collected data set. First, in Section 2 we review similar attempts to solve related problems, albeit in a different domain. We describe our data collection process in Section 3. In Section 4, we describe our approach to training convolutional neural networks to learn the mapping between neuronal data and natural scenes. We focus on the construction of loss functions that balance accuracy and realism. In Section 5 we present novel metrics for evaluating different reconstructions of natural scenes. We then use the metrics in Section 6 to choose a best model from the ones

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

we train and present the results on a held-out test set.

2. Related Work

Likely due in part to the availability of data sets relating fMRI signals to the images that elicited them (Miyawaki et al., 2008; Van Gerven et al., 2010; Schoenmakers et al., 2013; Horikawa & Kamitani, 2017), there has been substantial work on reconstructing images viewed by humans from fMRI signals — see (Huang et al., 2021) for a survey of techniques. Of particular relevance to our work is image reconstruction from fMRI using deep neural networks. One line of research comprises end-to-end networks that seek to directly predict images from fMRI signals (Lin et al., 2019; Shen et al., 2019a; Beliy et al., 2019). The end-to-end networks use a regularization technique, often an adversarial loss, to ensure the predicted images are realistic or in some meaningful way drawn from the same distribution as the images shown to the subject. Using an adversarial loss is difficult because it requires two networks to reach an equilibrium to improve results opposed to decreasing a traditional loss. The other branch of research (Seeliger et al., 2018; VanRullen & Reddy, 2019; Shen et al., 2019b; Ren et al., 2021) breaks the prediction into two steps. First, they create a mapping from images to a latent space and back to images — for example using a generative adversarial network or autoencoder. Then, they train a simple model to fit the latent space representation of each fMRI signal to that of the image that elicited it. While training a good adversarial network or autoencoder can be difficult there is a wide breadth of literature on the subject and even pre-trained models one can use. The drawback is that a preexisting mapping from latent to image space enforces a strong prior on what the simple model from fMRI to latent space can learn.

Image reconstruction from data collected at the single-neuron level is scarcer, in part because the number of neurons needed for reconstruction of complex images often exceeds the affordances of traditional recording techniques. (Glaser et al., 2020) present an array of traditional machine learning techniques applied to neuronal data from a monkey brain, but not from visual cortex. (Yoshida & Ohki, 2020) collected two-photon calcium imaging data on the simulta-

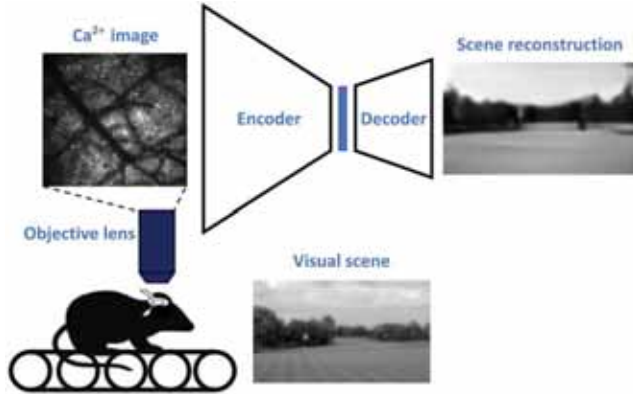


Fig A-11

Figure 1. Representation of data collection from mouse and reconstruction of image

neous activity of several hundred neurons in mouse primary visual cortex. They used Gabor filters to encode images and a linear model to predict the Gabor features associated with each neuron, and were thence able to reconstruct regions of light and dark in the viewed stimuli.

3. Data collection

In this study, we visually present a laboratory mouse with a large selection of natural scene images while recording the neuronal activity in primary visual cortex¹. We use a triple transgenic GCaMP6f-tTA-dCre (Rasgrf2-2A-dCre; Camk2a-tTA; Ai93) mouse developed by the Allen Institute. To prepare the mouse for in vivo imaging sessions, we perform surgeries while the mouse is mounted in a stereotaxic frame under isoflurane anesthesia. We create a cranial window by removing a 5-mm-diameter skull flap over the primary visual cortex and surrounding cortical tissue and cover the exposed cortical surface with a 5-mm-diameter glass coverslip through which we optically record fluorescence activity of layer 2/3 pyramidal neurons. For neuronal imaging, we use a custom-built wide-field fluorescence macroscope with a field of view spanning 2 mm in diameter. We record the individual Ca²⁺ activity of hundreds of pyramidal neurons within our recording window. During a 1-hour recording session, the mouse is head-fixed under the macroscope while freely running on a treadmill. To present grayscale natural images as visual stimuli, the mouse left eye is positioned directly 20cm away from the center of a 32" monitor screen. Using a color filter (Rosco, 382 Congo Blue), we block screen light frequencies around our imaging band.

During the recording session we present the mouse with 3600 distinct natural scene images from the 30 diverse categories in the SUN database (Xiao et al., 2010) — see Ap-

pendix B. Each image fills the entire screen and lasts for 0.5 seconds, then is followed by a 0.5-second gray screen. After recording, small motion artifacts in the neuronal imaging data are computationally removed and trials with high artifacts are discarded (Thevenaz et al., 1998). Border areas are cropped, and the neuronal imaging data are down-sampled to 208 by 208 pixels. We also down-sample the data in time to 5 Hz and save the resulting 4 frames that reliably lie within the visual stimulation period — 200 ms, 400 ms, 600 ms, and 800 ms after stimulus onset.

As an additional preprocessing step, we take the mean of the 4 frames for each trial and calculate the baseline drift with a 3rd order low pass Butterworth filter applied to each mean pixel’s time course, with a cutoff frequency of 20 trials per cycle. We subtract the resulting baseline drift from each frame in the trial data.

We divide the data into 2806 training trials, 200 validation trials, and 200 test trials. We ensure that all data in the testing set are at least 10 trials away from the training and validation sets by discarding the intervening time trials, yielding a total of 3206 trials. Each trial comprises 4 frames of 208 by 208 pixels of cleaned neuronal data, along with the grayscale image that was presented to elicit those neuronal responses. Our modeling goal is to predict the grayscale image from the neuronal data.

4. Model

Inspired by approaches on other image-to-image translation problems (Isola et al., 2017; Shen et al., 2019a), we use an encoder-decoder architecture to reconstruct the presented image based on brain data (see Figure 1). The architectures for the encoder and decoder are detailed in Appendix A.1 and Appendix A.2 respectively and are based on the models from (Isola et al., 2017). In this section, we work up to a final model that uses a combination of three losses to generate realistic images (Dosovitskiy & Brox, 2016): A pixel loss, $\mathcal{L}_{img}$, to guide predictions to be close to the true image; and an adversarial loss, $\mathcal{L}_{adv}$, to encourage the predicted images to be “realistic”, a feature loss (also called perception loss), $\mathcal{L}_{feat}$, that encourages predictions to contain similar objects to those in the stimuli without strictly enforcing object location.

First, let $\mathbf{x}_i$ denote the neuronal data from the i^{th} trial, in which the image $\mathbf{y}_i$ was shown to the mouse. Denote the model as G_θ , so the reconstructed image is $G_\theta(\mathbf{x}_i)$.

The most natural place to start is with a loss that encourages each pixel of the reconstruction to match the corresponding pixel of the original image. We simply use mean squared error,

$$\mathcal{L}_{img} = \sum_i (\mathbf{y}_i - G_\theta(\mathbf{x}_i))^2.$$

¹The Stanford University Administrative Panel on Laboratory Animal Care approved all procedures using animals.



Figure 2. Reconstructed images with models using various loss terms. From left to right: original image, reconstruction with pixel loss, reconstruction with pixel and adversarial loss, reconstruction with pixel, feature and adversarial loss.

Fig A-12

Figures 2 and 12, second column, shows reconstructions using only this pixel loss. As noted in (Dosovitskiy & Brox, 2016), — and (Isola et al., 2017) using the mean absolute error — the reconstructions capture the coarse structure of the images but fail to capture fine details.

In order to increase the realism and level of detail of the reconstructed images, we introduce an adversarial loss. We train a discriminator network — detailed in Appendix A.4 — D_ϕ , to predict whether 34×34 pixel patches come from actual images or from reconstructed images. Let $y_{i,j}$ denote the j^{th} patch in the i^{th} image and $G_\theta(\mathbf{x}_i)_j$ denote the j^{th} patch in the i^{th} reconstruction. We train D_ϕ with training images and reconstructions from G_θ using the loss function

$$\mathcal{L}_{discr} = - \sum_{i,j} \log(D_\phi(\mathbf{y}_{i,j})) + \log(1 - D_\phi(G_\theta(\mathbf{x}_i)_j)).$$

We define the adversarial loss using the discriminator net-

work:

$$\mathcal{L}_{adv} = - \sum_{i,j} \log(D_\phi(G_\theta(\mathbf{x}_i)_j)).$$

We update D_ϕ and G_θ concurrently in our training process. The adversarial loss encourages the reconstruction network G_θ to output images with patches that could have come from real images. In the third column of Figures 2 and 12, we show reconstructions from a model trained with pixel and adversarial losses. The reconstructions are indeed sharper and more realistic.

