Cell Systems

Methods

BioAutoMATED: An end-to-end automated machine learning tool for explanation and design of biological sequences

Highlights

- We build a platform to automate machine-learning model search and optimization
- BioAutoMATED enables model interpretation and computer-aided design of sequences
- We benchmark BioAutoMATED on diverse datasets and generate biological insights
- Design decisions are abstracted for biologists with limited machine learning expertise

Authors

Jacqueline A. Valeri, Luis R. Soenksen, Katherine M. Collins, ..., Felix Wong, Timothy K. Lu, James J. Collins

Correspondence

jimjc@mit.edu

In brief

Valeri, Soenksen, Collins et al. develop a platform called BioAutoMATED that automates end-to-end machine learning for biological sequences by integrating three automated machine learning tools. With BioAutoMATED, researchers can automatically analyze, interpret, and design DNA, RNA, peptide, and glycan sequence datasets with minimal machine learning expertise.

Valeri et al., 2023, Cell Systems *14*, 525–542
June 21, 2023 © 2023 The Author(s). Published by Elsevier Inc.
https://doi.org/10.1016/j.cels.2023.05.007

CellPress

Cell Systems

CellPress
OPEN ACCESS

Methods

BioAutoMATED: An end-to-end automated machine learning tool for explanation and design of biological sequences

Jacqueline A. Valeri,[1,2,3,4,13] Luis R. Soenksen,[2,3,5,13] Katherine M. Collins,[3,6,7,8,13] Pradeep Ramesh,[3] George Cai,[3] Rani Powers,[3,9] Nicolaas M. Angenent-Mari,[1,2,3] Diogo M. Camacho,[3] Felix Wong,[1,2,4] Timothy K. Lu,[1,2,4,6,10] and James J. Collins[1,2,3,4,11,12,14,*]

[1]Department of Biological Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
[2]Institute for Medical Engineering and Science, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
[3]Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA 02115, USA
[4]Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA
[5]Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
[6]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
[7]Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
[8]Department of Engineering, University of Cambridge, Trumpington St, Cambridge CB2 1PZ, UK
[9]Pluto Biosciences, Golden, CO 80402, USA
[10]Synthetic Biology Group, Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[11]Harvard-MIT Program in Health Sciences and Technology, Cambridge, MA 02139, USA
[12]Abdul Latif Jameel Clinic for Machine Learning in Health, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[13]These authors contributed equally
[14]Lead contact
*Correspondence: jimjc@mit.edu

https://doi.org/10.1016/j.cels.2023.05.007

SUMMARY

The design choices underlying machine-learning (ML) models present important barriers to entry for many biologists who aim to incorporate ML in their research. Automated machine-learning (AutoML) algorithms can address many challenges that come with applying ML to the life sciences. However, these algorithms are rarely used in systems and synthetic biology studies because they typically do not explicitly handle biological sequences (e.g., nucleotide, amino acid, or glycan sequences) and cannot be easily compared with other AutoML algorithms. Here, we present BioAutoMATED, an AutoML platform for biological sequence analysis that integrates multiple AutoML methods into a unified framework. Users are automatically provided with relevant techniques for analyzing, interpreting, and designing biological sequences. BioAutoMATED predicts gene regulation, peptide-drug interactions, and glycan annotation, and designs optimized synthetic biology components, revealing salient sequence characteristics. By automating sequence modeling, BioAutoMATED allows life scientists to incorporate ML more readily into their work.

INTRODUCTION

The advent of massive, high-dimensional biological datasets in recent years has facilitated the widespread application of machine-learning (ML) methods to investigate and predict biological phenomena,[1,2] delivering exciting breakthroughs in genomics and promising more in fields such as systems biology,[1] synthetic biology,[1,3] and structural biology.[4] Medium- to large-scale biological sequence datasets, including those of nucleic acid, peptide, and glycan sequences, are ubiquitous. The use of ML on these datasets could aid investigators in extracting biological insights and accelerate the design of sequences with desired properties.

Computational analyses and ML techniques have become more accessible to scientists through online tutorials, open-source code, interactive notebooks, and software packages.[5–7] Nevertheless, ML expertise is often required to build, train, and deploy ML models. Various user-made decisions can dramatically affect the quality and performance of ML models. Understanding which design decisions matter and how to make the most appropriate decisions for any given dataset remain especially pertinent barriers for life science researchers with limited ML experience. Even among skilled ML practitioners, the appropriate selection of algorithmic techniques and tuning of model parameters—typically ranging in number from thousands to hundreds of millions—is difficult.[8,9] Indeed, manual model definition

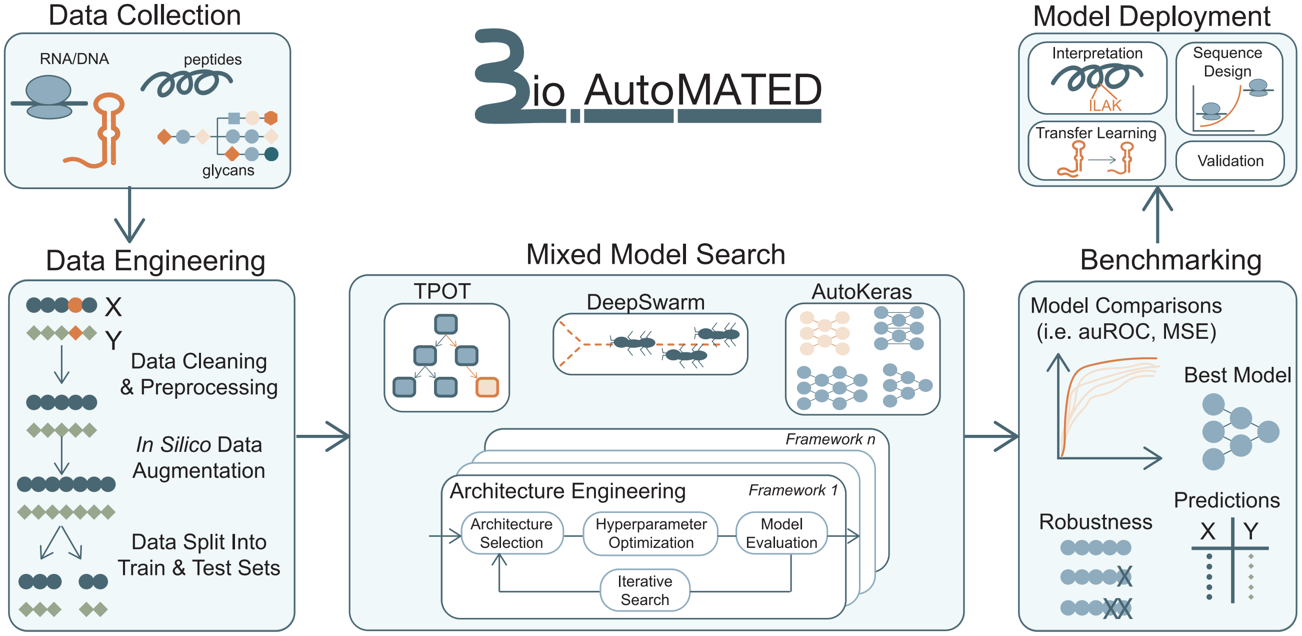


Figure 1. The BioAutoMATED framework automates the identification of predictive ML architectures for any set of nucleic acid, peptide, or glycan sequences

The integrated pipeline includes standard data processing, a mixed model search via three open-source automated ML (AutoML) libraries, and general benchmarking of models. Trained models are automatically deployed for interpretation and sequence design. BioAutoMATED also includes features to externally validate models with additional datasets and re-train models on new data using transfer learning.

and model parameter optimization require considerable expertise and time to implement[10–12] and may only provide limited benefit in many applied ML studies in biology and biomedicine.[8,13]

A promising avenue toward easing the adoption of ML to analyze biological datasets is the use of automated machine learning (AutoML). AutoML encompasses methods that automate the design and deployment of ML pipelines with minimal user intervention.[10,14] End-to-end AutoML would provide life scientists with easy data pre-processing, feature extraction, model selection and optimization, and performance evaluation.[10,14,15] AutoML techniques can automatically identify appropriate model architectures (the types of algorithms used in the model) and model hyperparameters (the parameters that can be tuned in any model to affect model performance). Because determining the optimal model architectures and hyperparameter values is oftentimes difficult, even for ML experts, appropriately implemented AutoML strategies may assist life scientists in building initial predictive models. Furthermore, AutoML may be useful for more experienced ML practitioners as a way to quickly generate baseline models to compare against or rapidly identify broad classes of models with encouraging performance.

There are currently a rich variety of AutoML tools available.[16–22] Many established AutoML tools search exclusively among classes of neural network models. However, among the most exciting AutoML tools in use are tree-based optimization methods that search among "shallow" or simpler scikit-learn-based models such as random forest classifiers.[23,24] These techniques, which may be better suited for smaller, sparser biological datasets than neural networks, have not yet been used jointly with neural architecture search methods to accelerate biological sequence analysis. Indeed, architecture choice is important for model performance[12] and recent studies suggest that there is no single "best" AutoML tool,[25] underscoring the importance of evaluating many classes of models on one platform. Thus, there is a need for AutoML integration within a scalable system that can also handle data pre-processing, model deployment, and system reporting.

Accordingly, here, we present the Biological Automated MAchine learning Tool for Explanation and Design, or BioAutoMATED, an end-to-end AutoML framework optimized for building models with nucleic acid, peptide, and glycan sequence inputs (Figure 1). The BioAutoMATED system integrates multiple open-source AutoML tools[23,26,27] with a diverse set of search mechanisms that allow for a wider breadth of architecture search spaces than previously reported. We augment this integration with automatic data importing, pre-processing, architecture selection, hyperparameter search, model training, model deployment, and performance reporting, all embedded into an easy-to-use, high-level programming interface that can be accessed in a Jupyter Notebook. We show that BioAutoMATED facilitates biological model interpretation by automatically using techniques that predict salient regions and motifs within sequences. We also include multiple methods, as reported in Bogard et al.[28] and Valeri, Collins, Ramesh et al.,[29] to enable the computer-aided design of novel biological sequences.

To demonstrate the utility of our platform, we evaluate BioAutoMATED by testing and deploying ML models based on a wide range of datasets relevant to gene regulation, peptide-drug interactions, and glycan annotation. The inputs, outputs, and salient features learned by our model in each case are as follows:

(1)Gene regulation. We explore the effect of ribosome-binding site (RBS) sequence inputs on translation efficiency outputs in *Escherichia coli*.[30] We demonstrate that models are predictive of translation efficiency and that we can achieve state-of-the-art performance comparable to that of manually tuned models in less than 30 minutes of runtime—from only 10 lines of user input. These results enable rapid and accurate prediction of RBS sequences for synthetic biology applications.

(2)Peptide-drug interactions. We explore the effect of antibody sequence inputs on drug-binding affinity outputs, focusing on antibody sequences varying in their CDR-H3 regions and how they lead to differential binding to the drug ranibizumab as a target antigen.[31] We show that BioAutoMATED can produce models that, although not as high performing as a manually tuned model comprising six different custom architectures, remain highly predictive. Our models can be readily used to inform the development of antibody variants that have improved target specificity based on sequence alone.

(3)Glycan annotation. We explore the effect of glycan sequence inputs on the outputs of taxonomic group classification and immunogenicity in humans. Building on a previously published database of short glycan sequences,[32] we use BioAutoMATED to identify a best-performing model for the prediction of glycan immunogenicity to humans. We further show that a model can identify phylogenetic domains based on sequence information, aiding the annotation of these sequences.

Finally, to demonstrate its additional features, we apply BioAutoMATED in an end-to-end case study relevant to RNA toehold switch design[29,33] for detecting nucleic acids. In particular, we show that BioAutoMATED can find sequence regions of functional relevance for toehold switches, prokaryotic riboregulators that sense the presence of trigger RNAs. Building on previously generated data, we show that BioAutoMATED produces ML models that predict the performance of toehold switches that detect RNA from Zika virus.

Taken together, our applications of BioAutoMATED demonstrate its efficient identification and training of ML models with minimal user input for biologically relevant sequence datasets, outperforming other biology-focused AutoML tools and achieving performances comparable to those of manually defined architectures. BioAutoMATED thereby offers a versatile platform for life scientists to easily develop and deploy ML models built on sequence-based datasets.

RESULTS

BioAutoMATED automates the development of ML models

BioAutoMATED is a highly customizable end-to-end Python framework with the capacity to handle diverse sequence-based datasets, including those of nucleic acids, peptides, and glycans (Figure 1). The inputs to BioAutoMATED are biological sequences and can be DNA, RNA, amino acid, and glycan sequences of any length, type, or function. Based on these inputs, BioAutoMATED generates models that can predict function from sequence information alone. Here, the function may be any user-defined function including, for example, the transcription efficiency of a regulatory gene element, the enrichment of a peptide sequence in a binding assay, or the pathogenicity of glycan sequences in a microbiome sample.

To operate our system, the user first uploads a CSV or Excel spreadsheet to one of our provided Jupyter notebooks with a list of biological sequence inputs and their corresponding target value outputs that the user wishes to predict. These target values may be continuous values, binarized values (pre-processed to zeros and ones), or text labels (for example, ‘‘bacteria’’ and ‘‘human’’). Correspondingly, BioAutoMATED can produce multiple types of models: (1) binary classification models that are assessed on their ability to predict negative ‘‘0’’ and positive ‘‘1’’ classes, (2) multi-class classification models that are assessed on their ability to predict multiple classes (e.g., ‘‘bacteria,’’ ‘‘human,’’ and ‘‘virus’’), and (3) regression models that are assessed on their ability to predict continuous values. Although the user can provide target values in any range, for simplicity, BioAutoMATED can automatically normalize the target values (e.g., to lie between a minimum value of -1 and a maximum value of +1). For binary classification tasks on sequences with continuous labels, the platform thresholds between positive and negative classes to binarize the data, either automatically or with a user-provided cut-off value. For example, if the user wishes to predict ‘‘good’’ or ‘‘bad’’ classes of sequences using a model trained on sequences with target values between -1 and 1, the user can indicate that sequences with values greater than 0.75 should be treated as positive (‘‘good’’) examples and vice versa.

The user’s dataset may consist of any number of biological sequences of a given type (e.g., nucleic acid, amino acid, or glycan), but we recommend some guidelines. For optimal results and reasonable timeframes, we recommend datasets between 1,000 and 500,000 sequences and sequences up to several hundred ‘‘subunits’’ (nucleotides, amino acids, or glycan monosaccharides and connecting bonds) in length (see STAR Methods for more information). After the user uploads their selected dataset, BioAutoMATED proceeds relatively hands-free. Upon selection of a small set of user-defined options such as the sequence type, permitted time to run each AutoML search, and desired prediction task (regression, binary classification, or multi-class classification), BioAutoMATED conducts all subsequent data pre-processing, including cleaning of incorrectly formatted inputs. The input sequence alphabet, or set of nucleotides, amino acids, or monosaccharides and bonds represented in the user’s dataset of sequences, is automatically inferred. For example, the alphabet for peptide sequences is composed of all amino acids which appear in the input data, which allows for the relative importance of input letters to be learned rather than assuming all amino acids are represented in the dataset. This alphabet is then used to generate vector representations for all input sequences (see STAR Methods).

After automatic data pre-processing, BioAutoMATED performs an AutoML model search by implementing modified versions of DeepSwarm,[27] AutoKeras,[26] and Tree-based Pipeline Optimization Tool (TPOT)[23] to jointly search different architectures and hyperparameters. We selected these three AutoML programs because of their broad and distinct architecture search spaces and reported high performance in recent benchmark evaluations across multiple datasets.[34,35] In brief, their methodologies are as follow:

(1)AutoKeras[26] is an open-source framework to efficiently search neural network architectures, in which neural network kernels and tree-structured acquisition functions are used to iteratively search for optimal architectures.

(2)DeepSwarm[27] performs neural architecture search based on simulated ant colony behavior (or swarm-based) algorithms, wherein a population of search agents sense local and global paths of previous architecture explorations as a way to collectively search for optimal convolutional neural network architectures.

(3)The TPOT framework,[23] built on top of scikit-learn, covers non-neural network architectures using genetic programming, feature engineering, and self-learning algorithms.

BioAutoMATED produces results for all three AutoML systems that can be jointly evaluated and compared. A key reason for this is that our integration methodology depends only on standardized inputs and outputs. Consequently, other AutoML systems could be feasibly incorporated by adapting the current platform.

BioAutoMATED accommodates different data types, sizes, and processing options

BioAutoMATED automatically offers several options for more user involvement in model selection, controls, and dataset expansion. For example, a systems biology researcher may not be comfortable determining the amount of data needed to train a typical ML model, as this judgment depends on a variety of factors, such as the difficulty of the prediction task, quality of the data, and dimensionality of the sequence space. To evaluate if the amount of training data were sufficient for optimizing the model, the user can opt-in to run a data ablation experiment with models trained on randomly selected datasets of decreasing sizes (Figure S1). This feature may prove helpful for future experimental planning; for example, if models achieve high performance when trained on only 20,000 sequences, it may not be necessary to conduct the expensive procedures needed to generate data for a full 40,000 sequences. Additionally, the BioAutoMATED platform automatically calculates the elapsed time spent on each AutoML stage (architecture search, dataset ablation studies, scrambled controls, etc.), which can be evaluated by the user (Figure S2).