As observed out by (Dosovitskiy & Brox, 2016; Shen et al., 2019a), we can further improve our results with a feature loss. Separately, we train a network using contrastive learning (Chen et al., 2020a), using images in the training and validation set as well as additional images. We use a convolution neural network to generate a 512-dimensional embedding and then a multi-layer perception head in the contrastive learning task — see Appendix A.3 for architecture details and Appendix C for the details of the dataset we used. Let $C(\mathbf{y}_i)$ denote the vector embedding of $\mathbf{y}_i$ by the network. Then the loss is

$$\mathcal{L}_{feat} = \sum_i (C(\mathbf{y}_i) - C(G_\theta(\mathbf{x}_i)))^2.$$

The loss encourages the reconstructions of the model to have similar visual features without strictly enforcing location. The final column of Figures 2 and 12 shows reconstructions of a model we train with all three losses.

The complete loss that we use to train our final model is

$$\mathcal{L}_{total} = \lambda_{img} \mathcal{L}_{img} + \lambda_{feat} \mathcal{L}_{feat} + \lambda_{adv} \mathcal{L}_{adv}. \quad (1)$$

This combined loss works to overcome the low fidelity of the mouse visual system, the lack of eye-tracking, and the tendency of naïve pixel-based models to yield very blurry predictions. We train several versions of our model with

$$\begin{aligned} \lambda_{adv} &= 1, \\ \lambda_{img} &= [25, 50, 100, 200, 400], \\ \lambda_{feat} &= [200, 400, 800, 1600, 3200] \end{aligned} \quad (2)$$

in Equation (1). Models with an adversarial loss do not have a conventional early stopping rule (Heusel et al., 2017). As such, we store the results at 75 epochs and 100 epochs for each parameter combination. We find in our experimentation that running the model for more epochs does not necessarily yield better results. We then have a total of 50 versions of the model to pick from. In the next section, we develop an evaluation criterion for doing so.

5. Evaluation

The evaluation of models using an adversarial loss is an open problem; see (Borji, 2019; 2021) for a survey of methods for evaluating generative models. The literature on

translating from fMRI to images presents two methods for the evaluation of images, both with significant drawbacks. First, (Shen et al., 2019a; Lin et al., 2019; Ren et al., 2021) use pixel-wise metrics — for example, Pearson correlation coefficient, structural similarity index measure, or root mean squared error. These pixel-wise metrics are sensitive to edge misalignment, favor blurrier versions of the reconstructed images, and are sensitive to artifacts imperceptible to humans. Furthermore, (Zhang et al., 2018) shows that such pixel-wise metrics do not correspond to human perception. An alternative approach, used by (Seeliger et al., 2018; Belyi et al., 2019; Shen et al., 2019b; VanRullen & Reddy, 2019; Ren et al., 2021), involves conducting a human perceptual study to evaluate several different models. However, human evaluation is time-consuming and expensive, and thus only practical when evaluating relatively few models and images.

We propose an adaptation of the improved precision and recall metric proposed by (Kynkäänniemi et al., 2019) for evaluating unconditional generative models, or models that seek to produce realistic images without matching a particular image. To define this metric, we first define an auxiliary function. Let $\mathcal{D}$ denote a set of vectors. Let $\text{NN}_k(\mathbf{z}, \mathcal{D})$ be the k^{th} nearest neighbor of $\mathbf{z}$ in $\mathcal{D}$. Define the following function:

$$f(\mathbf{q}, \mathbf{z}; \mathcal{D}, k) = \begin{cases} 1 & \|\mathbf{q} - \mathbf{z}\| \leq \|\mathbf{z} - \text{NN}_k(\mathbf{z}, \mathcal{D})\| \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

This function asks if $\mathbf{q}$ is closer to $\mathbf{z}$ than $\mathbf{z}$'s k^{th} nearest neighbor in $\mathcal{D}$.

Now, define precision at k as

$$\text{precision}_k(G_\theta; \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, \mathcal{D}_t, C) = \frac{1}{N} \sum_{i=1}^N f(C(G_\theta(\mathbf{x}_i)), C(\mathbf{y}_i); C(\mathcal{D}_t), k) \quad (4)$$

where C is a neural network that maps an image to an embedding and $\mathcal{D}_t$ is the training set of images. In this definition, f asks if the reconstructed image is closer to the real image than a threshold based on how many similar examples to the real image exist in the training set. Precision $_k$ thus quantifies how accurate the model's reconstruction is. Next, define recall at k :

$$\text{recall}_k(G_\theta; \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, \mathcal{D}_t, C) = \frac{1}{N} \sum_{i=1}^N f(C(\mathbf{y}_i), C(G_\theta(\mathbf{x}_i)); C(\mathcal{D}_t), k). \quad (5)$$

In this definition, f evaluates whether the generated image is closer to the target than to other images in the training set. Recall $_k$ quantifies generalization and diversity of the model—is the model producing new images that are close

to the target or is it producing near copies of images in the training set?

To get a sense of precision $_k$ and recall $_k$, imagine we had a naïve model that for any input would output a random image from the training set; the precision $_k$ would be near $\frac{k}{|\mathcal{D}_t|}$. If the distributions of the testing and training sets are the same, the recall $_k$ would also be close to this quantity.

Precision $_k$ and recall $_k$ depend on two hyperparameters: the embedding model C and the number of nearest neighbors k .

The choice of embedding model matters a lot. The network C used to calculate precision $_k$ and recall $_k$ should be different than the network used in the feature loss to train the reconstruction models. Weighting the feature loss too heavily in the training process may result in artifacts that are ignored when using the same embedding model to calculate precision $_k$ and recall $_k$ (or any metric in feature space). To illustrate this point, we train our reconstruction models without the adversarial loss. Figure 3 compares the precision $_k$ and recall $_k$ of reconstruction models without the adversarial loss to the reconstruction models with adversarial loss using different embeddings by three different networks: our embedding model used in the feature loss, the simplest simCRLV2 model trained in (Chen et al., 2020b) using contrastive learning, and the ResNet50 model trained on ImageNet classification accessible through TensorFlow, referred to hereafter as ImageNet ResNet50. Using our embedding model we would conclude that the best reconstruction model (closest to the top right corner) is one that does not use adversarial loss. However, as shown in Figure 4, this “best” model has obvious artifacts that are ignored by our embedding model when calculating precision $_k$ and recall $_k$. On the other hand, the simCRLV2 and ImageNet ResNet50 embedding models rightly give the models trained without adversarial loss much worse precision $_k$ and recall $_k$ than the models with the adversarial loss.

From Figure 3 it is clear that, despite using the same k , the range of precision $_k$ varies substantially between the embedding models. Thus, one should choose the embedding model first and then choose the appropriate k . Another consideration is that different embedding models give different rank orderings. For example, the Spearman correlation of the precision $_{280}$ of the 50 models between the simCRLV2 and ImageNet ResNet50 embedding models is 0.727 and the Spearman correlation for the recall $_{280}$ is 0.750.

For this paper, we will use the simCRLV2 model to calculate precision $_k$ and recall $_k$ for four reasons. First, as mentioned above, we want to choose a different model from the embedding model used in the feature loss so that it does not ignore artifacts. Intuitively, a network with a different architecture, trained on a different data set, should penalize the artifacts.

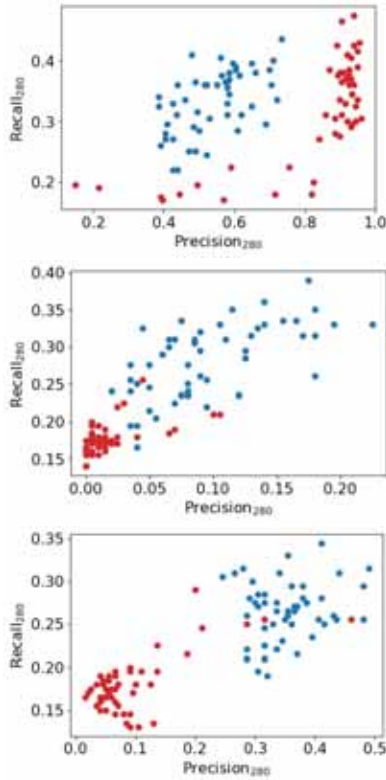


Figure 3. The precision_k and recall_k of the different models at $k = 280$ for our embedding model (top), simCRLV2 (middle), and ImageNet ResNet50 weights (bottom). The models with adversarial loss are in blue and the models without adversarial loss are in red.

Fig A-13

Second, simCRLV2 is readily available and was trained using considerable computational resources. Third, as (Cao & Wu, 2021) show, fine-grain details are less important to networks trained with contrastive learning, like simCRLV2, than to networks trained with classification, like ImageNet ResNet50. The same should hold for the resulting feature vector. Given poor mouse vision and no eye-tracking data, we prefer such a feature vector that weights the coarse details more than the fine. Forth, as we show quantitatively in Appendix D, the embedding by simCRLV2 captures more details of our data set than ImageNet ResNet50 does.

The second hyperparameter for precision_k and recall_k is k itself, which determines the strictness of our threshold. In Figure 5, we show precision_k and recall_k for different values of k for 5 different models (the five with the highest average precision₉₈₀ and recall₄₉₀). We recommend choosing different k for precision_k and recall_k so that top models are have scores around 0.5. As such, we choose $k = 980$ for precision_k and $k = 490$ for recall_k. Further work may reveal better ways to choose k for precision_k and recall_k. In the next section, we use these metrics to pick a recon-



Figure 4. The original image (left) and a reconstruction of the model with the highest average of precision₂₈₀ and recall₂₈₀ using our embedding model (right). The reconstruction has obvious artifacts.

Fig A-14

struction model and present the model’s reconstruction of stimulus images from neuronal data.

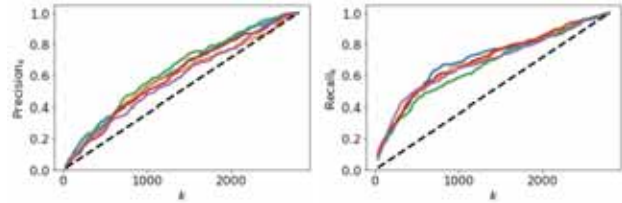


Figure 5. The precision_k and recall_k of five models for different values of k . We choose the five models with the highest average precision₉₈₀ and recall₄₉₀. The dashed black line is how our theoretical baseline model would perform.

Fig A-15

6. Results

6.1. Preferred model

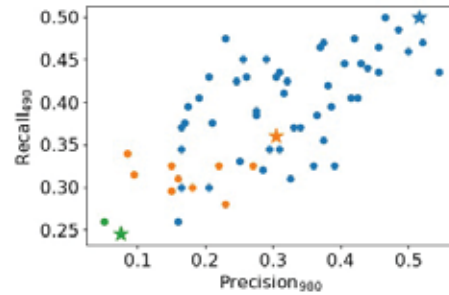


Fig A-16

Figure 6. The precision₉₈₀ and recall₄₉₀ of the different models. The points in green are the models trained with only the pixel loss. Models in orange are models that use the pixel and adversarial loss. The points in blue are the models that use the complete loss. The model from each category with the highest average precision₉₈₀ and recall₄₉₀ is indicated with a star.

We show in Figure 6 the precision₉₈₀ and recall₄₉₀ for our 50 candidate models and, for comparison, models using only the pixel loss and models using the pixel and adversarial loss. Using a combination of losses results in higher precision₉₈₀ and recall₄₉₀ scores. We choose as our preferred model the candidate with the highest average of precision₉₈₀ and recall₄₉₀: $\lambda_{img} = 100$, $\lambda_{feat} = 3200$, trained for 75 epochs

(indicated in Figure 6 by a blue star). The model has a precision₉₈₀ of 0.515 and a recall₄₉₀ of 0.500.

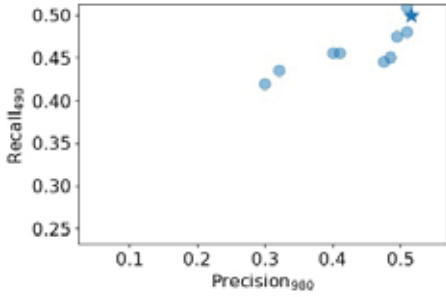


Fig A-17

Figure 7. The precision₉₀₈ and recall₄₉₀ of are preferred model, indicated by a star, and nine models with the same parameters as the preferred model.

As a consistency test, we train from scratch 9 additional models with the same hyperparameters as our preferred model and plot each of them in Figure 7. There is variation in the performance of these models, especially in terms of precision. This indicates that random elements — e.g. the initialization of the model or the order with which data are presented during optimization — may be as important to the performance of the final model as the exact parameters chosen.

Now we look at the preferred model’s results on the test set. The preferred model has the 4th highest score on the test set with precision₉₈₀ of 0.475 and a recall₄₉₀ of 0.395, while the highest performing model on the test set — which scores 5th on the validation set — has a precision₉₈₀ of 0.530 and a recall₄₉₀ of 0.400 on the test set. The Spearman correlation between the scores of the 50 models on the validation and test set is 0.893. Thus, the validation set does a great job of predicting performance on the test set.

Next, we will look at how well the preferred model reconstructs the individual images.

6.2. Performance of preferred model on individual test images

To rate individual images, we adapt realism score (RS_k) from (Kynkäänniemi et al., 2019):

$$RS_k(G_\theta(\mathbf{x}), \mathbf{y}; \mathcal{D}_t, C) = \frac{\|C(\mathbf{y}) - \text{NN}_k(C(\mathbf{y}), C(\mathcal{D}_t))\|}{\|C(G_\theta(\mathbf{x})) - C(\mathbf{y})\|}.$$

The realism score can be thought of as the continuous version of the function f from Equation (3). We choose $k = 980$ to match the k from the precision_k in evaluation. In Figure 8, we give the distribution of RS_{980} for images in the test set. The scores loosely follow a normal distribution.

In Figure 9, we present the reconstructed images with the

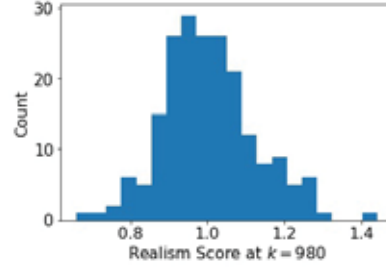


Figure 8. Histogram of realism scores (for $k = 980$) of the images in the testing set.

Fig A-18

highest realism score. Figure 10 presents the reconstructions with the lowest realism score and Figure 11 shows a random sampling of middling reconstructions. We show more reconstructions in Figures 13 and 14 in the appendix. In general, regardless of realism score, the model does a good job of capturing the overall distribution of light and dark areas, presumably coming largely from the mouse’s brain. The major difference between a good realism score and a bad one is the finer details of what fills those areas, which are in part encoded in the model thanks to the adversarial loss. In some cases, the locations of image elements are shifted relative to the stimulus images, perhaps because the mouse was looking in an unusual direction in the trial in question. In the future, muddling of finer details could be resolved by controlling for eye position, a larger or more diverse data set, and/or training the adversarial loss with a richer stimulus set; it may also reflect the trade-offs between decoding information present in the brain and making guesses based on natural image statistics.

7. Discussion and Conclusion

Despite some limitations in the data, we are able to build a model that reconstructs what the mouse was looking at from its neuronal activity surprisingly well. We took the end-to-end network approach, building a network to directly predict the images that the mouse is seeing. As mentioned in Section 2, many fMRI approaches instead build a network that maps a latent space to natural images and then predicts the latent space representation from the fMRI data. It remains to be determined whether this approach might be successful at reconstructing stimuli from neuronal data.

Both predicting the latent representation and the end-to-end approach use an adversarial loss, albeit at different points in the training procedure. The adversarial loss imposes a prior — via the images used to train the discriminator function — on the images the model reconstructs. The modeler chooses a domain from which to draw the image statistics for the prior of the model. In our case, this prior and the training data both span a larger set of categories than the testing data.

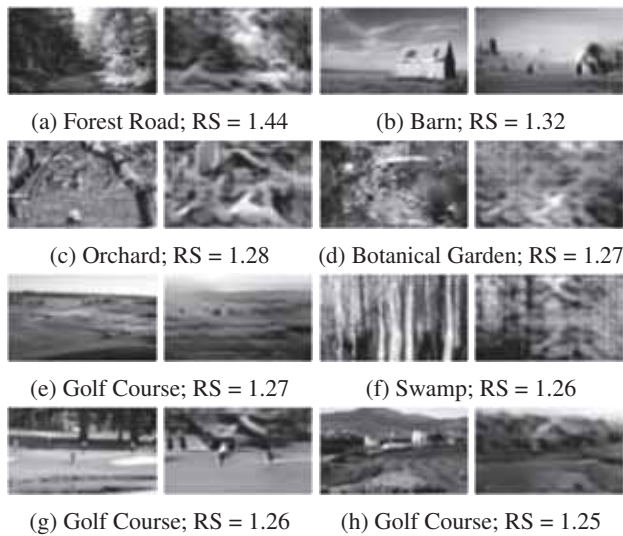


Fig A-19

Figure 9. Reconstructed images with the highest realism scores. The original image is on the left and the reconstructed image is on the right.

With a richer and more varied data set, one could draw a testing set from a significantly different distribution from the training data and prior to see how well a model can reconstruct novel kinds of images.