For nucleic acid sequences, the user is also offered the option to augment the dataset by computing complementary sequences, reverse complementary sequences, or both, and by spiking these synthetic sequences into the dataset with the labels of the original sequence (Figure S3). This augmentation step is inspired by data augmentation in image classification, in which the dataset size can be artificially increased by cropping, rotating, or transforming the original images.[36] In some cases, with small training datasets, this opt-in data augmentation step can boost model performance, especially for sequences in which the complement or reverse complement is expected to confer a similar meaning as the original sequence. Because all sequences in a training set must have the same length to be compatible with the three AutoML tools, BioAutoMATED provides several options to standardize the length of datasets with heterogeneous sequence lengths. Sequences can be padded to the maximum length, truncated to the minimum length, or standardized to the average sequence length (Figure S4).

To extend the applicability of BioAutoMATED models, the user is provided with a transfer learning module in which new data can be used to ‘‘fine-tune’’ existing models (see STAR Methods). Users can re-train any existing DeepSwarm, AutoKeras, or TPOT model produced by BioAutoMATED with additional datasets, further enabling model predictive capability even as datasets grow in size and quality.

BioAutoMATED provides interpretation tools to facilitate sequence analysis and model explanation

Before demonstrating the use of BioAutoMATED to create predictive models, we note here some additional features that are relevant to different applications. A key feature is interpretability: for many researchers, creating a predictive model is often just a step toward knowledge generation and not the end goal.[37] Especially in AutoML, where models can be developed in a ‘‘black-box’’ fashion, it is useful to enable model transparency and leverage trained models to understand the underlying biology.

To this end, BioAutoMATED provides interpretability tools to automatically analyze the best models from each of the three implemented AutoML programs and interrogate model behavior. These tools can assist users in identifying regions of importance in their sequences, as well as motifs that may be functionally relevant. These tools consist of the following:

(1)Feature important plots and *in silico* mutagenesis. These tools can reveal the relative importance of each position in a sequence but do not provide any information about identities of the subunits (e.g., nucleotides) lending to that importance. To illustrate the utility of such tools, we have used a feature importance plot to assess TPOT models in which each position in an average sequence is ascribed an importance by the model. To assess BioAutoMATED models more broadly, we also developed a simple mutagenesis technique to evaluate each position’s effect on the sequence’s prediction score in a model-agnostic way. The mutagenesis protocol automatically changes each subunit of a sequence to all other possible options (‘‘subunit changes’’). BioAutoMATED generates plots of the standard deviation of predicted scores of sequences with subunit changes. Doing so reveals ‘‘hotspots,’’ or regions of elevated functional importance, by querying variable inputs instead of directly investigating the model. This *in silico* sequence mutagenesis strategy provides model-agnostic explanations for all model types, as not every type of AutoML model is amenable to more complex gradient-based interpretation methods (for example, the scikit-learn-based models found by TPOT cannot be directly interrogated by interpretability techniques designed for neural networks as described below).

(2)Visualization of model attention. Tools inspired by computer vision, such as saliency maps and class activation maps, can be used on some neural network models to reveal positions of importance and specific subunits that the model preferentially focuses ‘‘attention’’ on. We convert these maps to sequence logos, wherein a higher attention value increases the size of the letter in the sequence, yielding a visually interpretable view of attention. BioAutoMATED offers both class activation maps and saliency maps to visualize model attention with respect to the input sequences (see STAR Methods for details and a

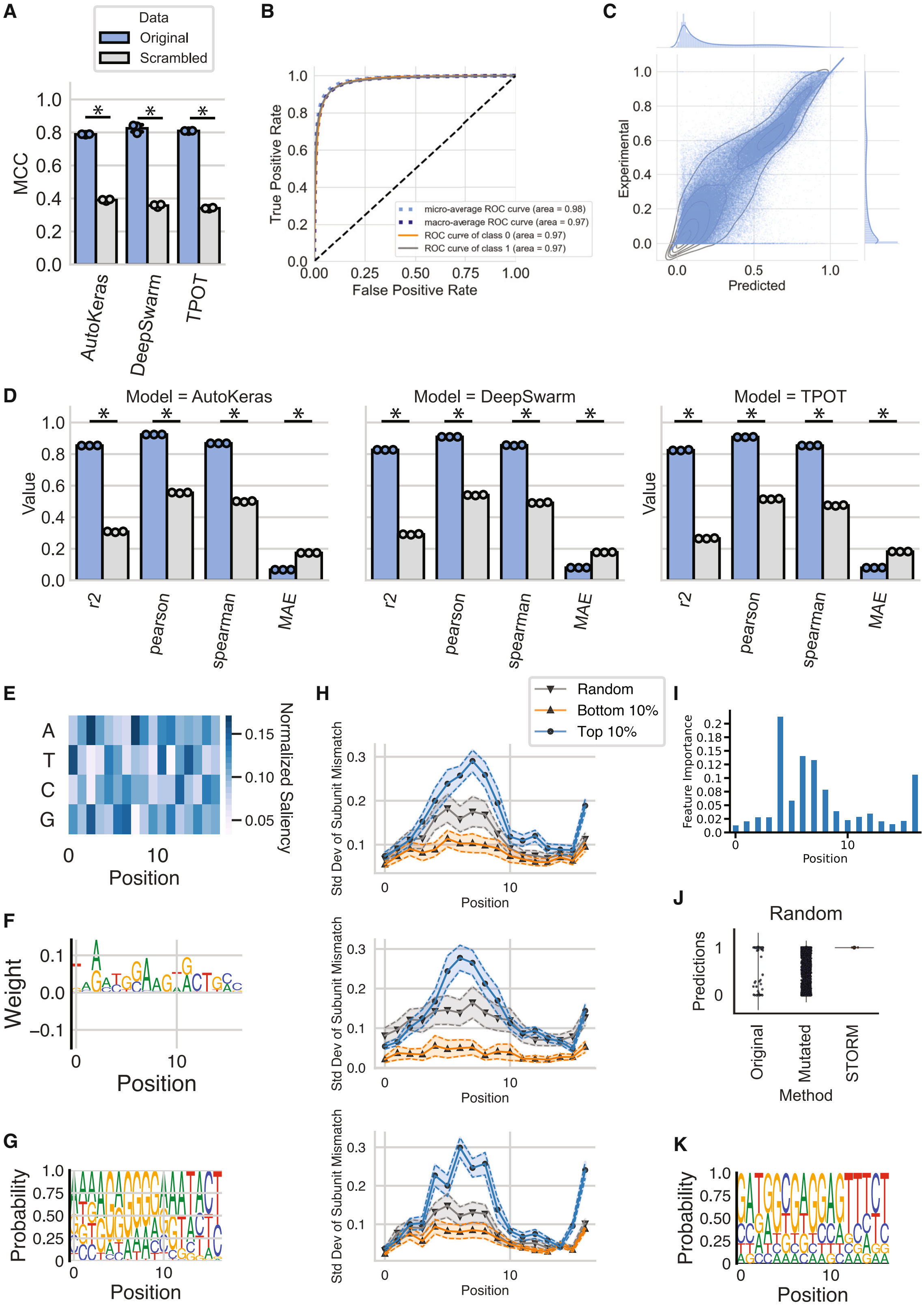


Figure 2. BioAutoMATED generates models that predict translation efficiency from ribosome binding site sequences

(A and B) BioAutoMATED produces models that achieve high Matthews Correlation Coefficients (MCCs) on classifying sequences as “good” or “bad” RBS sequences, better classifying the original sequences (blue) as compared with models trained on scrambled controls (gray).

(legend continued on next page)

comparison of the techniques). To facilitate meaningful comparison of these tools, we also generate sequence logos of the raw data. For any input dataset, BioAutoMATED constructs sequence logos for a random set of sequences, top-performing sequences, poorly performing sequences, and/or sequences with different labels (Figure S5).

Together, these interpretability techniques offer several different views of any input sequence which the user can leverage to inform experimental follow-ups, for example, deep sequencing all mutants in a targeted region of the sequence or researching potential binding motifs that emerge from this analysis.

BioAutoMATED enables computer-aided design of biological sequences

Another key feature that BioAutoMATED provides is *de novo* design. These functionalities build on growing interest in using ML for sequence design, and recent approaches have expanded the boundaries of nucleic acid[38,39] and peptide sequence design.[40,41] Specifically, after training any BioAutoMATED model, the user can design *de novo* sequences by specifying a target value (e.g., 75th percentile of all sequences) that the *de novo* designed sequences should obtain. BioAutoMATED enables two methods for this task: random mutagenesis and directed design.

Random mutagenesis allows users to assess random, subunit-based mutations of sequences as specified previously. This simplistic method may be sufficient for cases in which optimal sequences are only a few mutations away from existing sequences, or with datasets of relatively short sequences whose search spaces can be thoroughly explored via this "brute force" method. This technique is compatible with all models generated by BioAutoMATED, including simple models generated by TPOT.

In many cases, more sophisticated techniques can generate more diverse or higher-performing sequences. We therefore implemented a directed design method, gradient ascent, by extending the SeqProp framework described in Bogard et al.[28] and adapted in Valeri, Collins, Ramesh et al.[29] as STORM. Here, any user-specified constraint involving specific sequence subunits, specific positions, and/or base-pairing constraints can be specified. The entire sequence is allowed to vary its composition simultaneously, which occurs in a directed manner using gradient ascent, resulting in simultaneous mutations of all allowed positions using the learned model weights to optimize the target value. This tool therefore allows BioAutoMATED to explore complex sequence spaces more fully than random mutagenesis alone, and it is agnostic to the model and sequence alphabet (see STAR Methods for more information).

BioAutoMATED elucidates ribosomal binding site design for gene regulation

We present a series of case studies using our integrated AutoML framework to answer key biological questions on previously published experimental datasets. We first explored the relationship between gene regulatory element sequence and effect, specifically the effect of RBS sequence on translation efficiency in *E. coli*.[30] Höllerer et al. employed a DNA-based phenotypic recording technique using a site-specific recombinase to investigate the translation kinetics of ~276,000 different 17-nucleotide long RBS sequences.[30] We applied BioAutoMATED to this dataset to predict translation efficiency from sequence alone.

BioAutoMATED readily generated binary classification models with Matthews Correlation Coefficients (MCCs) ~0.8 for all three search algorithms (Figure 2A). The most predictive model was identified and trained by the DeepSwarm algorithm, with an auROC of 0.971 and MCC of 0.825 (Figure 2B). In an analogous regression task, BioAutoMATED identifies a model architecture for the RBS dataset with the AutoKeras search method with an average Pearson R of 0.924 and an average R^2 of 0.854 (Figures 2C and 2D), in line with the manually defined and tuned models reported by Höllerer et al.[30]; these models achieved an R^2 between 0.8 and 0.95 using residual network (ResNet[42]) architectures.

We tested this regression model on the same test set of RBS sequences as Höllerer et al.[30] and achieved similar performance with an auROC of 0.939, Pearson R of 0.931, and R^2 of 0.867—compared with an R^2 of 0.927 from Höllerer et al. (Table S1). Although BioAutoMATED performs slightly worse than the best reported model, the top-performing DeepSwarm regression model was found in just 26.5 minutes (Figures S2A and S2E) and required ten lines of user input (in contrast to over 750 lines of code for Höllerer et al.'s model). Additionally, Höllerer et al.'s model required the manual selection of architectures and hyperparameters for optimization using a random grid search, a

(C and D) BioAutoMATED produces regression models that can predict continuous values of translation efficiency from the same RBS dataset. The best AutoKeras regression model has an R^2 value of 0.854. Significantly more predictive models for original sequences than scrambled sequences were obtained, as assessed with R^2 values, Pearson R coefficients, Spearman R coefficients, and mean absolute error (MAE). For **(**A) and (D**)**, points correspond to one of N = 3 cross-validation folds (see STAR Methods for details) and asterisk denotes $p < 0.005$, two-sided t test.

(E) RBS sequences were evaluated via a saliency map computed for N = 100 randomly selected sequences using the DeepSwarm binary classification model. Normalized saliency represents arbitrary model attention units.

(F) The saliency map and (F) its corresponding sequence logo display Shine-Dalgarno-like motifs described in Höllerer et al.[30]

(G) This Shine-Dalgarno-like motif can also be visualized in the sequence logo for the top 10% of RBS sequences (N = 50 sequences).

(H) *In silico* sequence mutagenesis methods facilitate BioAutoMATED model explanation for all sequence types in a model-agnostic way, here showing consensus between DeepSwarm (top), AutoKeras (middle), and TPOT (bottom) models. RBS models show elevated importance of positions 5–8 for all models. Error bars represent the 95% confidence intervals for N = 50 samples of each class.

(I) The feature importances of the TPOT regression model show elevated importance for positions 5–8.

(J) From a selection of N = 100 random RBS sequences, directed STORM optimization produces *de novo* designed sequences with higher translation efficiency scores as predicted with the best DeepSwarm binary classification model.

(K) STORM-designed RBS sequences optimized from random sequences can be visualized with sequence logos, showing a predominance of A and G nucleotides.

process requiring some ML expertise. BioAutoMATED is not limited by these requirements.

To better understand the model predictions and the effects of sequence motifs on translation efficiency, we performed a control experiment wherein the input sequences were scrambled. BioAutoMATED enables this experiment to be performed automatically and generates scrambled sequences by shuffling the order of nucleotides. The resulting dataset has the same number of sequences and can be used to train a model that predicts translation efficiency given only nucleotide frequency. When evaluating the RBS models with this scrambled control, we found that these models exhibit significantly worse performance on nucleotide composition alone for RBS sequences (Figures 2A and 2D). These results quantitatively confirm the importance of RBS sequence order to translation efficiency and are consistent with models of translation initiation.[30]

Building on our finding that RBS sequence order affects translation efficiency in *E. coli*, we asked whether any motifs or regions within the RBS are particularly important. BioAutoMATED allows us to address this question using its interpretability tools. We calculated RBS sequence logos from the saliency map (Figures 2E and 2F) and found that the logos contain AGATGG and TGGAAG motifs, which resemble the Shine-Dalgarno-like motifs (‘‘AGGAGG and subsequences thereof’’) reported in Höllerer et al.[30] This motif can also be cross-referenced with the sequence logo of the best-performing raw experimental sequences (Figures 2G and S5A), which includes an ‘‘AGAGGG’’ sequence and a preponderance of A and G nucleotides. Although the saliency map motifs are only marginally enriched compared with background, this motif appears in the middle of the sequences, around positions 5–8: these are positions that we found to have the highest predicted change to model score (Figure 2H) and the highest feature importance (Figure 2I). These findings illustrate the range of results that can be automatically produced by BioAutoMATED to facilitate interpretation.

To apply BioAutoMATED to RBS design, we applied its random mutagenesis and the STORM directed design tools to a starting subset of sequences, a random set of sequences (Figure 2J). We found that the directed design method designs RBS sequences with high predicted translation efficiency, whereas the naive random mutagenesis method resulted in a wide range of predicted translation efficiency. The STORM-optimized sequences, visualized with sequence logos (Figure 2K), can help BioAutoMATED users identify motifs or sequences that can be experimentally tested to validate high translation efficiency. For example, the predominance of A and G nucleotides resembles the sequence logos described above and the most-improved sequence ‘‘GATGGCGAGGAGTTTCT,’’ which improves from a predicted score of 0 to a predicted score of 1, contains both GAGGAG and AGGAGT motifs. Together, our results show that BioAutoMATED can productively generate predictive models, elucidate biologically important sequence motifs, and facilitate *de novo* RBS design for studies of gene regulation.

BioAutoMATED enables optimization of drug-binding antibody sequences

We next investigated precision targeting of human IgG antibodies, asking how antibody sequences varying in their CDR-H3 regions have different binding affinities to the drug ranibizumab.[31] We applied the BioAutoMATED platform to a peptide sequence dataset (N = 67,769 sequences) with target values corresponding to the enrichment of peptide binding against ranibizumab, as collected and analyzed by Liu et al.[31] Here, phage display was used to assay a library of tens of thousands of antibody fragment sequences for their binding affinities to multiple drugs, with the goal of predicting antibody affinity from sequence and using these trained models to design *de novo* antibodies.[31]

As with the RBS dataset, BioAutoMATED identified accurate model architectures that are significantly more predictive than models trained on scrambled peptide sequences (Figure 3A). Our best model achieved high predictive capability, with an auROC of 0.880 and an MCC of 0.748, indicating that antibody affinity can be accurately predicted from sequence. Additionally, we evaluated the BioAutoMATED framework on the same held-out test set as Liu et al.[31] The best convolutional neural network from Liu et al. achieved an auROC of 0.960 and Pearson R of 0.79, whereas our best regression model achieved an auROC of 0.868, Spearman R of 0.678, and Pearson R of 0.659 (Figure 3B; Table S1). Although Liu et al.’s model performs better than our best regression model, it is important to note that Liu et al.’s model was manually designed and comprised six different custom convolutional architectures, each with different network-layer widths and filters.[31] These types of adjustments and parameter selection processes require ML expertise, which BioAutoMATED abstracts away from the user. Here, the encouraging performance of BioAutoMATED models still allows us to generate predictive models and address related questions of interpretability and design.

To study interpretability, we asked whether any amino acids were particularly important to the predicted antibody affinity for binding to ranibizumab. Generating the feature importance (Figure 3C) and *in silico* mutagenesis plots (Figure 3D), we found that the first two and last two positions in the sequence were clearly predicted to be of minimal importance to binding affinity. This suggestion is further supported by the saliency map (Figures 3E and 3F). A closer examination of the saliency map sequence logos (Figure 3F), as well as the top-performing peptide sequences visualized by sequence logos (Figure S5B), reveals an FDY motif in positions 15–17. This motif was also described in figure 6 of Liu et al.,[31] in which it was found to correspond to the top-performing sequences. The saliency map also reveals amino acids of decreased importance, such as methionine, cysteine, and asparagine, which have barely any saliency values ascribed to them across the sequence. These observations were again supported by Liu et al.,[31] who used their trained ML models to optimize sequences and found that neither cysteine nor asparagine appeared in any of their ML-optimized sequences.

We next used STORM to design *de novo* peptide sequences with high predicted enrichment of ranibizumab binding (Figure 3G). Consistent with our prior results, we found that STORM-optimized peptide sequences display the aforementioned FDY motif (Figure 3H), as well as a preference for alanine in the beginning and end of the sequence. Our best designed sequence, ‘‘AEGHSLYGQDTTWPHFDYAA,’’ has a predicted

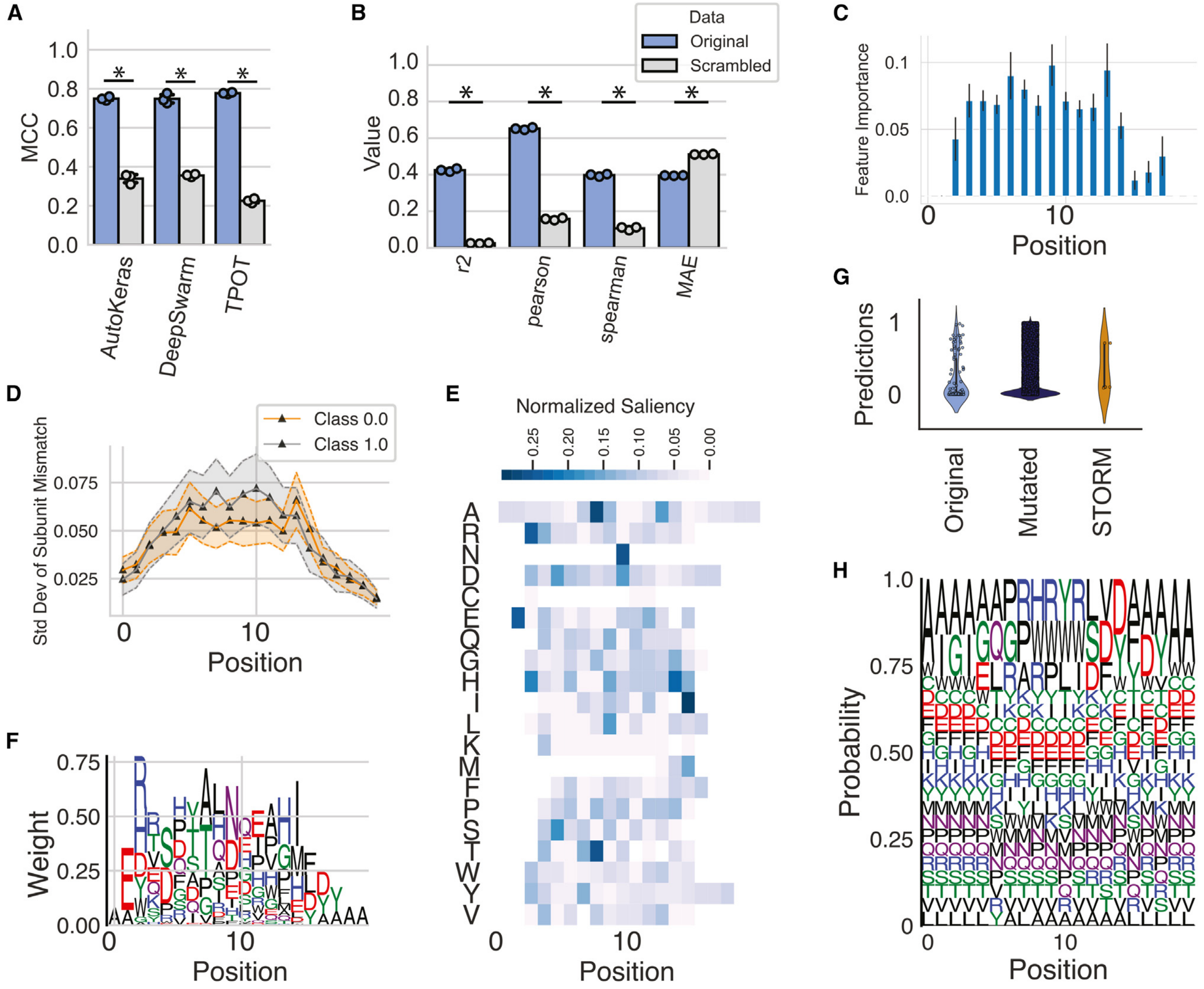


Figure 3. BioAutoMATED generates models that predict ranibizumab-binding affinity from peptide sequences

(A) Peptide sequences were tested in the BioAutoMATED pipeline, achieving higher MCCs on the original sequences (blue) compared with models trained on scrambled controls (gray).

(B) Peptide sequence regression is significantly more predictive for original sequences (blue) than scrambled sequences (gray) for all models, as assessed with R^2 values, Pearson R coefficients, Spearman R coefficients, and mean absolute error (MAE). Shown is the TPOT regression model. For (A) and (B), points correspond to one of N = 3 cross-validation folds and asterisk denotes $p < 0.005$, two-sided t test.

(C and D) (C) The binary classification models trained on peptide sequences evaluated with TPOT feature importance or (D) *in silico mutagenesis* displays low importance for ranibizumab-binding enrichment for the positions at either end of the sequences (N = 50 sequences). Error bars representing standard deviation of the computed feature importance are shown if the analyzed model is a collection of multiple "estimators" in scikit-learn, such as a random forest.

(E and F) (E) Peptides were evaluated via a saliency map and (F) its corresponding sequence logo on the DeepSwarm binary classification model, demonstrating the FDY motif in positions 15–17 similar to that seen in Liu et al.[31] for N = 100 sequences. Normalized saliency represents arbitrary model attention units. Note that saliency map sequence logos are plotted with respect to weights, whereas activation map sequence logos are plotted with respect to probability; this is an arbitrary design choice (see STAR Methods).

(G) Peptide sequences can be improved for ranibizumab-binding enrichment with directed optimization based on the DeepSwarm binary classification model (N = 100 starting sequences).

(H) Peptide sequences that have been designed with STORM display the same FDY motif as described earlier.

model score of 1.080, does not contain any methionine, cysteine, or asparagine, and does not appear in the original dataset. This sequence and other generated peptide sequences can be readily assembled, and phage display experiments could again be used to validate the model predictions of high binding affinity to ranibizumab.

BioAutoMATED accurately classifies glycans according to function

To demonstrate how BioAutoMATED can be applied to more complex search spaces, we used BioAutoMATED to accurately classify glycans, large carbohydrate biopolymers made up of monosaccharides in unique arrangements with extensive branching that play a crucial role in host-pathogen interactions, cell-cell communication, cell adhesion, autoimmunity, and cancer.[32,43] Recent successes in the use of ML for glycobiology[32,44] have fueled interest in applying computational analyses to growing databases of glycan sequences. Here, we build on work by Bojar et al.,[32] who provide a curated database of glycans called ''SugarBase,'' and use this dataset to explore the effects of glycan sequence composition on both taxonomic group classification, as well as on immunogenicity to humans. These glycan sequences are composed of both monosaccharides (e.g., galactose) and bonds (e.g., the α-1,3 bond), as denoted with the standard symbol nomenclature for glycans.

First, we used BioAutoMATED for multi-class classification of phylogenetic domains. Classification of taxonomic group based on sequence alone is an open problem relevant to microbiome sample analysis because many datasets lack appropriate phylogenetic annotation. BioAutoMATED models performed well (Figure 4A), with the best-performing TPOT models achieving MCCs of 0.594, 0.895, 0.875, and 0.351 for the archaea (N = 34 sequences), bacteria (N = 5,856 sequences), eukarya (N = 6,635 sequences), and virus (N = 149 sequences) domains, respectively. These results are comparable to the domain MCC values reported in Bojar et al.[32] ($\sim$0.869) and are significantly higher than corresponding values for scrambled controls, which we found to be 0.0, 0.623, 0.605, and $-$0.002, respectively.

Second, we used BioAutoMATED to perform binary classification on a set of labeled immunogenic or non-immunogenic glycan sequences (N = 1,320 sequences) from Bojar et al.[32] We found that BioAutoMATED models trained on these sequences are highly predictive, with the top TPOT model achieving an auROC of 0.936, an MCC of 0.873, and an accuracy of 92.7% (Figure 4B). These models are comparable to the top-performing language model in Bojar et al., which achieved an accuracy of 92%. The auto-generated scrambled control evaluated with the TPOT model achieved an auROC of 0.763 and MCC of 0.526, a statistically significant decrease from the non-scrambled model and more predictive than the language models from Bojar et al.[32] trained on scrambled glycan sequences (which achieved an accuracy of 51%). Bojar et al.[32] reported that their control models that did not treat glycan sequences as a language—analogous to BioAutoMATED—achieved accuracies ranging from 80% to 88%, due to the functional importance of order and patterns in glycans. It is possible that preserving glycans as sequences rather than passing them into language models improves learning from monosaccharide and bond composition alone.

Lastly, we used BioAutoMATED's interpretation and design tools to explore and leverage the monosaccharides that disproportionately affect glycan immunogenicity. Generating class activation sequence logos for the effects of sequences on immunogenicity, we found that mannose and rhamnose monosaccharides exhibit some importance to immunogenicity predictions (Figure 4C). The implications of these two monosaccharides are further supported with raw sequence logos (Figure S5C), in which we found that immunogenic sequences are rhamnose-rich and non-immunogenic sequences are mannose-rich (similar to the clusters shown in Figure 2E of Bojar et al.[32]). Furthermore, immunogenic importance is associated with fucose (Figures 4C and S5C), one of the reported immunogenic monosaccharides from Mohapatra et al.[44] that arose from a graph neural network-based substructure analysis. In both the activation map sequence logos (Figure 4C) and the raw data sequence logos (Figure S5C), Gal (and the a-1-3 bond, though this bond is not specific) was predicted to be important, consistent with previous findings that a-1,3-Gal (galactose-α-1,3-galactose) can be immunogenic to humans.[46]

Using STORM as a proof of principle, we found that glycan sequences can be designed to be optimally immunogenic (Figure 4D). Consistent with Figures 4C and S5C, the *de novo* designed immunogenic glycan sequences demonstrated some degree of rhamnose enrichment (Figure 4E). The *de novo* immunogenic glycan sequences were also marked by the presence of colitose monosaccharides (Figure 4E), a monosaccharide which was not present in the raw sequence logos and occurs in the LPS of some Gram-negative bacteria.[45] As with its predictions for RBS and antibody sequences, these BioAutoMATED-generated predictions for glycans could be empirically tested, and doing so could expand our understanding of glycan biology.

BioAutoMATED facilitates the design of toehold switches that detect nucleic acids

Finally, we used BioAutoMATED to optimize toehold switches to have high sensitivity and specificity for specific nucleic acids. We explore this question as a case study to assess how ML in general, and BioAutoMATED in particular, can assist an experimental effort in designing toehold switches for detecting the presence of a new virus. Toehold switches are riboregulators that produce a diagnostic output in response to exposure to a pre-defined ''trigger'' sequence, which is often a viral or synthetic circuit component. Here, we used a riboregulator dataset previously generated and analyzed by Angenent-Mari, Garruss, Soenksen et al.[33] and Valeri, Collins, Ramesh et al.[29] This riboregulator dataset, one of the largest of its kind, measured the sensitivity of 91,534 toehold switches (as measured by GFP production) in the presence of a 30-nucleotide trigger sequence using high-throughput flow-seq.

Predicting toehold switch activity from sequence alone is important, because secondary structure inputs have been the most commonly used determinant of riboregulator functionality.[47] Using BioAutoMATED, we found that a convolutional neural network was best at predicting the responsiveness of toehold switches based on sequence information alone (Figure 5A). The binary classification model performance of auROC of 0.925 outperforms that of manually tuned convolutional neural networks and language models reported in Valeri, Collins, Ramesh et al.,[29] both with auROC $\sim$0.85. The highest performing regression model identified by BioAutoMATED has an R^2 of 0.680 (Figure 5A), again comparable to results reported in Valeri, Collins, Ramesh et al.[29] Additional automated analyses indicated that our dataset was likely appropriate for the models at hand and that more training data would result in minimal performance improvements (Figure 5B).

To investigate what the best BioAutoMATED model learned about toehold switch biology, we applied the model to scrambled sequences, which indicated that the model uses more than just the nucleotide composition in making its predictions (Figure 5C). We then generated sequence logos of the raw

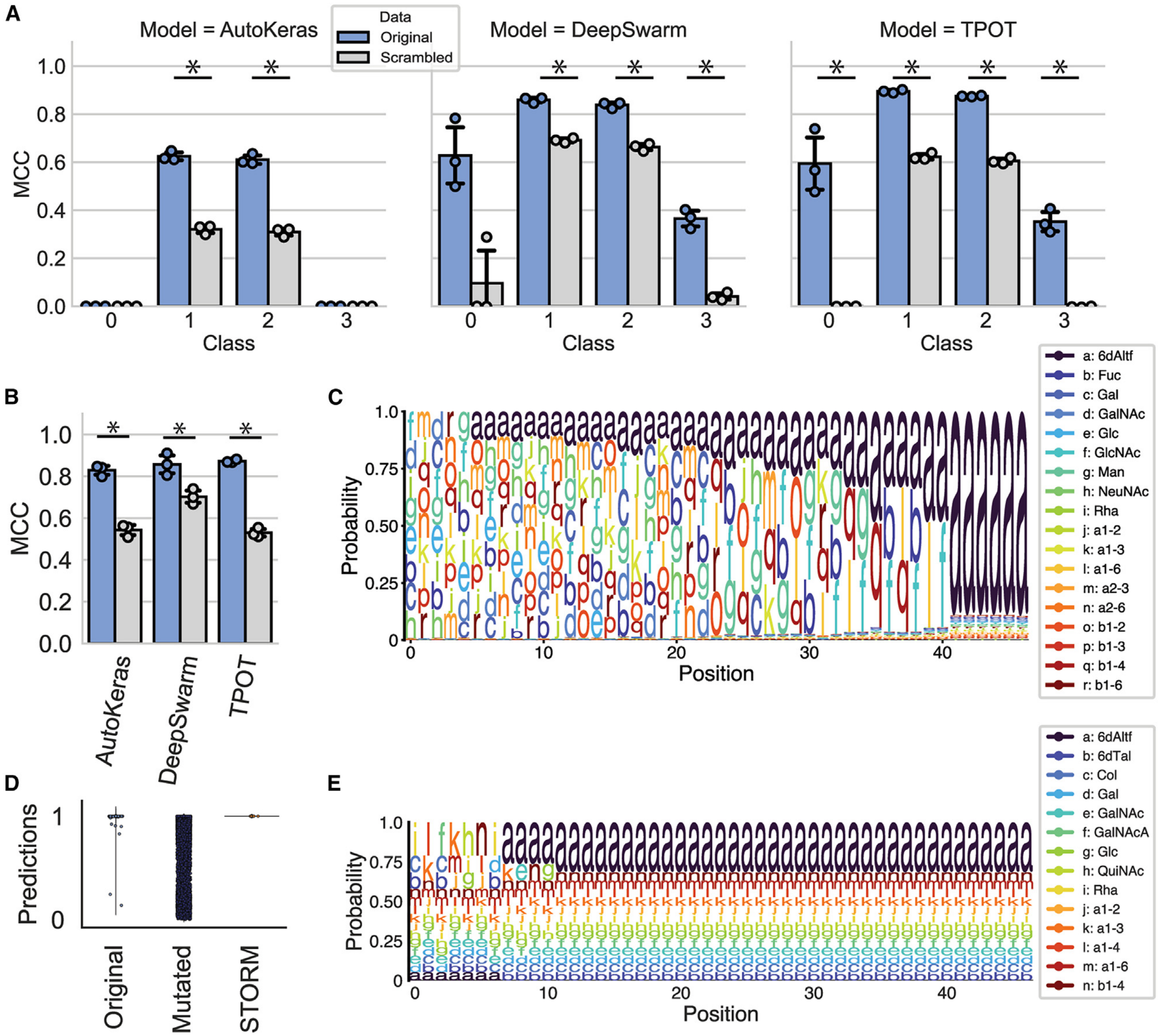


Figure 4. BioAutoMATED performs accurate binary and multi-class classification of glycan sequences

(A) Glycans were classified into archaea (N = 34), bacteria (N = 5,856), eukarya (N = 6,635), and virus (N = 149) domains, i.e., Classes 0, 1, 2, and 3, respectively. Multi-class models are significantly more predictive on original sequences compared with scrambled sequences for the largest two classes, Class 1 and Class 2.