Evaluation is another large issue facing both generative and reconstruction tasks. We apply precision_k and recall_k , previously developed to evaluate unconditional image generation (Kynkäänniemi et al., 2019), in a novel approach to reconstruction evaluation that is computationally efficient while providing reasonable agreement with human perception. Another option — if one is less concerned with a diversity/generalization metric — is to take the average reconstruction error in embedding space. In developing our metric, we experimented with average reconstruction error in embedding space and found that the results were visually far inferior to those presented here.

In our metrics, we would have preferred a more objective way to choose k . While we provide qualitative and quantitative reasons for choosing the embedding model, we can only give a suggestion for the choice k . We hope others can build upon our first step in applying generative model evaluation techniques to the reconstruction problem.

With more and richer data, further refinement of models, and improved evaluation techniques, image reconstruction from neuronal data could be even better. A good grasp on the reconstruction of static images may also lay a foundation for the more difficult problem of decoding dynamic stimuli such as videos. Collecting data in response to video at a higher temporal frequency will allow us to probe the brain’s mechanisms of encoding dynamic stimuli. Such knowledge

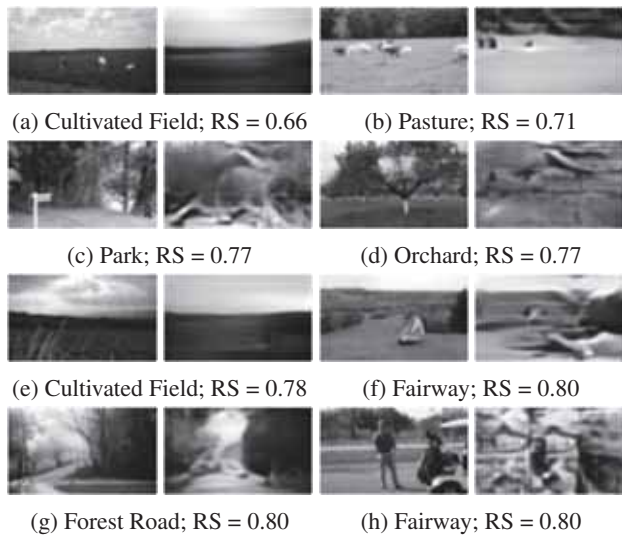


Fig A-20

Figure 10. Reconstructed images with the lowest realism scores. The original image is on the left and the reconstructed image is on the right.

may lay the groundwork for future brain-machine interfaces.

References

- Beliy, R., Gaziv, G., Hoogi, A., Strappini, F., Golan, T., and Irani, M. From voxels to pixels and back: Self-supervision in natural-image reconstruction from fMRI. *arXiv preprint arXiv:1907.02431*, 2019.
- Borji, A. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.
- Borji, A. Pros and cons of GAN evaluation measures: New developments. *arXiv preprint arXiv:2103.09396*, 2021.
- Cao, Y.-H. and Wu, J. Rethinking self-supervised learning: Small is beautiful. *arXiv preprint arXiv:2103.13559*, 2021.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.
- Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b.
- Dosovitskiy, A. and Brox, T. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, 29:658–666, 2016.

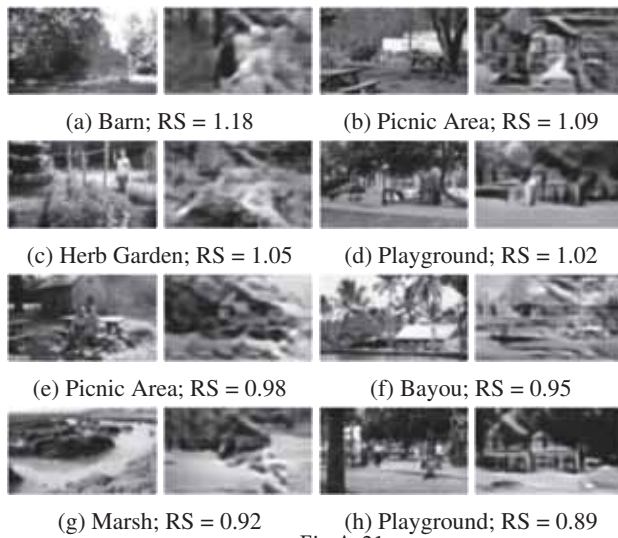


Fig A-21

Figure 11. Reconstructed images with middling realism scores. The original image is on the left and the reconstructed image is on the right.

Glaser, J. I., Benjamin, A. S., Chowdhury, R. H., Perich, M. G., Miller, L. E., and Kording, K. P. Machine learning for neural decoding. *Eneuro*, 7(4), 2020.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Horikawa, T. and Kamitani, Y. Generic decoding of seen and imagined objects using hierarchical visual features. *Nature communications*, 8(1):1–15, 2017.

Huang, S., Shao, W., Wang, M.-L., and Zhang, D.-Q. fMRI-based decoding of visual information from human brain activity: A brief review. *International Journal of Automation and Computing*, pp. 1–15, 2021.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. Improved precision and recall metric for assessing generative models. *arXiv preprint arXiv:1904.06991*, 2019.

Lin, Y., Li, J., and Wang, H. DCNN-GAN: Reconstructing realistic image from fmri. In *2019 16th International Conference on Machine Vision Applications (MVA)*, pp. 1–6. IEEE, 2019.

McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Miyawaki, Y., Uchida, H., Yamashita, O., Sato, M.-a., Morito, Y., Tanabe, H. C., Sadato, N., and Kamitani, Y. Visual image reconstruction from human brain activity using a combination of multiscale local image decoders. *Neuron*, 60(5):915–929, 2008.

Ren, Z., Li, J., Xue, X., Li, X., Yang, F., Jiao, Z., and Gao, X. Reconstructing seen image from brain activity by visually-guided cognitive representation and adversarial learning. *NeuroImage*, 228:117602, 2021.

Schoenmakers, S., Barth, M., Heskes, T., and Van Gerven, M. Linear reconstruction of perceived images from human brain activity. *NeuroImage*, 83:951–961, 2013.

Seeliger, K., Güçlü, U., Ambrogioni, L., Güçlütürk, Y., and van Gerven, M. A. Generative adversarial networks for reconstructing natural images from brain activity. *NeuroImage*, 181:775–785, 2018.

Shen, G., Dwivedi, K., Majima, K., Horikawa, T., and Kamitani, Y. End-to-end deep image reconstruction from human brain activity. *Frontiers in Computational Neuroscience*, 13:21, 2019a.

Shen, G., Horikawa, T., Majima, K., and Kamitani, Y. Deep image reconstruction from human brain activity. *PLoS computational biology*, 15(1):e1006633, 2019b.

Thevenaz, P., Ruttimann, U. E., and Unser, M. A pyramid approach to subpixel registration based on intensity. *IEEE transactions on image processing*, 7(1):27–41, 1998.

Van Gerven, M. A., De Lange, F. P., and Heskes, T. Neural decoding with hierarchical generative models. *Neural computation*, 22(12):3127–3142, 2010.

VanRullen, R. and Reddy, L. Reconstructing faces from fMRI patterns using deep generative neural networks. *Communications biology*, 2(1):1–10, 2019.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Yoshida, T. and Ohki, K. Natural images are reliably represented by sparse and variable populations of neurons in visual cortex. *Nature communications*, 11(1):1–19, 2020.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference*

on computer vision and pattern recognition, pp. 586–595, 2018.

A. Network architectures

In this section of the appendix we lay out our network architectures in detail.

A.1. The encoder architecture

Brain Image (208,208,4)
64 filters, 4 Kernel Size, 2 Stride (104,104,64)
128 filters, 4 Kernel Size, 2 Stride (104,104,128)
256 filters, 4 Kernel Size, 2 Stride (52,52,256)
512 filters, 4 Kernel Size, 2 Stride (26,26,512)
512 filters, 4 Kernel Size, 2 Stride (13,13,512)
512 filters, 4 Kernel Size, 2 Stride (7,7,512)
512 filters, 4 Kernel Size, 2 Stride (4,4,512)
512 filters, 4 Kernel Size, 2 Stride (2,2,512)
512 filters, 2 Kernel Size, 2 Stride (1,1,512)
Output (1,1,512)

Each layer is a convolution followed by batch normalization and a leaky relu ($\alpha = 0.2$) activation. The convolutions use “same” padding, no bias, and are initialized with normal random numbers with mean 0 and standard deviation 0.02.

A.2. The decoder architecture

Input (1,1,512)
512 filters, 2 by 3 Kernel Size, 2 by 3 Stride (2,3,512)
512 filters, 4 Kernel Size, 2 Stride (4,6,512)
512 filters, 4 Kernel Size, 2 Stride (8,12,512)
256 filters, 4 Kernel Size, 2 Stride (16,24,256)
128 filters, 4 Kernel Size, 2 Stride (32,48,128)
64 filters, 4 Kernel Size, 2 Stride (64,96,64)
Crop (56,96,64)
32 filters, 4 Kernel Size, 1 Stride (56,96,32)
1 filters, 4 Kernel Size, 1 Stride (56,96,1)
Generated Image (56,96,1)

The first layer is a convolution followed by 50% dropout with ReLU activation. Every other layer is a convolution followed by batch normalization, 50% dropout and a ReLU activation. The final layer is a convolution followed by tanh activation. The convolutions use “same” padding, no bias, and are initialized with normal random numbers with mean 0 and standard deviation 0.02.

A.3. The architecture for the embedding model in the feature loss

Image (56 height, 96 width, 1)
64 filters, 4 Kernel Size, 2 Stride (28,48,64)
128 filters, 4 Kernel Size, 2 Stride (14,24,128)
256 filters, 4 Kernel Size, 2 Stride (7,12,256)
512 filters, 4 Kernel Size, 2 Stride (4,6,512)
512 filters, 4 Kernel Size, 2 Stride (2,3,512)
512 filters, 2,3 Kernel Size, 2,3 Stride (1,1,512)
Flatten (512)
Dense 512 “Feature Vector”
Dense 512
Dense 64

The convolution layers are a convolution followed by batch normalization then a leaky ReLU ($\alpha = 0.2$) activation. All dense layers use ReLU activation. After training the network using contrastive learning, we use the output of the layer labeled “Feature Vector” as the embedding in the feature loss.

A.4. The Patch Discriminator

Image (56 height, 96 width, 1)
64 filters, 4 Kernel Size, 2 Stride (28,48,64)
128 filters, 4 Kernel Size, 2 Stride (14,24,128)
256 filters, 4 Kernel Size, 1 Stride (13,23,256)
1 filters, 4 Kernel Size, 1 Stride (12,22,1)
Matrix of probabilities 34 by 34 patch size (12,22,1)

The first layer is a convolution followed by 50% dropout then leaky ReLU ($\alpha = 0.2$) activation. The second layer is a convolution followed by batch normalization, 50% dropout then a leaky ReLU ($\alpha = 0.2$) activation. The third layer is a convolution followed by batch normalization then a leaky ReLU ($\alpha = 0.2$) activation. The final layer is a convolution followed by a sigmoid activation. The first two convolutions use “same” padding while the final two are zero padded with width one. The first three convolutional layers have no bias, but the final convolutional layer has a bias term. We initialized each convolution with normal random numbers with mean 0 and standard deviation 0.02. This architecture yields a 34 by 34 pixel patch size.

B. Image categories shown to mouse

We show the mouse images in the SUN database from the following categories: bamboo forest, barn, bayou, boardwalk, botanical garden, broadleaf forest, cultivated field, fairway, forest path, forest road, formal garden, golf course, hayfield, herb garden, marsh, needleleaf forest, orchard, out-house, park, pasture, picnic area, playground, putting green, rainforest, rice paddy, swamp, topiary garden, tree farm, tree house, and wild field.

Table 1. Loss, top 1 and top 3 accuracy of three embedding models on classification.

Embedding Model	Loss	Top 1 Acc	Top 3 Acc
Our Model	1.316	0.583	0.847
simCRLV2	1.427	0.573	0.807
ImageNet ResNet50	1.653	0.515	0.743

C. Image categories used in constrastive learning

For constrastive learning, we use 9604 images in the SUN database from the following categories: athletic field, bamboo forest, barn, bayou, boardwalk, botanical garden, broadleaf forest, butte, canyon, cemetery, chicken coop, cliff, corn field, corral, cottage garden, creek, cultivated field, desert vegetation, driving range, fairway, forest path, forest road, formal garden, golf course, hayfield, herb garden, marsh, mountain, needleleaf forest, orchard, outhouse, park, pasture, pavilion, picnic area, playground, pond, putting green, rainforest, rice paddy, river, rock arch, rope bridge, sandbar, sea cliff, stable, swamp, topiary garden, tree farm, tree house, vegetable garden, village, vineyard, wheat field, wild field, wind farm, and yard.

D. Quantitative evaluation of embedding models

Since the images we show the mouse have labeled classes, we can test the accessibility of the information represented by the different embedding models with a simple classification task over those embeddings. For each embedding, we train a multinomial logistical regression model. The intuition is that a better embedding should be better able to drive class discrimination by this simple model.

From the SUN database we took the 30 classes with the most images: botanical garden, butte, canyon, cemetery, chicken coop, cliff, corn field, cottage garden, creek, desert vegetation, fairway, forest path, formal garden, golf course, herb garden, mountain, orchard, outhouse, pasture, pavilion, playground, pond, river, sandbar, stable, vegetable garden, village, vineyard, wheat field, and wind farm. Because evaluating this classification model contributed to our selection of an embedding model, we exclude from consideration images in the image reconstruction test set. We make a testing set for this classification task with 20 images from each class, a validation set with 10 images from each class, and use the rest of the images as training data. We implement multinomial logistical regression with a single dense layer and use the validation set for early stopping. In Table I, we present the results of the classification task on the three embedding models mentioned in the paper. We run each three times and report the best results, though the loss of

each run was always within 0.5%. The main takeaway is that the embedding from simCRLV2 does decisively better than ImageNet ResNet50. This is an additional reason to use simCRLV2 over ImageNet ResNet50 in our calculation of precision_k and recall_k .

In addition, we also use a classification task to choose which embedding model to use for the feature loss. We do not report the results here, but we use a simpler classification task — 11 classes and a smaller training set — to choose between several self-supervised techniques and different network architectures. Contrastive learning with the architecture detailed in Appendix A.3 performs the best.

E. Additional Reconstructions

In this section, we show more reconstructions. In Figure 12 we display reconstructions of models as we add additional loss terms. In Figures 13 and 14, we display additional reconstructions from our preferred model along with the realism scores.



Figure 12. More reconstructed images with models using various loss terms. From left to right: original image, reconstruction with pixel loss, reconstruction with pixel and adversarial loss, reconstruction with pixel, feature and adversarial loss.

Fig A-22

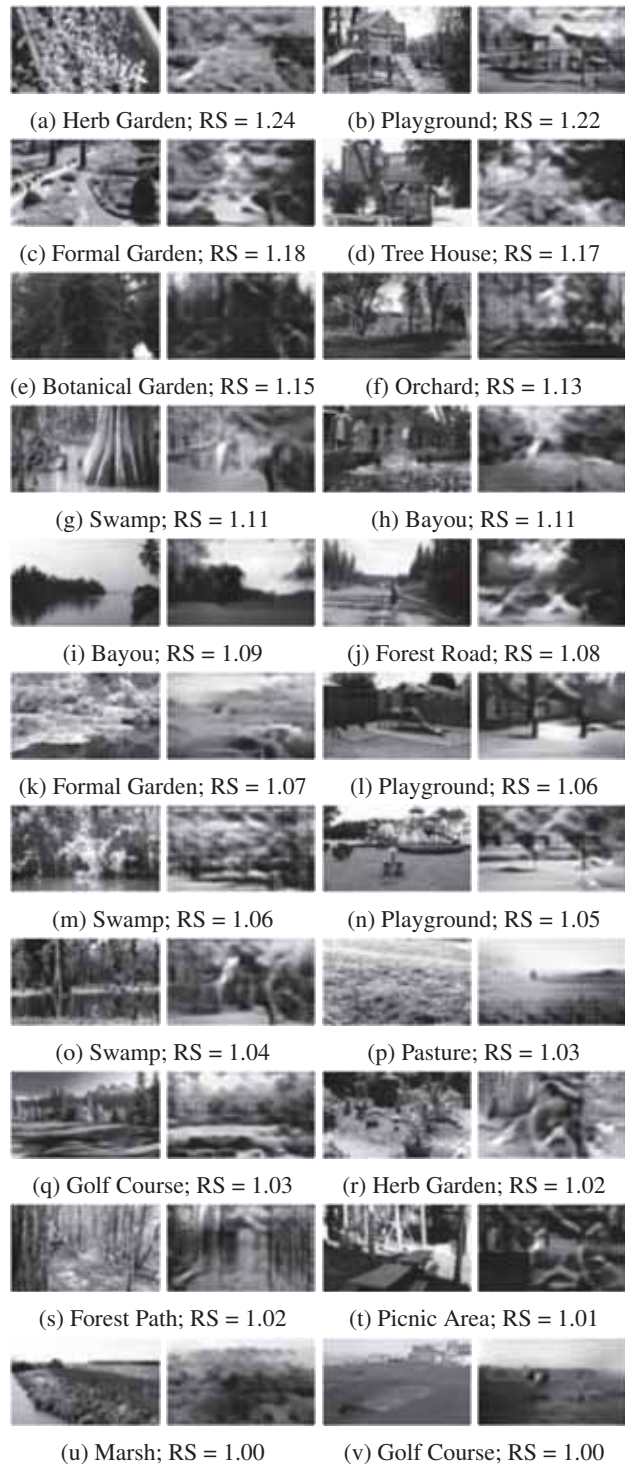


Figure 13. More reconstructed image from our preferred model. The original image is on the left and the reconstructed image is on the right.