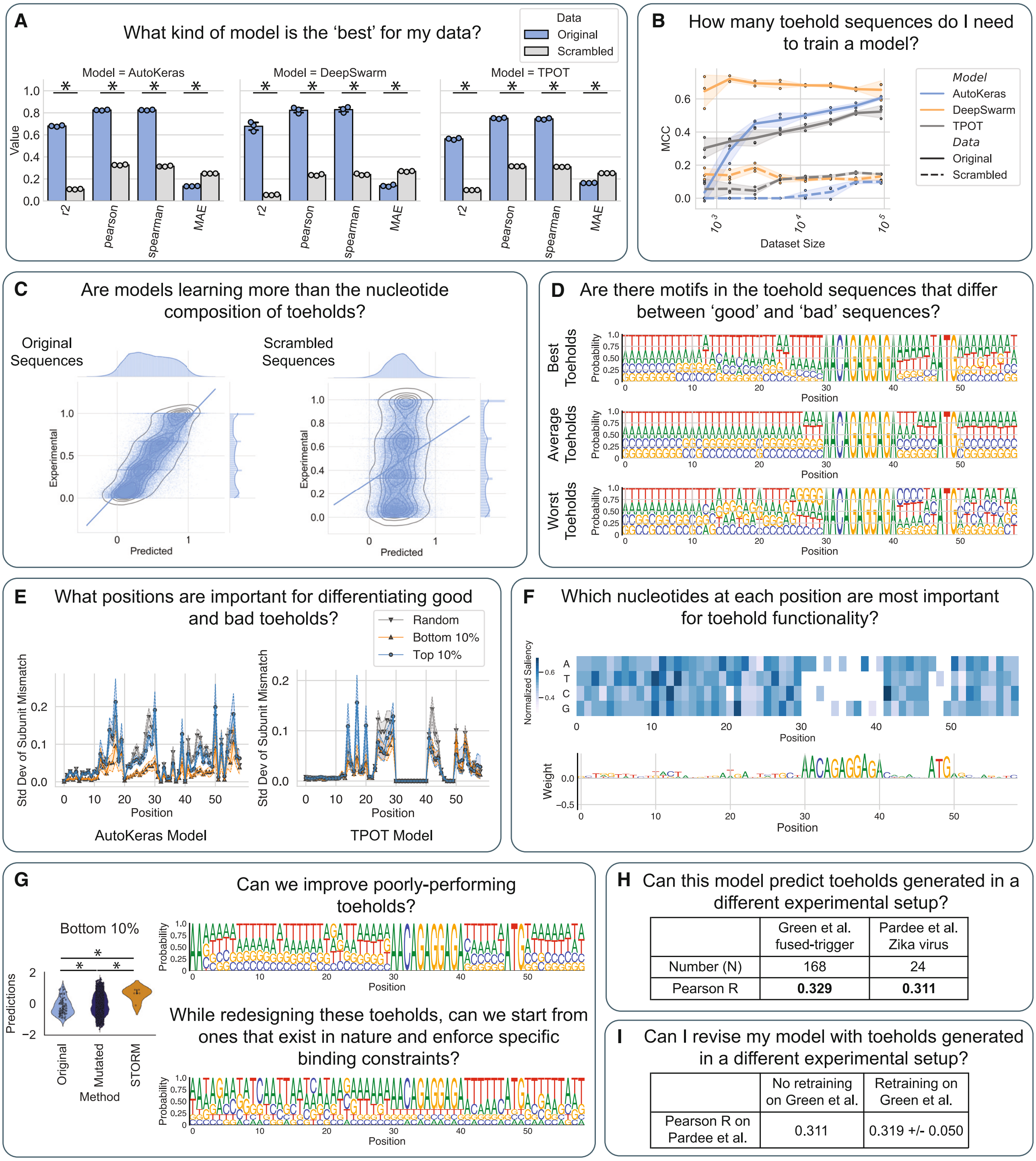
(B) Immunogenic glycan sequences were tested in the BioAutoMATED pipeline, achieving higher MCCs on the original sequences (blue) compared with models trained on scrambled controls (gray). For (A) and (B), points correspond to one of N = 3 cross-validation folds and asterisk denotes $p < 0.005$, two-sided t test.

(C) DeepSwarm models predicting the immunogenicity of glycans were evaluated with a sequence logo based on the class activation map for N = 100 sequences, with mannose (g characters) and rhamnose (i characters) showing some degree of importance to immunogenicity predictions. These results are consistent with sequence logos on the raw experimental data (Figure S5C), as well as with the rhamnose- and mannose-rich immunogenicity clusters reported in Bojar et al.[32]

(D) Glycan sequences can be designed via STORM with the top-performing DeepSwarm immunogenicity classifier, creating sequences that are designed to be as immunogenic (class 1) as possible in a proof-of-principle experiment (N = 100 starting sequences).

(E) These STORM-designed immunogenic glycan sequences display rhamnose enrichment, consistent with Figure S5C, and colitose enrichment, consistent with its characterization as a rare monosaccharide found in the LPS O-antigen of Gram-negative bacteria.[45]

data (Figure 5D) to confirm that known conserved regions—the RBS in positions 30–40 and the start codon in 47–49—were constant throughout all sequences. Furthermore, we found that the regions directly adjacent to each toehold's RBS varied the most between sequences with top 10% performance, sequences with bottom 10% performance, and a random subset of sequences. Additionally, the sequence logo of the top 10% of sequences accurately highlights an NUA motif (here TTA) in positions 21–23. This motif was previously described in Valeri, Collins, Ramesh et al.,[29] and is known to be a characteristic motif that influences toehold switch performance likely by regulating the binding within the hairpin. Feature importance plots again indicated that the models correctly identify the conserved regions in positions 30–40 and 47–49 (Figure 5E) and additionally



	Green et al. fused-trigger	Pardee et al. Zika virus
Number (N)	168	24
Pearson R	**0.329**	**0.311**

	No retraining on Green et al.	Retraining on Green et al.
Pearson R on Pardee et al.	0.311	0.319 +/- 0.050

Figure 5. BioAutoMATED can be used to address key biological questions about RNA toehold switches in an automated, end-to-end analysis

(A) To support a biologist exploring the use of ML on their dataset, BioAutoMATED automates the model search process and identifies a keras-based convolutional neural network model generated by AutoKeras as the most predictive. The built-in negative control of scrambled toehold sequences (gray) indicates that regression models trained on toehold switch sequences outperform models trained on scrambled toehold switch sequences.

(B) The BioAutoMATED pipeline tests the robustness of the ML pipelines to decreased dataset sizes, showing that DeepSwarm is saturated on the toehold switch dataset, whereas other methods converge to the highest performance value at the full provided dataset size. At the lowest dataset sizes, DeepSwarm achieves a significantly higher MCC value than both TPOT and AutoKeras ($p < 0.005$), whereas at the full dataset size, all models perform significantly better than their respective scrambled control models ($p < 0.005$). For (A) and (B), points correspond to one of N = 3 folds and asterisks indicate significance $p < 0.005$ with two-sided t test.

(legend continued on next page)

showed a steady increase in model attention between positions 20–30. In the saliency map sequence logos (Figure 5F), thymines appear somewhat enriched in the region of increased attention between positions 20–30, and we, indeed, found thymines to appear disproportionately in the best toehold switches' sequence logos (Figure 5D). These observations suggest that, in the hairpin (positions 20–30), weak A-T binding enhances toehold switch performance and might alter the dynamics of hairpin unwinding in the presence of the trigger sequence. Together, these results show that BioAutoMATED helps to identify regions, motifs, and individual nucleotides important for toehold switch performance.

To address our original goal of design, we used BioAutoMATED's design modules to build *de novo* designed toehold switches that maintain the hairpin binding constraint as well as the constant RBS and start codon sequences. When comparing both the mutagenesis approach and STORM, we found that STORM-designed toehold switch sequences have the highest predicted performance (Figure 5G). The STORM-designed sequences show more A-T binding in the regions of 20–30 and the corresponding descending hairpin binding region 41–46 in the designed sequences (bottom logo) than in the original poorly performing sequences (top logo). The STORM-designed sequences also incorporate greater variability in the first 10 nucleotides than the starting sequences (Figure 5G), a region which has minimal model importance (Figures 5E and 5F). The most-improved sequence starts from 'AAATCTTTGATGCTAGCAGTAGAACTGACCAACAGAGGAGAGGTCAGATGTACTGCTAG' which has a predicted score of -0.788; after design, this sequence becomes 'TAAAGAAGGTCAAATATAAATATCATAAAAAACAGAGGAGATTTTATATGATTTATATT' with a predicted score of 0.892, which does not exist in the original dataset. We further note that STORM preserves the user-specified constraints of exact sequences and base pairing: this is evident in the sequence logos, which show ACAGAGGAGA in positions 30–40 and the start codon in positions 47–49. Thus, as shown by these generated toehold switches, BioAutoMATED is able to produce toehold switch candidates *de novo* for subsequent experimental testing with minimal coding from the user.

In light of the challenge of translating between different experimental contexts, it is important to ask whether the models trained on the aforementioned dataset are generalizable. To address this, we used the trained models to predict the performance of toehold switches that were assayed in a different experimental context. We considered a set of 168 toehold switch sequences reported in Green et al.,[47] which were studied in a system without the trigger fused to the toehold switch. For this test set, BioAutoMATED's best-performing model achieved a Spearman R of 0.337 between the predicted and actual performance, comparable to a corresponding value of 0.36 reported in Valeri, Collins, Ramesh et al.[29] (Figure 5H; Table S2). In a different dataset of 24 Zika virus-detecting toehold switches reported in Pardee et al.[48] that were assayed using an additional amplification step, LacZ readout instead of fluorescence output, and different selection criteria, BioAutoMATED's best-performing model achieved a Spearman R of 0.307 and Pearson R of 0.311 between the predicted and actual performance (Figure 5H; Table S2). To explore whether experiment-specific training data could improve performance, we used BioAutoMATED's transfer learning capability (see STAR Methods) to re-train the models using the Green et al.[47] data and found a marginally higher Pearson R of 0.319 on Pardee et al.'s test set (Figure 5I). It is important to note that here the training and test sets are limited (N = 168 for re-training and N = 24 for testing). Nevertheless, these findings suggest that measuring toehold switch performance in one experimental context might have moderate generalizability for predicting toehold switch performance in other experimental contexts.

BioAutoMATED outperforms other AutoML tools

Given the broad applicability of BioAutoMATED, we further aimed to benchmark it against other published AutoML tools using the same datasets (Table 1, Table S3). Several solutions tailored to biological data, such as iLearnPlus,[49] BioAutoML,[50] and JADBio,[51] address a wide variety of predictive tasks with minimal user intervention. These tools perform automated machine learning in diverse ways and require variable amounts of pre-processing for biological sequences.

(C) Sample outputs demonstrate high performance for predicting continuous values (regression) of toehold sequence performance (left), with high degree of match between predicted and actual values. The predicted values for the scrambled sequences (right) trend significantly worse with the actual experimental values.

(D) Automatically generated raw sequence logos from experimental data (N = 50 sequences each) show that key differentiating factors between toehold performance are the nucleotides in positions 26–30, with strong binding (G-C bonds) correlating with worse performance than weak binding (A-T bonds) in that region. The first 10 nucleotides have very few differences across the three categories of sequences.

(E) *In silico* mutagenesis analysis on the AutoKeras model (left) and TPOT model (right) trained on toehold switches validates that the conserved regions in positions 30–40 and 47–49 are appropriately ignored by the toehold switch model. N = 50 sequences were tested for each model and each performance category.

(F) A saliency map and its corresponding sequence logo display low attention to the conserved regions positions in 30–40 and 47–49 for N = 100 sequences. Normalized saliency represents arbitrary model attention units.

(G) STORM-designed toehold sequences (bottom logo) demonstrate high predicted responsiveness scores compared with the starting sequences (top logo), which were a random selection (N = 100) of the bottom 10% of sequences. Asterisks indicate significance $p < 0.005$ with two-sided t test. Toehold switch sequences designed with STORM show the enforcement of constraints around the ribosome-binding site (positions 30–40) and start codon (positions 47–49), as well as hairpin binding enforced in these random sequences. Redesigned sequences have more A-T binding in the hairpin than the original starting sequences.

(H) The BioAutoMATED model is predictive on datasets from Green et al.[47] (N = 168) and Pardee et al.[48] (N = 24) despite those datasets being generated in different experimental contexts (e.g., the Zika virus toeholds had a LacZ readout, additional amplification step, and different selection criteria). Additional performance metrics can be found in Table S2.

(I) To improve generalizability of the models and exploit newly available data, the transfer learning module of BioAutoMATED can be used to re-train models on new datasets. Shown is the average of 25 trials using the Green et al.[47] dataset split into 90% training and 10% testing, with Pearson R shown as the performance metric for Pardee et al.'s dataset. Although the performance increase is marginal, it is likely that more re-training data could improve both model generalizability and predictiveness.

Table 1. BioAutoMATED incorporates more features and options than other available biologically focused AutoML tools

	Top model (or tied for top) in benchmarks	Takes in raw sequences as input	Performs regression	Interpretation functionality	Designs *de novo* sequences
JADBio	20% (36%)	no	yes	computes feature importances; paywall for dimensionality reduction	no
BioAutoML	4% (8%)	yes	no	computes feature importances	no
iLearnPlus	0% (8%)	requires hands-on pre-processing	no	computes feature importances, clustering, & dimensionality reduction	no
BioAutoMATED	56% (76%)	yes	yes	computes feature importances, attention maps, & in silico mutagenesis	yes

We evaluated performance of models from four tools (JADBio, BioAutoML, iLearnPlus, and our tool, BioAutoMATED) across 25 tasks, consisting of five performance measurements for five datasets each. The five datasets assessed comprise the nucleic acid and peptide datasets benchmarked here—RBS, peptide, toehold, 1,000 synthetic nucleic acids, and 100,000 synthetic nucleic acids—and the five performance evaluations consisted of binary classification auROC, binary classification MCC, regression R^2, regression MAE, and regression correlation coefficient. Overall, BioAutoMATED produced the best model or tied for the best model in 76% of these tasks. We note here that regression performance could not be compared across tools, as iLearnPlus and BioAutoML do not accommodate regression tasks. See Table S3 for comprehensive performance metrics across all four tools.

In brief, BioAutoML[50] assesses many diverse representations of DNA, RNA, and peptides. The optimal representation is used to vectorize sequences, or convert into a list of numbers, before optimizing models trained on input data. Similarly, iLearnPlus[49] provides both a website and a downloadable user interface in which the user can upload their dataset, select a method to vectorize DNA, RNA, or peptides, and choose a model type that is then trained and automatically evaluated. JADBio[51] is a website that allows for data analysis and model training, only accepts vector representations, and cannot handle raw biological sequences.

To aid in objectively benchmarking these different AutoML tools, we generated a synthetic control dataset (Figure S6). This dataset consists of randomly generated 20-nucleotide sequences (N = 100,000) that have target values according to a simple sum of their nucleotides, with an arbitrary scoring scheme as follows: add 1 for A, 2 for T, 3 for C, and 4 for G. These sequences have the same target value no matter the order of their nucleotides; therefore, models trained on original and scrambled sequences should perform equivalently, and, indeed, models trained on this synthetic control perform equally as well as models trained on the scrambled control (Figure S6A). BioAutoMATED models achieve perfect classification accuracies for binary classification, multi-class classification (Figure S6B), and regression (Figure S6C), providing a positive control for the BioAutoMATED platform.

The synthetic control benchmarks BioAutoMATED's interpretation functionality. As a negative control, an activation map on the synthetic nucleic acid control identifies no motifs but does ascribe the most importance to adenine nucleotides (Figures S6D and S6E), the nucleotide that adds the lowest score of 1 to the arbitrarily scored nucleotide sum target value. No biological meaning is artificially uncovered in this negative control attention map.

After suitable dataset pre-processing, we applied iLearnPlus,[49] BioAutoML,[50] and JADBio[51] to five benchmark datasets: RBS,[30] peptide,[31] and toehold switch sequences,[29] as well as a small set of synthetic nucleic acids (N = 1,000 sequences) and a larger set of synthetic nucleic acids (N = 100,000 sequences). We assessed each of the five datasets across two binary classification metrics (auROC and MCC) and three regression metrics (R^2, mean absolute error, and correlation coefficient), for a total of 25 "tasks." For 76% of these tasks, BioAutoMATED produces the best-performing model or ties for the best-performing model, representing a clear advantage over the other tools. The best BioAutoMATED model was distributed across AutoKeras, DeepSwarm, and TPOT model types, indicating that the strong performance of BioAutoMATED comes from integrating multiple architecture search methods and there is not one single "best" AutoML technique or model type. We also explored feature capabilities across all three tools, finding that BioAutoMATED accommodates more model interpretation methods than the other tools and is the only platform that incorporates a module for sequence design. More details on the comparison between BioAutoMATED, iLearnPlus, BioAutoML, and JADBio can be found in STAR Methods.

To further assess the robustness of BioAutoMATED, especially in light of new large language models that were published over the course of this study, we compared BioAutoMATED models with pre-trained language models. We re-trained two generic language models, DNABERT[52] for nucleic acids and ESM[53] for proteins, on the same benchmark datasets as above (see STAR Methods). For prediction of ribosome-binding sites, toeholds, and synthetic nucleic acids, DNABERT outperforms BioAutoMATED with a specific learning rate, or a parameter that influences the speed of model training (Table S4). However, when using the higher learning rate that is provided in the example code, DNABERT fails to train predictive models for any of the three tasks, with auROCs ~0.5. DNABERT's performance appears to depend heavily upon the model parameters selected, a barrier for easy adoption among non-ML practitioners. To evaluate language models on the peptide dataset, we used ESM,[53] a large-scale model that produces a lower-dimensional vector representation of each sequence called an embedding. ESM embeddings used as input to scikit-learn-based models exhibit substantial variations in performance depending on the model used (Table S5). Overall, the average R^2 value across the three models using ESM embeddings (0.538) was marginally higher than that of the best-performing BioAutoMATED model (0.425), indicating that ESM embeddings are a robust data representation method. These results suggest that large language models are a promising avenue for predictive model generation and that pretraining models on large datasets of nucleic acids and peptides can provide a strong performance boost. However, it is important to note that the current usage of both DNABERT and ESM requires architecture decisions and technical skills that BioAutoMATED abstracts away from, and automatically performs for, the user.

DISCUSSION

This work introduces and evaluates BioAutoMATED, a platform for integrating and deploying AutoML tools for research of biological sequences. Compared with other AutoML methods, BioAutoMATED offers several unique features: (1) it enables sequence-specific data pre-processing and corrects for variation in sequence length, (2) it handles glycan sequences in addition to nucleotide and protein sequences, and (3) it enables sequence design through a gradient ascent-based directed design module. Additionally, BioAutoMATED accommodates a wide variety of tasks and provides a comprehensive interpretation module for interrogating models. BioAutoMATED balances minimal user input with multiple degrees of freedom relevant to the computational and biological domains, ranging from thresholds for binary classification to selecting activation map gradients for model interpretation. After basic user inputs and data file uploading, BioAutoMATED proceeds hands-free. When the architecture search concludes, the Python notebooks provided by the platform can empower users to continue to explore BioAutoMATED's functionalities without needing to interact with the underlying model-generation code.

BioAutoMATED provides practical advantages to its users and can help domain experts validate their data-driven insights and ask appropriate questions pertaining to the biological data at hand. As exemplified by the various biological applications presented in this work, typical BioAutoMATED workflows might include the following steps:

(1)First, BioAutoMATED readily accommodates input sequence data, which can include nucleic acids, peptides, and glycans. Based on these inputs, BioAutoMATED automatically generates scrambled controls, which can help users answer the question of whether subunit frequency alone predicts performance or whether ML-driven analyses of sequence positions would be useful.

(2)Second, BioAutoMATED automatically tests if the input dataset is large enough to implement models with better predictive capabilities than that of randomized controls, which aids in providing proper comparisons for its users and can inform users when more data are needed.

(3)Third, BioAutoMATED automatically trains and evaluates a wide range of model architectures. The best-performing model can be used to make additional predictions and generate biological insight.

(4)Lastly, BioAutoMATED automatically generates data feature importance plots and attention maps, which readily provide information about which sequence regions of highest potential interest to further investigate (e.g., highlighted sequence logos from raw data that point to information-dense motifs). Indeed, such empirical insights exemplify how AutoML techniques can assist biologists in producing domain-specific knowledge that could lead to subsequent questions, hypotheses, models, and experiments.

The potential for BioAutoMATED to identify, train, and deploy predictive models for systems and synthetic biology researchers with minimal ML experience could help democratize the intersection of biology and computer science. State-of-the-art performance in many ML models has often been achieved through manual tailoring of architectures and associated parameters. However, automating the search for architectures and hyperparameters offers numerous benefits: for instance, the time and human resources required to implement ML models can be drastically reduced. As shown by the biological applications considered here, BioAutoMATED achieves competitive performance when benchmarked to published, manually tuned models, and BioAutoMATED also outperforms similar AutoML tools. Additionally, BioAutoMATED can improve reproducibility by facilitating objective comparisons of different analysis methods[54,55] – with BioAutoMATED, simple models to act as a baseline for new algorithms can be easily optimized and tuned for a fair comparison.

Using four different datasets from gene regulation, antibody-drug binding, glycan immunogenicity and classification, and toehold switch design, respectively, we have shown how BioAutoMATED can automatically generate ML models to predict various forms of biological activity, as well as offer interpretability and design features that can guide experiments in each of these areas. Specifically, BioAutoMATED readily generated high-performing binary classification and regression models that predict translation efficiency from RBS sequence alone, revealing that an adenine- and guanine-rich motif around RBS positions 5–8 is a salient pattern in top-performing sequences and producing *de novo* designed sequences that have high predicted translation efficiency using STORM. BioAutoMATED also identified accurate models for the prediction of ranibizumab binding to human IgG antibodies and revealed characteristics—the importance of an FDY motif, as well as the lack of importance of methionine, cysteine, and asparagine residues—relevant to antibody design. For the diverse space of glycans, BioAutoMATED generated models that accurately classify phylogenetic domains, as well as models that accurately predict immunogenicity to humans. Here, BioAutoMATED revealed that mannose, rhamnose, colitose, and fucose are particularly important to immunogenicity. Finally, BioAutoMATED identified high-performing models that predict toehold switch activity from sequence alone but showed that the models were only moderately generalizable to toehold switches in other experimental contexts. BioAutoMATED also revealed known conserved regions and motifs important for toehold switch activity, and that A-T binding between positions 20–30 is particularly meaningful. BioAutoMATED's STORM-designed sequences accommodate user-specified constraints and can be readily tested to validate what the models have learned.

Despite its broad applicability, BioAutoMATED, and AutoML as a whole, have limitations. Such methods cannot wholly replace "human-in-the-loop" ML usage. BioAutoMATED and other AutoML platforms should not replace critical, scientific evaluation: the models and predictions that they provide need to be carefully considered. For instance, as suggested by our study of toehold switches, BioAutoMATED models that achieve high performance on validation sets may not be accurate when applied to new experimental contexts. When possible, model predictions should be experimentally validated, and testing with external validation datasets can help prevent model overfitting and vet models that are less predictive than they seem. As such, BioAutoMATED models should represent helpful modules—and not endpoints—for any scientific workflow: especially in applications in which model performance is modest, the models should be used as a starting point for architecture and hyperparameter choices, helping scientific teams merge computationally obtained results with better-informed experiments.[56]

Due to the generality of its core functions, BioAutoMATED is widely adaptable. Moving forward, it will be useful to integrate BioAutoMATED with an ever-growing set of AutoML tools, such as those that focus on natural language processing (NLP)-based architectures,[57] residual networks (ResNets),[42] or ensemble models. Likewise, methods that focus on uncertainty modeling could foster more robust and data-efficient generalization. Future adaptations of BioAutoMATED could extend the platform beyond biological sequences and allow any sequence-like object (e.g., vector encodings or fingerprints) as inputs, in addition to more flexibly allowing interconversion between nucleic acid and peptide sequences by integrating tools such as the SeqLike library [https://github.com/modernatx/seqlike?ref=pythonrepo.com].

Given its ease of use, speed, and performance, we anticipate that BioAutoMATED will enable life scientists to readily develop and utilize ML models, helping to drive biological research forward.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - ○ Lead contact
  - ○ Materials availability
  - ○ Data and code availability
- METHODS DETAILS
  - ○ Installation and usage of BioAutoMATED
  - ○ User inputs
  - ○ Pre-processing and data cleaning
  - ○ Architecture search
  - ○ Class activation and saliency maps
  - ○ Naive design module
  - ○ Directed design module
  - ○ Transfer learning functionality
  - ○ Proof of concept with experimental data
  - ○ Scope of the platform
  - ○ Machine specifications and package environments
  - ○ Benchmarking against automated ML tools
  - ○ Benchmarking against pre-trained language models
- QUANTIFICATION AND STATISTICAL ANALYSIS

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.cels.2023.05.007.

ACKNOWLEDGMENTS

We thank Daniel Bojar for his input on glycan sequence analysis. We also thank Timothy Kassis for the helpful discussions and advice relating to the implementation of deep-learning model architectures. We thank Max Atti English and Miguel Alcantar for their helpful discussions on benchmarking datasets and synthetic biology. We also thank Randal S. Olson for maintaining TPOT, Haifeng Jin and François Chollet for maintaining AutoKeras, and Edvinas Byla for maintaining and giving support in the use of DeepSwarm API.

This work was supported by Defense Threat Reduction Agency grant HDTRA-12210032, the DARPA SD2 program, the Paul G. Allen Frontiers Group, and the Wyss Institute for Biologically Inspired Engineering, Harvard University (J.A.V., L.R.S., N.A.M., J.J.C.). This work is part of the Antibiotics-AI Project, which is directed by J.J.C. and supported by the Audacious Project, Flu Lab, LLC, the Sea Grape Foundation, Rosamund Zander and Hansjorg Wyss for the Wyss Foundation, and an anonymous donor. J.A.V. was also supported by an MIT-Takeda Fellowship and Siebel Foundation Scholarship. L.R.S. was also supported by CONACyT grant 342369 / 408970, while N.A.M. was also supported by an MIT Tata Center fellowship 2748460. K.M.C. was supported as a Johnson & Johnson Undergraduate Research Scholar and with a Barry Goldwater Scholarship and acknowledges current funding from the Marshall Scholarship and Cambridge Trust. F.W. was supported by the National Institute of Allergy and Infectious Diseases of the National Institutes of Health under award number K25AI168451.

AUTHOR CONTRIBUTIONS

J.A.V., L.R.S., K.M.C., P.R., and N.A.M. conceived the study. J.A.V., L.R.S., and K.M.C. integrated AutoKeras, DeepSwarm, and TPOT into the AutoML code. J.A.V. extended the library for data robustness, model interpretation, and sequence design. G.C. tested sequence design functions and wrote the sequence mutagenesis module. R.P. tested all other functions. J.A.V. and P.R. conceived and wrote the data pre-processing functions. J.A.V., L.R.S., and K.M.C. performed the AutoML analysis and designed figures. F.W. assisted with data interpretation. All authors wrote and edited the manuscript. J.J.C., T.K.L., and D.M.C. supervised the research.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: April 30, 2022
Revised: February 17, 2023
Accepted: May 22, 2023
Published: June 21, 2023

REFERENCES

1. Camacho, D.M., Collins, K.M., Powers, R.K., Costello, J.C., and Collins, J.J. (2018). Next-generation machine learning for biological networks. Cell *173*, 1581–1592. https://doi.org/10.1016/j.cell.2018.05.015.

2. Ching, T., Himmelstein, D.S., Beaulieu-Jones, B.K., Kalinin, A.A., Do, B.T., Way, G.P., Ferrero, E., Agapow, P.M., Zietz, M., Hoffman, M.M., et al. (2018). Opportunities and obstacles for deep learning in biology and medicine. J. R. Soc. Interface *15*, 20170387. https://doi.org/10.1098/rsif.2017.0387.

3. Carbonell, P., Radivojevic, T., and García Martín, H. (2019). Opportunities at the intersection of synthetic biology, machine learning, and automation. ACS Synth. Biol. *8*, 1474–1477. https://doi.org/10.1021/acssynbio.8b00540.

4. Yang, K.K., Wu, Z., and Arnold, F.H. (2019). Machine-learning-guided directed evolution for protein engineering. Nat. Methods *16*, 687–694. https://doi.org/10.1038/s41592-019-0496-6.

5. Chen, K.M., Cofer, E.M., Zhou, J., and Troyanskaya, O.G. (2019). Selene: a PyTorch-based deep learning library for sequence data. Nat. Methods *16*, 315–318. https://doi.org/10.1038/s41592-019-0360-8.

6. Avsec, Ž., Kreuzhuber, R., Israeli, J., Xu, N., Cheng, J., Shrikumar, A., Banerjee, A., Kim, D.S., Beier, T., Urban, L., et al. (2019). The Kipoi

repository accelerates community exchange and reuse of predictive models for genomics. Nat. Biotechnol. *37*, 592–600. https://doi.org/10.1038/s41587-019-0140-0.

7. Liu, B. (2019). BioSeq-Analysis: a platform for DNA, RNA and protein sequence analysis based on machine learning approaches. Brief. Bioinform. *20*, 1280–1294. https://doi.org/10.1093/bib/bbx165.
8. Rawat, W., and Wang, Z. (2017). Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput. *29*, 2352–2449. https://doi.org/10.1162/NECO_a_00990.
9. Zoph, B., Vasudevan, V., Shlens, J., and Le, Q.V. (2018). Learning transferable architectures for scalable image recognition. arXiv. https://doi.org/10.48550/arXiv.1707.07012.
10. Feurer, M., and Hutter, F. (2019). Hyperparameter optimization. In Automated Machine Learning: Methods, Systems, Challenges the Springer Series on Challenges in Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, eds. (Springer International Publishing), pp. 3–33. https://doi.org/10.1007/978-3-030-05318-5_1.
11. Pfisterer, F., Thomas, J., and Bischl, B. (2019). Towards human centered AutoML. arXiv. https://doi.org/10.48550/arXiv.1911.02391.
12. Liang, J., Meyerson, E., Hodjat, B., Fink, D., Mutch, K., and Miikkulainen, R. (2019). Evolutionary neural AutoML for deep learning. Proceedings of the Genetic and Evolutionary Computation Conference GECCO ’19 (Association for Computing Machinery), pp. 401–409. https://doi.org/10.1145/3321707.3321721.
13. Faes, L., Wagner, S.K., Fu, D.J., Liu, X., Korot, E., Ledsam, J.R., Back, T., Chopra, R., Pontikos, N., Kern, C., et al. (2019). Automated deep learning design for medical image classification by health-care professionals with no coding experience: a feasibility study. Lancet Digit. Health *1*, e232–e242. https://doi.org/10.1016/S2589-7500(19)30108-6.
14. He, X., Zhao, K., and Chu, X. (2021). AutoML: a survey of the state-of-the-art. Knowl. Based Syst. *212*, 106622. https://doi.org/10.1016/j.knosys.2020.106622.
15. Elshawi, R., Maher, M., and Sakr, S. (2019). Automated machine learning: state-of-the-art and open challenges. arXiv. https://doi.org/10.48550/arXiv.1906.02287.
16. Zoph, B., and Le, Q.V. (2017). Neural architecture search with reinforcement learning. arXiv. https://doi.org/10.48550/arXiv.1611.01578.
17. Mendoza, H., Klein, A., Feurer, M., Springenberg, J.T., and Hutter, F. (2016). Towards automatically-tuned neural networks. Proceedings of the Workshop on Automatic Machine Learning *64*, 58–65.
18. Cai, H., Chen, T., Zhang, W., Yu, Y., and Wang, J. (2017). Efficient architecture search by network transformation. arXiv. https://doi.org/10.48550/arXiv.1707.04873.
19. Elsken, T., Metzen, J.H., and Hutter, F. (2019). Neural architecture search: a survey. arXiv. https://doi.org/10.48550/arXiv.1808.05377.
20. Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., and Hutter, F. (2021). Auto-sklearn 2.0: hands-free AutoML via meta-learning. arXiv. https://doi.org/10.48550/arXiv.2007.04074.
21. Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., and Leyton-Brown, K. (2019). Auto-WEKA: automatic model selection and hyperparameter optimization in Weka. In Automated Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, eds. (Springer International Publishing), pp. 81–95. https://doi.org/10.1007/978-3-030-05318-5_4.
22. Alaa, A., and Schaar, M. (2018). AutoPrognosis: automated clinical prognostic modeling via bayesian optimization with structured kernel learning. arXiv. https://doi.org/10.48550/arXiv.1802.07207.
23. Olson, R.S., and Moore, J.H. (2019). TPOT: A tree-based pipeline optimization tool for automating machine learning. In Automated Machine Learning: Methods, Systems, Challenges the Springer Series on Challenges in Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, eds. (Springer International Publishing), pp. 151–160. https://doi.org/10.1007/978-3-030-05318-5_8.
24. de Sá, A.G.C., Pinto, W.J.G.S., Oliveira, L.O.V.B., and Pappa, G.L. (2017). RECIPE: A grammar-based framework for automatically evolving classification pipelines. In Genetic Programming Lecture Notes in Computer Science, J. McDermott, M. Castelli, L. Sekanina, E. Haasdijk, and P. García-Sánchez, eds. (Springer International Publishing), pp. 246–261. https://doi.org/10.1007/978-3-319-55696-3_16.
25. A Romero, R.A., Y Deypalan, M.N., Mehrotra, S., Jungao, J.T., Sheils, N.E., Manduchi, E., and Moore, J.H. (2022). Benchmarking AutoML frameworks for disease prediction using medical claims. BioData Min. *15*, 15. https://doi.org/10.1186/s13040-022-00300-2.
26. Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: an efficient neural architecture search system. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining KDD ’19 (Association for Computing Machinery), pp. 1946–1956. https://doi.org/10.1145/3292500.3330648.
27. Byla, E., and Pang, W. (2019). DeepSwarm: optimising convolutional neural networks using swarm intelligence. arXiv. https://doi.org/10.48550/arXiv.1905.07350.
28. Bogard, N., Linder, J., Rosenberg, A.B., and Seelig, G. (2019). A deep neural network for predicting and engineering alternative polyadenylation. Cell *178*, 91–106.e23. https://doi.org/10.1016/j.cell.2019.04.046.
29. Valeri, J.A., Collins, K.M., Ramesh, P., Alcantar, M.A., Lepe, B.A., Lu, T.K., and Camacho, D.M. (2020). Sequence-to-function deep learning frameworks for engineered riboregulators. Nat. Commun. *11*, 5058. https://doi.org/10.1038/s41467-020-18676-2.
30. Höllerer, S., Papaxanthos, L., Gumpinger, A.C., Fischer, K., Beisel, C., Borgwardt, K., Benenson, Y., and Jeschek, M. (2020). Large-scale DNA-based phenotypic recording and deep learning enable highly accurate sequence-function mapping. Nat. Commun. *11*, 3551. https://doi.org/10.1038/s41467-020-17222-4.
31. Liu, G., Zeng, H., Mueller, J., Carter, B., Wang, Z., Schilz, J., Horny, G., Birnbaum, M.E., Ewert, S., and Gifford, D.K. (2020). Antibody complementarity determining region design using high-capacity machine learning. Bioinformatics *36*, 2126–2133. https://doi.org/10.1093/bioinformatics/btz895.
32. Bojar, D., Powers, R.K., Camacho, D.M., and Collins, J.J. (2021). Deep-learning resources for studying glycan-mediated host-microbe interactions. Cell Host Microbe *29*, 132–144.e3. https://doi.org/10.1016/j.chom.2020.10.004.
33. Angenent-Mari, N.M., Garruss, A.S., Soenksen, L.R., Church, G., and Collins, J.J. (2020). A deep learning approach to programmable RNA switches. Nat. Commun. *11*, 5057. https://doi.org/10.1038/s41467-020-18677-1.
34. Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C.B., and Farivar, R. (2019). Towards automated machine learning: evaluation and comparison of AutoML approaches and tools. 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI) (IEEE Publications), pp. 1471–1479. https://doi.org/10.1109/ICTAI.2019.00209.
35. Olson, R.S., Bartley, N., Urbanowicz, R.J., and Moore, J.H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO ’16 (Association for Computing Machinery), pp. 485–492. https://doi.org/10.1145/2908812.2908918.
36. Perez, L., and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv. https://doi.org/10.48550/arXiv.1712.04621.
37. Chen, V., Li, J., Kim, J.S., Plumb, G., and Talwalkar, A. (2021). Interpretable machine learning: moving from mythos to diagnostics. arXiv. https://doi.org/10.48550/arXiv.2103.06254.
38. Lopez, R., Gayoso, A., and Yosef, N. (2020). Enhancing scientific discoveries in molecular biology with deep generative models. Mol. Syst. Biol. *16*, e9198. https://doi.org/10.15252/msb.20199198.
39. Linder, J., Bogard, N., Rosenberg, A.B., and Seelig, G. (2020). A generative neural network for maximizing fitness and diversity of synthetic DNA and protein sequences. Cell Syst. *11*, 49–62.e16. https://doi.org/10.1016/j.cels.2020.05.007.