Fig A-23

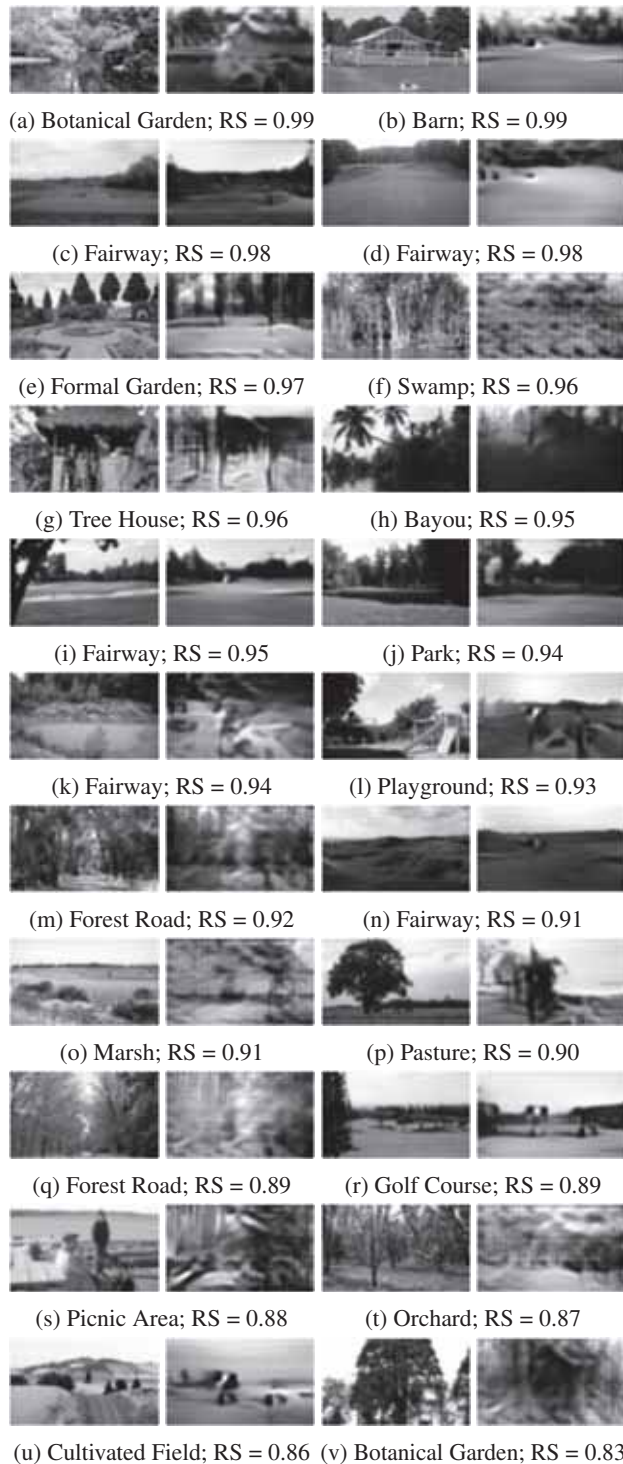


Figure 14. Even more reconstructed image from our preferred model. The original image is on the left and the reconstructed image is on the right.

Fig A-24



Subject Section

Comparison and evaluation of data-driven protein stability prediction models

Jennifer A. Csicsery-Ronay^{1*}, Alexander Zaitzeff¹, Jedediah M. Singer¹

¹ Two Six Technologies, Arlington, Virginia, United States of America

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Predicting protein stability is important to protein engineering yet poses unsolved challenges. Computational costs associated with physics-based models, and the limited amount of data available to support data-driven models, have left stability prediction behind the prediction of structure. New data and advancements in modeling approaches now afford greater opportunities to solve this challenge. We evaluate a set of data-driven prediction models using a large, newly published dataset of various synthetic proteins and their experimental stability data. We test the models in two separate tasks, exercising extrapolation to new protein classes and prediction of the effects on stability of small mutations.

Results: Small convolutional neural networks trained from scratch on stability data and large protein embedding models passed through simple downstream models trained on stability data are both able to predict stability comparably well. The largest of the embedding models yields the best performance in all tasks and metrics. A blended ensemble yields performance exceeding that of its constituent models, suggesting that they are encoding complementary information.

Contact: jennifer.csicsery-ronay@twosixtech.com

Supplementary information: The code, data, and results are available at *Bioinformatics* online.

1 Introduction

A protein's stability—the likelihood it assumes its folded configuration—is an essential property that interacts with its structure, function, solubility, and durability Nisthal *et al.* (2019). Additions, deletions, and alterations of single amino acids can change not only the folded structure but also its stability Razban and Shakhnovich (2020). Predicting a change in a protein's stability under the effect of a mutation can guide engineering in medically significant domains, such as diagnoses, therapeutics, and vaccine development Burgess (2015); Fang (2020). However, stability prediction remains challenging.

Physics-based software like Rosetta drives computational protein design and is responsible for many stable synthetic proteins Rocklin *et al.* (2017); Leman *et al.* (2020). Despite recent progress, however, statistical and physics-based methods for stability prediction remain computationally expensive and struggle to model and predict the quantities and interactions governing protein stability Magliery (2015); Kulshreshtha *et al.* (2016). Meanwhile, advancements in data-driven methods such as high throughput stability measurement, deep sequencing, and machine

learning are beginning to mitigate issues of computational cost and prediction accuracy Fang (2020); Wang *et al.* (2021); Alley *et al.* (2019); Rao *et al.* (2019); Singer *et al.* (2021).

Here we present a rigorous comparison and evaluation of several data-driven stability prediction models using 112,455 synthetic proteins from a recently published dataset Singer *et al.* (2021) of experimental stability data. We evaluate performance on held-out protein classes to test models' ability to extrapolate new kinds of protein designs. We also further the work of Rao *et al.* (2019) and evaluate models' predictions on single-site mutations to test their ability to predict small changes in the stability landscape. Models tested include a convolutional neural network (CNN) model from Singer *et al.* (2021) and three simplified variants of this model, as well as gradient boosting models built using three recently-published protein embeddings inspired by natural language processing: (i) the Evolutionary Scale Model (ESM, Rives *et al.* (2020)), (ii) ProtBERT Elnaggar *et al.* (2021), and (iii) UniRep Alley *et al.* (2019). We use XGBoost Chen and Guestrin (2016) to construct these gradient boosting models, and test both default and optimized parameters. Finally, we use a linear blending technique to assemble ensemble models, and evaluate their performance against their constituents.

2 Methods

2.1 Data

Models were trained on data for 112,455 proteins, reported in Singer *et al.* (2021)—a superset of data from Rocklin *et al.* (2017). We restrict ourselves to expert-designed proteins with explicitly intended topologies, as we use design methodology and intended topology to test model extrapolation. The data consist of primary sequences for small synthetic proteins, as well as experimentally measured stability scores Rocklin *et al.* (2017). We evaluate models on subsets of these data as well as near-comprehensive single-site mutations for seventeen proteins (three natural and fourteen synthetic), also originally reported in Rocklin *et al.* (2017).

2.2 Stability Prediction Tasks

We evaluate each model on two stability prediction tasks: (i) predicting stability for members of a single class (defined by secondary structure topology and design method) held out from the training set, and (ii) predicting stability for near-comprehensive single-site mutations of known proteins. For the held-out class approach, We use each class in turn as the test set and train each of the models on the other 26 topologies. In order to verify that the proteins in each topology were distinct, Singer *et al.* (2021) used BLAST to identify the nearest homolog of each protein from the other 26 classes. The results yielded low maximum bit scores and maximum percent identities (mean $\pm$ SD of 34.5 ± 5.1 and 47.0 ± 8.3 respectively), suggesting considerable differences among each topology class Singer *et al.* (2021); Altschul *et al.* (1990). This task evaluates the models' ability to generalize or extrapolate to new kinds of proteins, and could be important for downselecting proteins to test from a larger library of designs. For the single-site mutation predictions, we follow the approach of Rao *et al.* (2019), testing on a set constructed from 17 1-Hamming distance single-site mutations from Rocklin *et al.* (2017). Note that for our experiments we used a larger set of training data than in Rao *et al.* (2019). This task evaluates models' ability to predict the effects of small mutations on protein stability, and could be important for making incremental improvements to the stability of proteins with desired properties.