40. Repecka, D., Jauniskis, V., Karpus, L., Rembeza, E., Rokaitis, I., Zrimec, J., Poviloniene, S., Laurynenas, A., Viknander, S., Abuajwa, W., et al. (2021). Expanding functional protein sequence spaces using generative adversarial networks. Nat. Mach. Intell. *3*, 324–333. https://doi.org/10.1038/s42256-021-00310-5.

41. Wan, F., Kontogiorgos-Heintz, D., and de la Fuente-Nunez, C. (2022). Deep generative models for peptide design. Digit. Discov. *1*, 195–208. https://doi.org/10.1039/D1DD00024A.

42. He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv. https://doi.org/10.48550/arXiv.1512.03385.

43. Dalziel, M., Crispin, M., Scanlan, C.N., Zitzmann, N., and Dwek, R.A. (2014). Emerging principles for the therapeutic exploitation of glycosylation. Science *343*, 1235681. https://doi.org/10.1126/science.1235681.

44. Mohapatra, S., An, J., and Gómez-Bombarelli, R. (2021). GLAMOUR: graph learning over macromolecule representations. arXiv. https://doi.org/10.48550/arXiv.2103.02565.

45. Alam, J., Beyer, N., and Liu, H.W. (2004). Biosynthesis of colitose: expression, purification, and mechanistic characterization of GDP-4-keto-6-deoxy-d-mannose-3-dehydrase (ColD) and GDP-l-colitose synthase (ColC). Biochemistry *43*, 16450–16460. https://doi.org/10.1021/bi0483763.

46. Planinc, A., Bones, J., Dejaegher, B., Van Antwerpen, P., and Delporte, C. (2016). Glycan characterization of biopharmaceuticals: updates and perspectives. Anal. Chim. Acta *921*, 13–27. https://doi.org/10.1016/j.aca.2016.03.049.

47. Green, A.A., Silver, P.A., Collins, J.J., and Yin, P. (2014). Toehold switches: de-novo-designed regulators of gene expression. Cell *159*, 925–939. https://doi.org/10.1016/j.cell.2014.10.002.

48. Pardee, K., Green, A.A., Takahashi, M.K., Braff, D., Lambert, G., Lee, J.W., Ferrante, T., Ma, D., Donghia, N., Fan, M., et al. (2016). Rapid, low-cost detection of Zika virus using programmable biomolecular components. Cell *165*, 1255–1266. https://doi.org/10.1016/j.cell.2016.04.059.

49. Chen, Z., Zhao, P., Li, C., Li, F., Xiang, D., Chen, Y., Akutsu, T., Daly, R., Webb, G., Zhao, Q., et al. (2021). iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization. Nucleic Acids Res. *49*, e60. https://doi.org/10.1093/nar/gkab122.

50. Bonidia, R.P., Santos, A.P.A., de Almeida, B.L.S., Stadler, P.F., da Rocha, U.N., Sanches, D.S., and de Carvalho, A.C.P.L.F. (2022). BioAutoML: automated feature engineering and metalearning to predict noncoding RNAs in bacteria. Brief. Bioinform. *23*, bbac218. https://doi.org/10.1093/bib/bbac218.

51. Tsamardinos, I., Charonyktakis, P., Papoutsoglou, G., Borboudakis, G., Lakiotaki, K., Zenklusen, J.C., Juhl, H., Chatzaki, E., and Lagani, V. (2022). Just Add Data: automated predictive modeling for knowledge discovery and feature selection. npj Precis. Oncol. *6*, 38. https://doi.org/10.1101/2020.05.04.075747.

52. Ji, Y., Zhou, Z., Liu, H., and Davuluri, R.V. (2021). DNABERT: pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. Bioinformatics *37*, 2112–2120. https://doi.org/10.1093/bioinformatics/btab083.

53. Meier, J., Rao, R., Verkuil, R., Liu, J., Sercu, T., and Rives, A. (2021). Language models enable zero-shot prediction of the effects of mutations on protein function. bioRxiv, 29287–29303. https://doi.org/10.1101/2021.07.09.450648.

54. Bergstra, J., Yamins, D., and Cox, D. (2013). Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. Proceedings of the 30th International Conference on Machine Learning (PMLR), pp. 115–123.

55. Sculley, D., Snoek, J., Rahimi, A., and Wiltschko, A. (2018). Winner's curse? On pace, progress, and empirical rigor. International Conference on Learning Representations, ICLR, 1–4.

56. Seeber, I., Bittner, E., Briggs, R.O., de Vreede, T., de Vreede, G.-J., Elkins, A., Maier, R., Merz, A.B., Oeste-Reiß, S., Randrup, N., et al. (2020). Machines as teammates: a research agenda on AI in team collaboration. Inf. Manag. *57*, 103174. https://doi.org/10.1016/j.im.2019.103174.

57. Li, H.L., Pang, Y.H., and Liu, B. (2021). BioSeq-BLM: a platform for analyzing DNA, RNA and protein sequences based on biological language models. Nucleic Acids Res. *49*, e129. https://doi.org/10.1093/nar/gkab829.

58. Torrey, L., and Shavlik, J. (2009). Transfer learning. In Handbook of Research on Machine Learning Applications, E. Soria, J. Martin, R. Magdalena, M. Martinez, and A. Serrano, eds. (IGI Global), pp. 1–22.

59. McKinney, W. (2010). Data structures for statistical computing in python. Proceedings of the of the 9th Python in Science conference (SCIPY 2010), pp. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a.

60. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al. (2020). Array programming with NumPy. Nature *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2.

61. Budach, S., and Marsico, A. (2018). pysster: classification of biological sequences by learning sequence and structure motifs with convolutional neural networks. Bioinformatics *34*, 3035–3037. https://doi.org/10.1101/230086.

62. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2016). TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv. https://doi.org/10.48550/arXiv.1603.04467.

63. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: an imperative style, high-performance deep learning library. arXiv. https://arxiv.org/abs/arXiv:1912.01703.

64. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: machine learning in python. J. Mach. Learn. Res. *12*. 2825–28.

65. Yao, J., and Shepperd, M. (2020). Assessing software defection prediction performance: why using the Matthews correlation coefficient matters. Proceedings of the Evaluation and Assessment in Software Engineering (Association for Computing Machinery), pp. 120–129. https://doi.org/10.1145/3383219.3383232.

66. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2.

67. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*2016* (IEEE), pp. 2921–2929. https://doi.org/10.1109/CVPR.2016.319.

68. GitHub (2017). raghakot/keras-vis: neural network visualization toolkit for keras. https://github.com/raghakot/keras-vis.

69. Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep Inside convolutional networks: visualising image classification models and saliency maps. arXiv. https://arxiv.org/abs/arXiv:1312.6034.

70. Tareen, A., and Kinney, J.B. (2020). Logomaker: beautiful sequence logos in python. Bioinformatics *36*, 2272–2274. https://doi.org/10.1093/bioinformatics/btz921.

71. Garruss, A.S., Collins, K.M., and Church, G.M. (2021). Deep representation learning improves prediction of LacI-mediated transcriptional repression. Proc. Natl. Acad. Sci. USA *118*. e2022838118. https://doi.org/10.1073/pnas.2022838118.

72. Lipton, Z.C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv. https://arxiv.org/abs/arXiv:1506.00019.

73. Bryant, D.H., Bashir, A., Sinai, S., Jain, N.K., Ogden, P.J., Riley, P.F., Church, G.M., Colwell, L.J., and Kelsic, E.D. (2021). Deep diversification

of an AAV capsid protein by machine learning. Nat. Biotechnol. *39*, 691–696. https://doi.org/10.1038/s41587-020-00793-4.

74. Dallago, C., Mou, J., Johnston, K.E., Wittmann, B.J., Bhattacharya, N., Goldman, S., Madani, A., and Yang, K.K. (2021). FLIP: benchmark tasks in fitness landscape inference for proteins. bioRxiv. https://doi.org/10.1101/2021.11.09.467890.

75. Sarkisyan, K.S., Bolotin, D.A., Meer, M.V., Usmanova, D.R., Mishin, A.S., Sharonov, G.V., Ivankov, D.N., Bozhanova, N.G., Baranov, M.S., Soylemez, O., et al. (2016). Local fitness landscape of the green fluorescent protein. Nature *533*, 397–401. https://doi.org/10.1038/nature17995.

76. Gelman, S., Fahlberg, S.A., Heinzelman, P., Romero, P.A., and Gitter, A. (2021). Neural networks to learn protein sequence–function relationships from deep mutational scanning data. Proc. Natl. Acad. Sci. USA *118*. e2104878118. https://doi.org/10.1073/pnas.2104878118.

77. Xu, Y., Verma, D., Sheridan, R.P., Liaw, A., Ma, J., Marshall, N.M., McIntosh, J., Sherer, E.C., Svetnik, V., and Johnston, J.M. (2020). Deep dive into machine learning models for protein engineering. J. Chem. Inf. Model. *60*, 2773–2790. https://doi.org/10.1021/acs.jcim.0c00073.

78. Zhang, C., Shine, M., Pyle, A.M., and Zhang, Y. (2022). US-align: universal structure alignments of proteins, nucleic acids, and macromolecular complexes. Nat. Methods *19*, 1109–1115. https://doi.org/10.1038/s41592-022-01585-1.

79. Zhang, Z., Zhou, L., Gou, L., and Wu, Y.N. (2019). Neural architecture search for joint optimization of predictive power and biological knowledge. arXiv. https://arxiv.org/abs/arXiv:1909.00337.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Ribosome binding site sequences	Höllerer et al.[30]	https://github.com/BorgwardtLab/SAPIENs/tree/main/data
Peptide sequences	Liu et al., 2020[31]	https://github.com/gifford-lab/antibody-2019/tree/master/data/training%20data
Glycan sequences	Bojar et al., 2021[32]	https://doi.org/10.1016/j.chom.2020.10.004
Toehold switch sequences	Valeri, Collins, Ramesh, et al., [29] Angenent-Mari, et al.[33]	https://github.com/midas-wyss/engineered-riboregulator-ML/blob/master/data/newQC_toehold_data.csv
Software and algorithms		
DeepSwarm	Byla et al., 2019	https://github.com/Pattio/DeepSwarm
TPOT	Olsen et al., 2019	https://github.com/EpistasisLab/tpot
AutoKeras	Jin et al.[26]	https://autokeras.com
JADBio	Tsamardinos et al.[51]	https://jadbio.com
BioAutoML	Bonidia et al.[50]	https://github.com/Bonidia/BioAutoML/
iLearnPlus	Chen et al. [37]	https://github.com/Superzchen/iLearnPlus
DNABERT	Ji et al.[52]	https://github.com/jerryji1993/DNABERT
ESM	Meier et al.[53]	Zenodo: https://doi.org/10.5281/zenodo.7566741
BioAutoMATED	This paper	Zenodo: https://doi.org/10.5281/zenodo.7842505

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, James J. Collins (jimjc@mit.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

This paper analyzes existing, publicly available data. The accession numbers for these datasets are listed in the key resources table. All data needed to reproduce the figures and evaluate the conclusions in the paper can be found in the paper itself, the DockerHub repository [https://hub.docker.com/repository/docker/jackievaleri/bioautomated/general], and/or the GitHub repository [https://github.com/jackievaleri/BioAutoMATED].

All original code has been deposited at the GitHub repository [https://github.com/jackievaleri/BioAutoMATED] and is publicly available as of the date of publication. DOIs are listed in the key resources table. Code used to reproduce the results presented in this manuscript is available in the DockerHub and GitHub repositories. Instructions for package installations and using Docker and GitHub are available at the GitHub repository.

Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

METHODS DETAILS

Installation and usage of BioAutoMATED

To facilitate the broad use of BioAutoMATED, we have made our code available as a GitHub repository and DockerHub repository. All modules are accessible with easy-to-modify Jupyter notebook code, which can be used for any new BioAutoMATED analysis. We also provide all code and notebooks used to parse results from folders and generate the applicable figures for all plots shown here. If users decide to install directly from the GitHub repository, they must download and install packages according to the detailed instructions provided in the GitHub repository. The DockerHub installation automatically handles all packages and dependencies required for running BioAutoMATED. With just two lines of coding after installing Docker, any Mac, Linux, or Windows user can run the provided Jupyter Notebooks and reproduce all figures and analyses in this manuscript.

We strive to support the use and re-use of models generated by BioAutoMATED to improve generalizability, robustness, and reproducibility. We have provided example notebooks in which BioAutoMATED was used to create models that predict new sequences (such as those for externally supplied validation datasets). We have also provided example notebooks to perform interpretation and design modules using trained BioAutoMATED models. As biological data collection is oftentimes incremental, we have provided an example notebook to perform transfer learning; that is, re-training existing models on new data.[58]

User inputs

Several parameters can be optionally specified by the user to guide overall system operation. These parameters include maximum run time per AutoML framework in minutes (max_runtime_minutes, default = 60); number of required cross-validation folds (num_folds, default = 3); an option to produce text output (verbosity [0=Limited, 1=Complete], default = 0); which AutoML tools to use (automl_search_techniques [’all’, ’deepswarm’, ’autokeras’, ’tpot’], default = ‘all’); and generation of an output backup folder (do_backup, default = False). While the aforementioned parameters are optional, we require users to select the task (binary_classification, multiclass_classification, or regression), as well as target folders for models and results. These folders can be generic (e.g., ./peptides/models/), and tags will automatically be added to specify the task and the AutoML tool used. Other internal search parameters specific to AutoKeras,[26] DeepSwarm,[27] and TPOT,[23] such as population_size for TPOT and ant_count for DeepSwarm, can be manually selected by the user. However, these search parameters are not required, and BioAutoMATED will revert to the default parameters stated in the original packages if these search parameters are unspecified.

Another set of user-defined parameters dictate data pre-processing operations. The path and data file name must be specified. The user must indicate both the column name for the input column of sequences (input_col, default = ‘input’) and the column name for the target column, which is usually a column of experimentally derived values (target_col, default = ‘target’). The sequence type (one of [‘nucleic_acid’, ‘protein’, ‘glycan’]) must be manually defined and cannot be inferred from the alphabet composition due to the overlap of letters in nucleic acid and protein sequences. We emphasize that the above parameters are easy and quick to specify, and their specification does not require ML expertise from the user. For sequences of heterogeneous lengths, the ‘pad_seqs’ argument can be specified: either pad all sequences to the maximum length (‘max’), truncate all sequences to the minimum length (‘min’), or truncate and pad all sequences to the average length (‘average’). Lastly, the automatic class binning (do_auto_bin, default = True) will split data into positive and negative classes on the median of target values or a manually defined threshold (bin_threshold, must be supplied additionally).

A data augmentation protocol for nucleic acid sequences provides an option to “spike in” complementary sequences and/or reverse complementary sequences to the original sequences in the dataset, labeled with the original sequence’s label (augment_data, one of [‘complement’, ‘reverse_complement’, or ‘both_comp’]). We recommend using this option only if the user is confident that their nucleic acid sequences have similar functionality in reverse complement or complement strands (e.g., sequences with strong secondary structure determinants). The user can specify an automatic data robustness test to evaluate model saturation (data_robustness, default = False). The user can also customize both interpretation modules (run_interpretation, default = False) and design modules (run_design, default = False), covered below in the interpretation and design sections.

Pre-processing and data cleaning

The data pre-processing module reads in a data file, where any data file must be of a supported format (.csv, .xls, or .xslx), using pandas[59]; the data file is then processed using NumPy.[60] If the user provides a dataset containing entries other than nucleic acid, peptide, or glycan sequences, the user is alerted, and the BioAutoMATED run is aborted. For minor, fixable issues (e.g., an unrecognized character is found in a sequence), we report this issue, replace the offending character, and do not abort the BioAutoMATED run. Glycan sequences are processed into “glycoletters”, a set of monosaccharides and bonds described in Bojar et al.,[32] by splitting on parentheses, stripping and replacing all brackets, and appending the remaining glycoletters as a list. Sequences with non-standard characters are flagged: nucleic acids with characters other than A, T, C, and G; protein sequences with amino acids other than the 20 canonical amino acids, plus selenocysteine and pyrrolysine; and glycan sequences with glycoletters other than the list of 1,027 glycoletters compiled by Bojar et al.[32] For nucleic acid sequences, uracils are converted to thymines. Furthermore, a dictionary to convert non-standard letters into the standard IUPAC alphabet code is employed: for example, R to A or G, Y to C or T, S to G or C, etc. We emphasize that the actual sequence is not changed in this step, but all indicated letters are set to 1 in the one-hot encoded matrix (see one-hot encoding details below). For protein sequences, a similar protocol is followed for abbreviations J (isoleucine or leucine), Z (glutamine or glutamic acid), and B (arginine or asparagine). All sequences (excluding glycans) are converted to uppercase letters for standardization, and all dash, space, period, and asterisk characters are replaced with the relevant gap letter for that sequence type (N for nucleic acids, X for proteins and glycans). These gap letters are handled similarly to substitutions as described above; e.g., a position with an N in a nucleic acid sequence will have a 1 in every row in the one-hot encoded matrix signifying all nucleotides are possible.

Sequences with heterogenous lengths are padded according to the user-specified input (‘max’, ‘min’, or ‘average’, default = ‘max’). Data are augmented either with complement or reverse complement sequences (or both), wherein outputs are extended with analogous labels to the original sequences. Only unique sequences distinct from the original list of input sequences will be added to the augmented dataset. A list of scrambled sequences is generated by randomly shuffling every position for every sequence in the set. Finally, a trimmed alphabet (where the alphabet corresponds to ATCG for nucleic acid sequences, etc.) is produced such that the alphabet is only comprised of letters that appear in any of the sequences. This step can reduce the dimensionality of the glycan alphabets from the maximum 1,027 glycoletters to a more streamlined ∼400 glycoletters, improving both compute time and model performance. Both the original sequence list and the scrambled sequence lists have their order shuffled to ensure randomness.

Next, all sequences, original and scrambled, are vectorized as one-hot encodings and reshaped as input for each specific model type. One-hot encoding is a way to represent sequences with a set of characters in matrix form (a requirement for many ML systems). For example, a one-hot encoding representation of a DNA sequence is a matrix with four rows, one for each nucleotide, and k columns, one for each position in the sequence. In the one-hot encoding matrix, there is a one if a nucleotide occurs in that position and a zero elsewhere, so that each column typically sums to one. This one-hot encoding function is adapted from the pysster[61] library. For TPOT model inputs, the one-hot encoded matrix has an argmax function applied along the columns to generate a 'numerical_data_input'. For DeepSwarm and AutoKeras model inputs, the one-hot encoded matrix is generated along with a numerical data array that has been reformatted along four dimensions: a list of x arrays each of size y, z becomes an array of size x, z, y, 1, where the last dimension is a dummy dimension, so that sequences are treated like three-dimensional images. Functions to revert one-hot matrices to sequences were developed for all models and can be used by the user or different modules.

Two independent pre-processing routines are automatically applied to the target values, depending on the prediction task defined by the user (i.e., regression or binary classification). For regression tasks, target numerical values are transformed to follow a uniform distribution between -1 and +1 using quantile information to standardize model creation and evaluation. In the case of classification tasks for targets with floating point numerical values, automatic or manual thresholding (depending on 'automatic_bin') can be applied after quantile transformation, with the default threshold at 0.5. For multi-class classification tasks, we expect the output column to consist of labels with text-based categorical classes (e.g., three classes with the labels "a", "b", and "c"). As such, we do not perform additional pre-processing on multi-class classification labels.

Lastly, the dataset of sequences is automatically split into a user-specified number of "folds", where each fold is an equivalent portion of the dataset used to assess the model's ability to generalize, called k-fold cross validation. For example, a three-fold split would split the dataset into thirds, and train three models: one model trained on the first two folds and then tested on the third fold; one model trained on the first and third fold and tested on the second fold; and one model trained on the last two folds and tested on the first one. All performance values reported in the text as "one of k models" refers to k-fold cross validation.

Architecture search

After the pre-processing module is completed, appropriate inputs and outputs can be supplied to the model searches run by the three different AutoML backends. DeepSwarm takes the numerical data input as input and the target values as output (transformed as described above) with a 'to_categorical' transformation applied to it. TPOT takes the same numerical data as input and the target values as output without applying a categorical transformation. AutoKeras takes the one-hot encoded matrix input and the target values as output with no categorical transformation applied. Before DeepSwarm proceeds with the architecture search, a .yaml file is updated with parameters as follows: binary classification tasks optimize with respect to the binary cross-entropy loss, with the final layer having an output shape of 2; multi-class classification tasks optimize with respect to categorical cross-entropy loss, with an output shape equivalent to the number of categories in the data; and regression tasks optimize with respect to the mean squared error, with an output shape of 1. All models use the Adam optimizer by default. The user specifies the ant count, maximum depth search, epochs to train each model in the iterative loop, and final epochs to train the best model. For classification models, a softmax activation function is applied to the output node, while regression models apply a linear activation on the output node. The model architecture search explores filter sizes of 8, 16, 32, and 64, and kernel sizes of 1, 3, 5, and 7 for Conv2DNodes (standard convolutional filters). The model search explores output sizes of 30, 64, 128, and 256 for DenseNodes. The input node shape is automatically inferred from the shape of the data. All other arguments revert to DeepSwarm defaults. This parameter file (.yaml) is saved for the user's records.

Architecture search then proceeds for DeepSwarm models as specified in the original package documentation, where the DeepSwarm backend responsible for optimization is defined; the topology, or architecture, is found for a given dataset; the discovered topology is evaluated; and the base trained model is evaluated. Likewise, for AutoKeras, either the ImageClassifier or ImageRegressor object is used to train and find the optimal AutoKeras model, and said model is then fit to the data. The optimal AutoKeras graph is written to a PDF file for ease of interpretation. Analogously, the TPOT architecture search uses the TPOTClassifier and TPOTRegressor objects to find and fit the best model. When the optimal architecture has been found using all available training data, a k-fold model is trained over the user-defined number of folds and evaluated according to the task. The DeepSwarm, AutoKeras, and TPOT architecture training stages are back-ended by TensorFlow,[62] PyTorch,[63] and scikit-learn.[64]

For classification tasks, the ROC curves for each class are plotted along with the macro- and micro-averaged auROC. The Matthews correlation coefficient (MCC) for each class is computed as well. We report both MCC and the auROC, noting that the MCC metric provides a complete and reliable picture of the classifier's power in a single metric that is agnostic to class-size imbalances.[65] The MCC collapses the confusion matrix down to a single number, and as such provides information on both sensitivity and specificity of a classifier. In comparison, auROC does not readily account for class-size imbalances, which makes it prone to potential misrepresentations of the predictive value of trained systems. However, auROC is a common metric that is ubiquitous in the ML literature for classifiers, and thus is also reported in our system to permit ease of comparison with previously developed classifiers (which usually only report auROC). For regression tasks, the Spearman R, Pearson R, mean absolute error (MAE), and R^2 values are computed. These metrics are computed for each fold individually and globally for all folds (compiled_predictions) using scikit-learn[64] and SciPy[66] functions. Models trained on scrambled inputs, adhering to the optimal architecture found with the original data, are evaluated similarly to evaluate the effect of nucleotide, amino acid, or monosaccharide/bond composition on model performance. Results are written to text files (all_results.txt) in the output folders for each model. If the user indicates that a data robustness test should be performed, then provided the dataset length is greater than 1,000 sequences, the size of the dataset is sequentially halved, and the best architecture is used to train a model on that available data. We note that this experiment does not address the effect of less data on finding the optimal architecture; rather, it analyzes how effectively the architecture returned by AutoML can be trained on sequentially less and less data. Lastly, one final model is fit with all data with the optimal architecture, and a deployable model is saved along with a model architecture summary.

Feature importance and in silico mutagenesis plots

Our interpretation module includes several options for users. First, we offer the ability to assess feature importance for each position in an average sequence as predicted by the best-performing TPOT model. Although the TPOT models sometimes underperform models generated by AutoKeras or DeepSwarm, we offer feature importance analysis for TPOT models, as it is standard and supported by other automated ML tools like JADBio, BioAutoML, and iLearnPlus. Feature importance—the degree to which each position in a sequence causes variability in model predictions—is assessed within many scikit-learn models via the built-in feature importance functionality. Some scikit-learn models do not support feature importance, which triggers a notification to the user that feature importance cannot be computed for this model. For models that are a collection of multiple models—such as a random forest model with multiple estimators—error bars are computed by calculating the standard deviation of the feature importance for each model in the collection. If the model is not a collection of multiple estimators, no error bars can be computed.

Beyond TPOT models, *in silico* sequence mutagenesis plots are generated for all models produced by BioAutoMATED as an additional way to assess the importance of each position in a sequence. First, subsets of sequences are selected for analysis. For datasets with two or more classes, sequences are randomly sampled from each class. For datasets with continuous labels (even if they are used in a binary classification task), sequences are randomly selected from the top 10th percentile, the bottom 10th percentile, and a random sampling between the 20th and 80th percentiles. We sample from these percentile-based subsets, even for binary classification, to elucidate trends on the extreme ends of performance, e.g., the top-performing and poorly-performing sequences. For each set of N sequences (where N = sample_number_mutagenesis, default = 50), a sequence logo is generated of the raw sequences to aid in baseline sequence interpretation. Then, all possible subunit changes are generated, converted into one-hot encoded sequences appropriate for use in the model, and then passed as input to the previously trained model to predict the scores of these mutated one-hot encoded sequences. These scores are recorded, and the standard deviation of the model scores at each position is computed and plotted with 95% confidence intervals.

Class activation and saliency maps

Along with the aforementioned interpretation tools, additional techniques from the computer vision community are included for analyses of more complex neural networks. Saliency maps and class activation maps both provide methods of assessing the model "attention" to each nucleotide, amino acid, or monosaccharide/bond at each position in a set of sequences. Class activation maps are generated by calculating user-specified gradients with respect to user-specified layers. In the absence of a specific user request, sensible defaults allow the automatic generation of these maps. By using the learned model weights directly, class activation maps uncover specific subunits at specific positions that play outsized roles in model predictions. Sequence logos are then generated automatically from the class activation maps to facilitate functional understanding of the sequence. Similarly, saliency maps—which differ from class activation maps in the method used to calculate attention—are generated by again finding user-specified gradients with respect to user-specified layers. The sequence logos of the weights of these saliency maps can then be generated.

For the optimal DeepSwarm model, class activation maps[67] are visualized with the visualize_activation function from keras-vis.[68] Gradients can be modified with a function of the user's choice (class_activation_map_grad_modifier), with one of the following options: 'absolute', 'relu', 'negative', 'invert', 'small_values', or None, which does not modify any gradients. The layer index – the specific model layer whose filters should be visualized – can also be specified (class_activation_map_layer_index). Class activation maps are then generated for a sample of N sequences (where N = sample_number_class_activation_maps, default = 100) and subsequently averaged over all N sequences. Likewise, the saliency maps[69] are generated with the best DeepSwarm model using the visualize_saliency function from keras-vis. Users can specify the following arguments: saliency_map_grad_modifier, saliency_map_layer_index, and sample_number_saliency_maps.

We note that class activation maps are most useful when visualizing a layer close to the last layer. If this is not the case, we recommend using saliency maps instead. We also note that for both saliency maps and class activation maps, the output may be less informative if the last layer is fully connected and has a softmax activation applied. In this case, the activation function can be swapped out for a linear activation function. For both visualization techniques, a sequence logo is generated using LogoMaker[70] to aid in interpretation. Glycoletters are converted into single character abbreviations to be represented in a sequence logo. For saliency map sequence logos, the matrix of saliency map counts is converted to weights, while activation map counts are converted to probabilities (an arbitrary design choice). BioAutoMATED automatically handles axis labelling, color schemes, and figure size standardization.

Naive design module

If the run_design option is selected, then the user has several options for design via random sequence mutagenesis. First, the user can select how many positions they would like to mutate per sequence (k) and how they would like to mutate sequences (‘random’, ‘constrained_random’, ‘blocked’, or ‘constrained_blocked’). The blocked mutagenesis method generates every possible k-mer and then tiles the sequence with these k-mers. Upon sampling sequences from similar sets as described in the interpretation module, a random sample of de_novo_num_seqs_to_test (default = 100) is selected from each set. Depending on the alphabet size, sequence length, and k, the module may lower k based on a pre-determined threshold set to prevent excessive compute use. All mutated sequences are then generated.

Sequence constraints are enforced before passing mutated sequences through a predictive model if a constraint_file_path is provided and the design method is either ‘constrained_random’ or ‘constrained_blocked’. This constraint file must contain the following information by column: (1) type of constraint (one of [‘exact_seq’ or ‘reverse_complement’]); (2) exact sequence if constraint type is ‘exact_seq’; (3) start position; (4) end position; (5) complement start; and (6) complement end. For example, a constraint specifying that the exact nucleotides “AGA” must be present in position 0–2 in any generated sequence would have a row reading: ‘exact_seq’; “AGA”; 0; 2; -; -. For a reverse complement constraint enforcing that the letters in positions 16–19 must be the reverse complement of bases in positions 0–3, the row would read: ‘reverse_complement’; -; 0; 3; 16; 19. The reverse complement constraint may only be specified for nucleic acid sequences, while any sequence type can have an exact sequence constraint. All constraint sequences are cleaned and processed according to the same guidelines in the pre-processing module (e.g., uracils are converted to thymines in nucleic acid sequences, and characters are standardized in upper case). The constraint-enforced list of mutated sequences and the original sequences are then passed to the previously trained predictive model. The mutated sequences are written to a csv file in the design folder and are plotted adjacent to the original sequences and the STORM-designed sequences (below). These plots are generated for all sequences, as well as the top 10% of sequences generated with each method. A histogram of the pairwise cosine distances between each set of sequences is plotted in order to evaluate similarity. These plots are written to the design folder along with a sequence logo of the mutated sequences for visual comparison.

Directed design module

The directed design module is extended from SeqProp, as described in Bogard et al.[28] and STORM as adapted in Valeri, Collins, Ramesh et al.[29] Here, our extension provides many additional degrees of flexibility and methods for customization. As with naive design, the user can specify any constraint file for design to adhere to (constraint_file_path), as well as several sequences to test (storm_num_seqs_to_test, default = 5). The user can set the class of interest (class_of_interest, default = 1 for classification and 0 for regression), the positive class in a binary classification task), and the target label value that sequences should be optimized to obtain (target_y, default = 1, the highest value in a normalized dataset). The user can also specify the number of STORM optimization rounds to perform (num_of_optimization_rounds, default = 5). After these user parameters are set, the computation proceeds as described in Valeri, Collins, Ramesh et al.[29]

To facilitate optimization for any nucleic acid, protein, or glycan sequence, we have adapted the entire SeqProp library from Bogard et al.[28] to execute with any alphabet and alphabet size (extended from the four-letter nucleotide library) as well as any sequence length, any constraint, and any architecture of DeepSwarm model. Complete documentation may be found in seqprop_helpers.py and integrated_design_helpers.py. In short, the position weight matrix (PWM) of a one-hot encoded sequence is added as a layer to a copy of the previously trained best DeepSwarm model. The PWM layer may vary during training, but all other layers are frozen so that backpropagation acts solely on the PWM, with the optimization minimizing the difference between the target value and the actual predicted value. An Adam optimizer with a learning rate of 0.001, beta_1 of 0.9, and beta_2 of 0.999, along with an Early Stopping regime that monitors the loss with a min_delta of 0.01 and patience of 2 is used to train the model for 50 epochs with 1000 steps per epoch. This Early Stopping regime is a relatively strict procedure intended to produce sequences with sufficient diversity. The optimized PWM is then cleaved from the model and converted back to a sequence. While the STORM optimizer may converge to a perfect target during the STORM optimization, the predicted value corresponding to the final sequence may be lower because the sparse one-hot matrix corresponding to the raw sequence may predict a different value than the tuned position weight matrix. Nevertheless, this optimization protocol usually results in a more substantial increase in predicted score than in random mutagenesis. We note that this gradient ascent optimization regime is appropriate for producing a small number of redesigned sequences because the resulting sequences are often limited in diversity and the protocol takes a substantial amount of time per sequence. Despite this limitation, these sequences are typically higher-performing than those obtained from random mutagenesis, and previous work has shown that the predicted scores translate well to *in vitro* efficacy.[29] The original model score is reported alongside the STORM-predicted score (i.e., the score predicted by the PWM-fusion model) and visualized adjacent to the naively designed sequences and original sequences.

Transfer learning functionality

Depending on the underlying model type, we demonstrate multiple techniques to re-train models: these include freezing certain layers of the model during re-training, initializing a new model with the parameters of the best BioAutoMATED model, and creating new models within an existing ensemble that are then trained on new data. Users may benefit from the option to specify multiple datasets for more robust fine-tuning or validation.[71] For example, a user may upload a large RBS dataset from *E. coli*, use BioAutoMATED to find a predictive model for translation efficiency, and subsequently re-train that model on a second RBS dataset from another strain of *E. coli*. These functionalities can contribute to maintaining the relevance of BioAutoMATED models, even as biological dataset sizes continue to grow in size and quality.