2.3 Metrics

For the held-out topology task, we evaluated the performance of each model using the R^2 goodness of fit score (i.e. the coefficient of determination), root mean squared error (RMSE), average precision (AP), and the area under the receiver operating characteristic curve (ROC AUC). R^2 and RMSE directly compare raw stability values to predicted stability scores. R^2 tells us what fraction of the variance in the data the model captures beyond simply predicting the mean. RMSE tells us how close, in stability score units, model predictions are to observed values, with more weight given to large errors. We prefer R^2 because different protein classes have different ranges of stability, some of which are quite small relative to the noise in the stability score assay; RMSE can therefore be misleading to compare across held-out classes. AP and ROC AUC depend on a definition of "stable", and measure the models' ability to rank stable proteins more highly than unstable ones. Average precision, essentially the area under the precision/recall curve, encapsulates the tradeoff between false positives and false negatives. The receiver operating curve measures true positives against false positives. AP tends to be more robust to class imbalance than ROC AUC, but ROC AUC sees wider use. We follow the convention adopted in Rocklin *et al.* (2017) and call a protein stable if its stability score is at least 1.0.

To evaluate performance in the single-site mutation prediction task we chose Spearman's rank correlation (ρ), as well as R^2 and RMSE.

Spearman's ρ assesses the degree to which predicted stability scores sort in the same order as observed stability scores.

2.4 Stability Prediction Algorithms

We evaluated the ability of two types of models to predict protein stability. One type consisted of convolutional neural network models—CNNs—trained on our stability data to directly predict stability scores. The other was protein sequence "embeddings", large models that attempt to convert an amino acid sequence into a fixed-dimensional vector that represents a variety of properties of the encoded protein. We then used gradient boosting to relate these embedding vectors to stability scores.

We considered four variants of CNN models. The most complex was the previously-published Evaluator Model (EM) from Singer *et al.* (2021). This model expands upon a typical CNN architecture by jointly predicting stability score and protein secondary structure, and learning from synthetic and natural proteins. We also considered three simpler variants of the EM: (i) trained with no natural proteins or secondary structure prediction, (ii) trained with natural proteins but not secondary structure prediction, and (iii) trained with secondary structure prediction but not natural proteins. Gradient boosting models were trained with XGBoost (XGB) Chen and Guestrin (2016) with the following parameters chosen via GridSearchCV: 1000 estimators, 50% of columns randomly being sampled per tree, a learning rate of 0.01, and a maximum tree depth of 8. These parameters optimized R^2 , and were chosen among an estimator range of 1, 100, and 1000, a random column sample range of 0.3, 0.5, and 0.7, a learning rate range of 0.001, 0.01, and 0.1, and a maximum tree depth range of 4, 6, and 8. The hyperparameters for each gradient boosting model were optimized independently, but all three models performed best with the same set of hyperparameters. We also evaluated gradient boosting models trained using the default XGB parameters: 50 estimators, 1 column to be randomly sampled per tree, a learning rate of 0.3, and a maximum tree depth of 6. We apply XGB to features generated by extracting embeddings from sequence data by the evolutionary scale model (ESM) Rives *et al.* (2020), the ProtBERT model Elnaggar *et al.* (2021), and the UniRep model Alley *et al.* (2019). Each of these models was trained in a self-supervised learning task; we used the published, pre-trained versions. Each yields a fixed-length vector for a given protein, which encodes some degree of biological information.

Introduced in 2017 by Vaswani *et al.* (2017), the Transformer architecture has evolved into a powerful model architecture for natural language processing tasks. The ProtBERT model Elnaggar *et al.* (2021) uses a deep transformer model, the Bidirectional Encoder Representations from Transformers (BERT) Devlin *et al.* (2019) architecture, applied to proteins. As a language model, BERT is trained to predict masked or "missing" words from sentences; ProtBERT treats protein primary sequences as sentences, and amino acids as words, and learns to predict masked amino acids. The ProtBERT model was trained on the UniRef100 database and yields a 1024-feature vector per protein, implicitly encoding structural and functional information.

The Evolutionary Scale Model (ESM, Rives *et al.* (2020)) is similar to ProtBERT in that it uses a series of deep Transformers Devlin *et al.* (2019) to learn the statistics and correlation of amino acid sequences and their contexts. It is a larger model, 650 million parameters to ProtBERT's 420 million, and is trained on a set of 250 million diverse proteins from the UniParc, UniRef50, and UniRef100 databases Rives *et al.* (2020). This process results in a 1280-feature vector for each protein sequence carrying information about biological, biochemical, and structural protein properties. Like ProtBERT, the model was trained to predict masked amino acids from the primary sequences of natural proteins.

The UniRep model Alley *et al.* (2019) implements a multiplicative Long-Short-Term-Memory recurrent neural network trained on 24 million

sequences from the UniRef50 database, without the use of structural or evolutionary data. This model processes the input sequence one amino acid at a time, and learns to predict at each point the next amino acid based on the sequence thus far. We used the UniRep Fusion variant, which yields 5700-dimensional vectors (concatenations of mean hidden state, final hidden state, and final cell state) containing approximations of encoded functional, structural, and evolutionary information.

XGBoost: We randomly selected 10% of the training data to use for validation, with early-stopping governed by the validation loss.

10k for validation For each CNN model, custom loss functions calculated the difference among a combination of the model's stability, natural comparator, and dssp loss given the variation of the model.

The XGBoost models have an early stopping value of 5 iterations for which the next decrease in loss is calculated. Each model was trained three times, with the final prediction for each model being the mean of the predictions from the three training runs.

Finally, we consider a Linear Regression ensemble model using a "blending" architecture for both the held-out topology and single-site mutation tasks. Blending is an ensemble learning technique that helps to improve upon the accuracy of results from a set of models by learning how best to combine their predictions. Each held-out topology task's ensemble model consists of the previously trained four CNN's and three tuned gradient boosting models. The single-site mutation task's ensemble model consists of the previously trained EM and three tuned gradient boosting models.

10k for validation held-out

3 Results and Discussion

3.1 Model Performance

3.1.1 Held-out Topology Task

The ESM gradient boosting model matched or outperformed all other individual models across all metrics for this task, with a mean R^2 of 0.22 across the 27 protein classes. When compared between the default XGBoost parameters and the GridSearchCV results, the ESM gradient boosting model outperformed both the ProtBERT and UniRep gradient boosting models. Given the ESM model had been trained on the largest dataset (86 billion amino acids across 250 million diverse proteins), with their highest-capacity model trained with 650 to 700 M parameters Rives *et al.* (2020), this suggests larger datasets could increase model performance.

The ensemble model, which combined the predictions of the EM and the three embedding models with tuned gradient boosting, yielded large improvements to both R^2 and AP (the most appropriate metrics for predictions of continuous and binary quantities, respectively). This suggests that either the different models are representing complementary aspects of proteins in terms of pertinence to stability, or are making use of their representations in complementary ways. Either way, it demonstrates the utility of combining different models.

3.1.2 Single-Site Mutation Task

The ESM gradient boosting model showed the highest ranking success when assessing how well the landscape of stability was learned (as suggested by Rao *et al.* (2019)) (Spearman $\rho = 0.73$). Both the EM and ESM gradient boosting model tied in performance ($R^2 = 0.5$). Scores were calculated from the mean of three runs per model.

We note that the UniRep gradient boosting model ($\rho = 0.67$) did not fare as well as was reported in Rives *et al.* (2020) on the same test set ($\rho = 0.73$), despite that we provide it with a larger training set. This likely reflects that they re-trained the full UniRep model on the stability data, rather than only training the simple downstream model (in their case a single dense layer, in our case gradient boosting). Alternatively, it is possible (though it would

Table 1. Held-Out Topology Task: Mean Model Performance

Model	R^2	RMSE	ROC AUC	AP
CNN	0.15	0.57	0.70	0.41
CNN_nat	0.14	0.57	0.83	0.40
CNN_sec	0.15	0.57	0.85	0.43
EM	0.16	0.57	0.86	0.44
ESM_XGB_untuned	0.19	0.55	0.88	0.44
ProtBERT_XGB_untuned	0.19	0.55	0.86	0.41
UniRep_XGB_untuned	0.13	0.58	0.85	0.37
ESM_XGB	0.22	0.55	0.88	0.44
ProtBERT_XGB	0.21	0.56	0.85	0.43
UniRep_XGB	0.17	0.58	0.87	0.41
Ensmbl_held-out	0.70	0.40	0.96	0.81

Table 2. Single-Site Mutation Task: Mean Model Performance

Model	R^2	RMSE	Spearman ρ
EM	0.50	0.53	0.72
ESM_XGB	0.50	0.53	0.73
ProtBERT_XGB	0.46	0.56	0.72
UniRep_XGB	0.41	0.58	0.67
Ensmbl_ssm	0.56	0.5	0.75

be surprising) that the additional training data actually hurt performance on the prediction task, or that the gradient boosting model was less able to capture the small changes in stability than the dense layer. Future work should evaluate these possibilities, and whether similar gains may be seen by updating the full ESM to solve a particular task.

4 Conclusion

Prediction of the effect of mutations on protein stability as well as predict protein stability itself is an important challenge for molecular and computational biologists. We show that multiple approaches to data-driven modeling of protein stability can be successful, though all depend on the benefits of large, community-accessible datasets. We evaluate these models in a prediction task where they have to extrapolate to a new class of previously-unseen protein, as well as in a task where they have to predict the effects on stability of small changes to a known protein.

We evaluate two kinds of models: relatively small convolutional neural networks trained directly on stability data, and large general-purpose embedding models trained on protein sequences and then interpreted through simple gradient boosting models fit to stability data. Both approaches are able to solve both classes of problem to a similar degree, with the ESM never worse than tied for best. We do not test a third approach, which involves retraining one of the large embedding models to tune it to the task at hand, as it requires substantially greater computational resources. Blending together multiple models provides a less computationally intensive way of achieving better results. Which approach is best likely reflects the needs and resources of the practitioner.

Acknowledgements

We are thankful for the support and computing resources from the Texas Advanced Computing Center (TACC).

Funding

This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory under Contract No. FA8750-17-C-0231 (and related contracts by SD2 Publication Consortium Members). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, or the United States Government.

References

- Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M., and Church, G. M. (2019). Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods*, **16**(12), 1315–1322.
- Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403–410.
- Burgess, D. J. (2015). Network effects of disease mutations. *Nature Reviews Genetics*, **16**(6), 317–317.
- Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. arXiv: 1810.04805.
- Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., and Rost, B. (2021). ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.
- Fang, J. (2020). A critical review of five machine learning-based algorithms for predicting protein stability changes upon mutation. *Briefings in Bioinformatics*, **21**(4), 1285–1292.
- Kulshreshtha, S., Chaudhary, V., Goswami, G. K., and Mathur, N. (2016). Computational approaches for predicting mutant protein stability. *Journal of Computer-Aided Molecular Design*, **30**(5), 401–412.
- Leman, J. K., Weitzner, B. D., Lewis, S. M., Adolf-Bryfogle, J., Alam, N., Alford, R. F., Aprahamian, M., Baker, D., Barlow, K. A., Barth, P., Basanta, B., Bender, B. J., Blacklock, K., Bonet, J., Boyken, S. E., Bradley, P., Bystroff, C., Conway, P., Cooper, S., Correia, B. E., Coventry, B., Das, R., De Jong, R. M., DiMaio, F., Dsilva, L., Dunbrack, R., Ford, A. S., Frenz, B., Fu, D. Y., Geniesse, C., Goldschmidt, L., Gowthaman, R., Gray, J. J., Gront, D., Guffy, S., Horowitz, S., Huang, P.-S., Huber, T., Jacobs, T. M., Jeliakov, J. R., Johnson, D. K., Kappel, K., Karanickolas, J., Khakzad, H., Khar, K. R., Khare, S. D., Khatib, F., Khrushch, A., King, I. C., Kleffner, R., Koepnick, B., Kortemme, T., Kuenze, G., Kuhlman, B., Kuroda, D., Labonte, J. W., Lai, J. K., Lapidoth, G., Leaver-Fay, A., Lindert, S., Linsky, T., London, N., Lubin, J. H., Lyskov, S., Maguire, J., Malmström, L., Marcos, E., Marcu, O., Marze, N. A., Meiler, J., Moretti, R., Mulligan, V. K., Nerli, S., Norm, C., Ó'Conchúir, S., Ollikainen, N., Ovchinnikov, S., Pacella, M. S., Pan, X., Park, H., Pavlovicz, R. E., Pethe, M., Pierce, B. G., Pilla, K. B., Raveh, B., Renfrew, P. D., Burman, S. S. R., Rubenstein, A., Sauer, M. F., Scheck, A., Schief, W., Schueler-Furman, O., Sedan, Y., Sevy, A. M., Sgourakis, N. G., Shi, L., Siegel, J. B., Silva, D.-A., Smith, S., Song, Y., Stein, A., Szegedy, M., Teets, F. D., Thyme, S. B., Wang, R. Y.-R., Watkins, A., Zimmerman, L., and Bonneau, R. (2020). Macromolecular modeling and design in Rosetta: recent methods and frameworks. *Nature Methods*, **17**(7), 665–680.
- Magliery, T. J. (2015). Protein stability: computation, sequence statistics, and new experimental methods. *Current Opinion in Structural Biology*, **33**, 161–168.
- Nisthal, A., Wang, C. Y., Ary, M. L., and Mayo, S. L. (2019). Protein stability engineering insights revealed by domain-wide comprehensive mutagenesis. *Proceedings of the National Academy of Sciences*, **116**(33), 16367–16377.
- Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X., Canny, J., Abbeel, P., and Song, Y. S. (2019). Evaluating Protein Transfer Learning with TAPE. *arXiv:1906.08230 [cs, q-bio, stat]*. arXiv: 1906.08230.
- Razban, R. M. and Shakhnovich, E. I. (2020). Effects of Single Mutations on Protein Stability Are Gaussian Distributed. *Biophysical Journal*, **118**(12), 2872–2878.
- Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. (2020). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*.
- Rocklin, G. J., Chidyausiku, T. M., Greshnik, I., Ford, A., Houlston, S., Lemak, A., Carter, L., Ravichandran, R., Mulligan, V. K., Chevalier, A., Arrowsmith, C. H., and Baker, D. (2017). Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*, **357**(6347), 168–175.
- Singer, J. M., Novotney, S., Strickland, D., Haddox, H. K., Leiby, N., Rocklin, G. J., Chow, C. M., Roy, A., Bera, A. K., Motta, F. C., Cao, L., Strauch, E.-M., Chidyausiku, T. M., Ford, A., Ho, E., Mackenzie, C. O., Eramian, H., DiMaio, F., Grigoryan, G., Vaughn, M., Stewart, L. J., Baker, D., and Klavins, E. (2021). Large-scale design and refinement of stable proteins using sequence-only models. Technical report. Type: article.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762.
- Wang, S., Tang, H., Shan, P., and Zuo, L. (2021). ProS-GNN: Predicting effects of mutations on protein stability using graph neural networks. preprint, Bioinformatics.

8 APPENDIX B – REFERENCES

- [1] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1–16.
- [2] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing, “Deep Neural Networks with Massive Learned Knowledge,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016, pp. 1670–1679.
- [3] M. P. Wahlström, Niklas and Schön, Thomas B. and Deisenroth, “Learning deep dynamical models from image pixels,” 2014.
- [4] M. P. Assael, John-Alexander M and Wahlstrom, Niklas and Schon, Thomas B and Deisenroth, “Data-Efficient Learning of Feedback Policies from Image Pixels using Deep Dynamical Models.”
- [5] H. Zimmermann, C. Tietz, and R. Grothmann, “Forecasting with Recurrent Neural Networks : 12 Tricks,” in *Neural Networks: Tricks of the Trade*, Berlin Heidelberg: Springer, 2012, pp. 687--707.
- [6] S. Klus and C. Sch, “On the numerical approximation of the Perron – Frobenius and Koopman operator,” 2015.
- [7] B. M. and R. M. and I. Mezić, “Applied Koopmanism,” *Chaos An Interdiscip. J. Nonlinear Sci.*, vol. 22, no. 4, p. 47510, 2012.
- [8] S. K. and C. Schütte, “Towards tensor-based methods for the numerical approximation of the Perron--Frobenius and Koopman operator,” *J. Comput. Dyn.*, vol. 3, no. 2, pp. 1–23, 2016.
- [9] and C. G. Ribeiro, Marco Tulio, Sameer Singh, ““Why Should I Trust You?’ Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135– 1144.
- [10] P. Ruvolo and E. Eaton, “ELLA: An Efficient Lifelong Learning Algorithm,” *ICML*, vol. 28, no. 1, pp. 507--515, 2013.

9 ACRONYMS

API	Application programming interface
BERT	Bidirectional Encoder Representations from Transformers
CDM	Combinatorial Design Model
CFU	Colony forming units
CNN	Convolutional neural networks
CPU	Central Processing Unit
CSV	Comma Separated Values
DARPA	Defense Advanced Research Projects Agency
DNA	Deoxyribonucleic Acid
ECFP	Extended-Connectivity Fingerprint for describing molecular structure as a string
EDMD	Extended Dynamic Mode Decomposition
ESM	Facebooks "Evolutionary-Scale Model" for featurizing protein sequences
GAN	Generative Adversarial Model
GPU	Graphics Processing Unit
ML	Machine Learning
LBNL	Lawrence Berkeley National Laboratory
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
NAND	Logical "not-and"
NRL	Naval Research Laboratory
SMILES	Simplified Molecular-Input Line-Entry System for describing molecular structure as a string
TACC	Texas Advanced Computing Center
XRD	X-Ray Diffraction