To facilitate re-use of the models generated by BioAutoMATED, we have provided a Jupyter notebook with examples demonstrating how to load and re-train any model produced by BioAutoMATED. We first load a generic DeepSwarm model and print a model summary with the current architecture, state of the model (i.e., a state variable describing whether the model is currently trainable), and the number of trainable parameters in the model. In the provided example, we arbitrarily specify that all layers except the last two layers should be frozen during the next round of training, which, as we also demonstrate, produces a deployable model. We then show an example loading an AutoKeras model and re-using the weights of this model to initialize and train a new model. We demonstrate how to load an AutoKeras model and re-use the architecture, but not the weights, to train a new model. Lastly, we apply several different methods for "incremental learning" on TPOT scikit-learn-based models. After loading a model, we first check if the model accommodates the partial_fit function, which allows a user to resume training based on the current state of the model. Unfortunately, the partial_fit functionality is only applicable for a small number of models (as detailed in [https://scikit-learn.org/0.15/modules/scaling_strategies.html#incremental-learning]). Models not compatible with the partial_fit function cannot be fully re-trained from a specified starting point, but a new estimator (i.e., a new model in the ensemble) can be added, so that re-training with new data will update the newest model only. For this option, the 'warm_start' parameter can be changed, and the number of estimators in the model can be increased by one. To facilitate these changes, we print out a list of keys that must be manually changed by the user and show an example modifying these parameters and re-training the model.

Proof of concept with experimental data

For ribosome binding site (RBS) sequence analyses, the training and test sets of RBS sequences from the dataset collected and analyzed by Höllerer et al.[30] were downloaded from the following GitHub link: [https://github.com/BorgwardtLab/SAPIENs/tree/main/data]. The validation tasks described in Table S1 were performed using the test set. For binary classification tasks, the BioAutoMATED platform inferred an automatic threshold for the median value used to separate positive and negative classes. For multi-class classification tasks, the data were split into four quartiles and labeled as "a", "b", "c", and "d" classes, where RBS sequences with the "a" label were in the bottom quartile and RBS sequences with the "d" label were in the top quartile. For our augmentation tests (Figure S3), we aimed to highlight a case where data augmentation is essential. As such, we artificially limited ourselves to the "small data" domain by using a dataset of only the first 2,000 sequences. All other experimental and model hyperparameters remained set to their default values.

For all results pertaining to peptide sequences, training and testing data as described in Liu et al.[31] were retrieved from the following GitHub link: [https://github.com/gifford-lab/antibody-2019/tree/master/data/training%20data]. Regression data were downloaded from the full_regression folder, giving a total of 67,769 sequences with labels. Classification data were downloaded from the hold_out_classification folder, giving a total of 63,400 sequences in the training set and 471 sequences in the test set (the latter of which was used to compute external validation tests, as in Table S1). Binary and multi-class classification pre-processing were identical to those employed for the RBS datasets. All other arguments were set to their default values.

For binary classification tasks on glycans, Table S2 from Bojar et al.[32] was used as input for glycan immunogenicity modeling, giving 636 immunogenic glycans and 684 non-immunogenic glycans. For multi-class classification tasks on glycans, Table S1 from Bojar et al.[32] was used as input for phylogenetic domain classification, with 34 archaea, 5,856 bacteria, 6,635 eukarya, and 149 virus glycan sequences, giving a total of 12,674 sequences. We note here that viruses are classified as a phylogenetic domain for analysis purposes. Unless otherwise specified, for all binary classification tasks, glycan sequences were padded to maximum length. For multi-class classification tasks, glycan sequences were padded to average length, due to the larger number of examples and the high dimensionality of the glycan sequence composition. All padding options are explored in Figure S4.

We analyzed toehold switch sequences as described in Valeri, Collins, Ramesh et al.[29] Training data were downloaded from the following GitHub link: [https://github.com/midas-wyss/engineered-riboregulator-ML/blob/master/data/newQC_toehold_data.csv]. We processed the data in the same manner as Valeri, Collins, Ramesh et al.[29] and retained only sequences with ON_qc and OFF_qc scores over 1, resulting in 91,534 59-nucleotide sequences. The "ON" column was used as the target value for all BioAutoMATED models. External datasets for the validation described in Table S2 were obtained from the same GitHub repository at the following link: [https://github.com/midas-wyss/engineered-riboregulator-ML/tree/master/clean_figures/fig4/make_tf_learning_models], containing both the Green et al.[47] and Pardee et al.[48] datasets.

For synthetic nucleic acids, we generated 100,000 random 20-nucleotide sequences. Then, an artificial target score was assigned to each sequence by summing up the scores for each nucleotide in the sequence. We arbitrarily designed a scoring system where A confers 1 point, T confers 2 points, C confers 3 points, and G confers 4 points. This set of sequences and targets should be easy for any model to learn due to its rule-based simplicity, so we used this dataset as a positive control task to ensure that BioAutoMATED performed as expected.

For all datasets, unless otherwise specified, the maximum runtime was set to 60 minutes. For glycan datasets, the maximum runtime was set to 180 minutes. For all datasets, the number of folds was set to 3, and DeepSwarm and TPOT parameters were set to default values. Interpretation module parameters were as follows: sample_number_class_activation_maps = 100; class_activation_grad_modifier = absolute; class_activation_layer_index = -2; sample_number_saliency_maps = 100; saliency_map_grad_modifier = absolute; saliency_map_layer_index = -1; and sample_number_mutagenesis = 50. Design module parameters were as follows: k = 3; substitutition_type = random; target_y = 1; class_of_interest = 1; de_novo_num_seqs_to_test = 100;

storm_num_seqs_to_test = 5; num_of_optimization_rounds = 5. For all sequences except toehold switch sequences, no constraints were specified. For toehold switch sequences, substitution_type was set to constrained_random and a constraint file that listed the following constraints was supplied: (1) 'exact_sequence'; ''AACAGAGGAGA''; 30; 40; -; -. (2) 'exact_sequence'; ''AUG''; 47;49; -; -. (3) 'reverse_complement'; -; 12; 20; 50; 58. and (4) 'reverse_complement'; -; 24; 29; 41; 46. Time measurements, as in Figure S2, were quantified with these design parameters. For the designed sequence results shown in Figures 2J, 2K, 3G, 3H, 4D, 4E, and 5G, the design parameters were modified to k = 1 and num_of_optimization_rounds = 3, with all other parameters kept the same.

Scope of the platform

To guide users in applying BioAutoMATED to their datasets, we offer recommendations for dataset properties. These recommendations enable users to best maximize the utility of our tool in practice. Note that these are not intended to be hard rules: rather, our suggestions provide additional context for specific types of datasets.

First, we consider sequence length. We recommend using sequences of up to 500 nucleotides, amino acids, or monosaccharides/bonds. Sequences greater than 1,000 subunits in length may be modeled well by BioAutoMATED, but the time required to produce models scales with the length of sequences. Additionally, recurrent neural networks, which are known for their ability to model longer-range sequence interactions,[72] are not currently evaluated within BioAutoMATED, so the types of models we assess may not be appropriate for extremely long-range sequences. With these limitations in mind, we have evaluated performance over additional datasets with longer-range sequences to assess the scalability of BioAutoMATED. We tested the AAV protein dataset introduced by Bryant et al.[73] and recommended as a benchmark dataset in Dallago, Mou et al.[74] for testing models that map adeno-associated virus capsid protein sequences ~740 amino acids long. These AAV capsid protein sequences have a large conserved region, with mutations limited to a 28 amino acid region corresponding to the binding interface, but were one of the longer protein datasets with labelled data. This dataset predictably takes a longer amount of time to run, with each AutoML tool taking between 1 and ~10 hours to find the best regression model. However, the best BioAutoMATED model outperforms the best reported model applied to this dataset for randomly sampled splits, with a Spearman correlation of 0.925 on cross-fold validation and a Spearman correlation of 0.930 on the test set (as compared to the best reported model, which achieves a Spearman correlation of 0.92 on the test set). To assess long range sequences with mutations throughout the length of a sequence, instead of clustered in a shorter region, we analyzed a set of avGFP proteins of length 237 reported by Sarkisyan et al.[75] and benchmarked in Gelman et al.[76] and Xu et al.[77] Here, BioAutoMATED produces models with a Pearson correlation of 0.970 on cross-fold validation and a Pearson correlation of 0.936 on the test set. These models are generated in a few hours, and their Pearson correlation value is comparable to the >0.9 Pearson correlation values reported by Gelman et al.[76]

Users may also benefit from additional guidelines on the number of sequences per dataset and the heterogeneity of datasets. We have tested BioAutoMATED with datasets of hundreds of thousands of sequences. Depending on the compute power afforded by the user's machine, any user may be able to model millions of sequences. However, BioAutoMATED is fastest for datasets of less than half a million sequences and has not yet been tested with larger datasets. One option that users have is to run BioAutoMATED with a subset of their sequences to find a starting point for their architecture search, and then explore a more limited set of models based on the results from BioAutoMATED. Additionally, BioAutoMATED has not been tested on datasets with large class imbalances, which can be common when studying certain questions. We recommend caution and additional scrutiny if using datasets with greater than 90%/10% class imbalance or datasets with unexpected distributions of target values. Lastly, we currently only provide sequence length normalization options by padding or truncating the right end of sequences. Future improvements on BioAutoMATED may include more advanced sequence length normalization or incorporate more sophisticated alignment abilities, such as those reported recently by Zhang et al.[78]

Machine specifications and package environments

A MacBook with a 2.2 GHz Intel Core i7 chip and 16GB RAM was employed to run our platform and used to generate the elapsed times shown in Figure S2. We also tested our code on a virtual machine on Google Cloud Platform (GCP), utilizing a n1-highmem-8 instance with 8 vCPUs with approximately 55 GB of RAM and a 100 GB boot disk running Ubuntu 16.04. Instructions for installing packages and dependencies can be found in the installation guide included in our GitHub repository. A full list of tested systems is detailed in the installation guide. We note that managing package compatibility is important, and we recommend that users exercise caution when changing any packages or versions. For users who feel less comfortable with package installations and/or users with hardware that does not support TensorFlow 1.13.1 (for example, those with Macs containing M1 chips), the DockerHub repository can be easily downloaded and installed. Complete instructions for using Docker can be found in the installation guide included in the GitHub repository. All installation methods provide Jupyter notebooks containing examples of using BioAutoMATED and the resulting models for external validation, interpretation, design, and transfer learning. We also provide instructions for running BioAutoMATED via the command line, instead of within Jupyter notebooks.

Benchmarking against automated ML tools

To compare BioAutoMATED against other AutoML tools, we performed benchmarking experiments with several available and biologically focused automated ML tools, namely the iLearnPlus,[49] BioAutoML,[50] and JADBio[51] platforms. BioNAS[79] was not used in this investigation due to installation issues [https://github.com/zj-zhang/BioNAS-pub/issues/1] that had not yet been resolved by the time of publication of this work. While there are other platforms designed to operate and train select ML architectures for generic biological sequences (e.g., pyssster,[61] Selene,[5] BioSeq-Analysis2.0,[7] etc.), these platforms do not contain specialized algorithms to conduct automatic evaluations of multiple types of architectures and/or perform automated optimization for these models. With these non-automated ML tools, users must still select the architecture of interest and relevant parameters. Thus, these tools are not suitable for direct comparison with BioAutoMATED. We therefore limited our evaluation to platforms that conduct automated architecture and hyperparameter searches for biological data. Five datasets corresponding to RBS sequences, peptides, toehold switches, 1,000 synthetic nucleic acids (''small synthetic dataset''), and 100,000 synthetic nucleic acids (''large synthetic dataset'') were tested using five performance metrics each: binary classification area-under-ROC curve (auROC), binary classification Matthews correlation coefficient (MCC), regression R^2, regression mean absolute error (MAE), and regression correlation coefficient.

First, we explored iLearnPlus,[49] one of the first tools that automated ML for biological sequences (i.e., DNA, RNA, and peptides). iLearnPlus provides a website, as well as the ability to download the code and run its interface locally on any machine. We assessed our datasets through the latter option, as the website only accommodates up to 2,000 sequences. The user must select the feature representation for sequences (e.g., k-mer representation, electron-ion interaction potential, etc.). To mimic a user without prior knowledge of the best feature representations, we selected the binary feature representation for all datasets, which effectively creates a one-hot encoding for each sequence and then flattens this matrix into a one-dimensional vector. An additional hands-on step is required, wherein a user must generate this input and then upload the input into the AutoML module. iLearnPlus[49] enables 21 different ML algorithms (12 conventional ML methods, two ensemble-learning frameworks, and seven deep learning frameworks). Again, in the spirit of mimicking the behavior of an average user interacting with the system, we selected a representative set of four model types: (1) random forest classifiers; (2) decision trees; (3) multi-layer perceptrons or MLPs; and (4) XGBoost, and selected the ''auto-optimize'' option when offered. For all model types, we used five-fold cross validation. A user who assesses all model types with all possible feature encodings (representing hundreds of combinations) may obtain better performance than we reported; nevertheless, it is infeasible to expect the average user to conduct these optimizations on their own. Additionally, iLearnPlus[49] only accommodates classification, and regression tasks could not be assessed using this platform. We binarized continuous values by assigning any sequence with a value above the median to a positive example and any sequence with a value equal to or below the median to a negative example.

We next assessed BioAutoML,[50] which has both automated feature engineering and automated ML capabilities. Like iLearnPlus,[49] BioAutoML[50] only works for classification tasks. Unfortunately, BioAutoML[50] does not report the MCC. BioAutoML[50] reports the confusion matrix and does not report the models' predictions on the held-out test sequences; the information provided is therefore insufficient to calculate the MCC. Due to these limitations, BioAutoML[50] was used to benchmark five out of the 25 tasks, corresponding to the five auROC metrics for the binary classification tasks. Of these five tasks, we found that BioAutoML[50] outperforms other tools on the peptide dataset and performs equally well as iLearnPlus[49] and BioAutoMATED for the large synthetic dataset. We note that the feature engineering in BioAutoML[50] can, in some cases, provide a compelling advantage, especially on sequences with a higher dimensionality such as peptides. However, BioAutoML[50] cannot handle non-canonical (e.g., J representing leucine/isoleucine) amino acids for peptides. We therefore replaced all J amino acids with leucine, which could have affected the final set of metrics computed. BioAutoML[50] also cannot be used on a Mac; for researchers with limited computational experience who desire to leverage the power of ML in their work, the process of setting up a virtual machine or using a different operating system to accommodate BioAutoML could be a barrier.

Lastly, we assessed JADBio,[51] an ML platform for generic biological datasets. JADBio, due to its exclusive use of vectors as input, is the only platform we assessed that cannot handle raw nucleic acid or peptide sequences. The input for JADBio[51] is restricted to one-dimensional vectors, so we used the flattened binary representation generated by iLearnPlus. We note that other representations may have been used, but typical users may not know how to create and assess different feature representations. JADBio[51] is limited to datasets of less than 100,000 sequences, so we used a smaller dataset of 50,000 ribosome binding site sequences instead of the full set of ~276,000 sequences. By default, JADBio[51] only uses a select subset of features to train its ML models. Here, we also explored the alternative option of selecting all features, which we hypothesized would result in a fairer performance comparison. We found that JADBio[51] did not achieve perfect classification or regression performance on the synthetic nucleic acid datasets (our positive controls), and the authors of this tool report that the JADBio[51] platform intentionally underestimates the performance of its models for better generalizability projections. It is possible that, had the tool reported raw performance values, its models may exceed those of BioAutoMATED in predictive capability; with only the values provided by JADBio,[51] we cannot currently perform this comparison. Additionally, JADBio reports ''correlation coefficient'', which we compare to our Pearson correlation coefficient. We also note that JADBio's interface is exceptionally easy to use, and they also provide computing power for their users, as this tool is a commercial product. However, it is difficult to appropriately compare a commercial product with an academic tool; some of their features, for example, some model interpretation, are hidden behind paywalls, hindering closer investigation.

Benchmarking against pre-trained language models

Unlike language models that are trained on large datasets of nucleic acid and peptide sequences with varying sizes and unknown functions, BioAutoMATED models must take input sequences with labelled functions of interest. Thus, for these comparisons, we re-trained generic language models on a task-specific dataset of sequences of specific lengths; for example, we retrained a generic DNA model on synthetic nucleic acids that are all 20 nucleotides long.

First, we compared BioAutoMATED's performance on three nucleic acid sequence datasets to the performance of DNABERT.[52] DNABERT is a pre-trained language model that was trained for 25 days on eight NVIDIA 2080Ti GPUs on a massive dataset of unbiased DNA sequences.[52] DNABERT can be re-trained on user-provided data if the user specifies the training parameters, which include the k-mer length, learning rate, weight decay, and other parameters. We performed a minimal parameter search by evaluating pre-trained models with k-mer sizes of 3 and 6 and learning rates of $2x10^{-4}$ (provided in the example GitHub code) and $5x10^{-5}$ (the default learning rate within DNABERT's source code). We set all other parameters as default for the fine-tuning stage: per_gpu_eval_batch_size = 8; per_gpu_train_batch_size = 8; num_train_epochs = 3.0; warmup_percent = 0; hidden_dropout_prob = 0.1; weight_decay = 0.0. As the early_stop parameter did not appear to effectively stop the model training regimen [https://github.com/jerryji1993/DNABERT/issues/98], we evaluated performance of all models at the final training step.

To assess peptide sequence performance, we compared BioAutoMATED models with Facebook's ESM[53] model, a model that has been pre-trained on large-scale protein datasets. Unfortunately, ESM does not support fine-tuning of their models and reports that they do not plan to support fine-tuning when asked to add that feature in GitHub requests (please see, for instance, https://github.com/facebookresearch/esm/discussions/33 and https://github.com/facebookresearch/esm/issues/30). The ESM authors have published a protein-based pre-trained model and code to calculate embeddings from this pre-trained model, which can then be used as inputs to simple scikit-learn-based regression heads; we thus used this code to benchmark BioAutoMATED. It is possible that fine-tuning ESM may result in better performance on task-specific datasets, but this is currently not supported. It is also possible that using ESM embeddings as input to more complex models may lead to higher performance, but we did not explore this option.

We compared three scikit-learn model types: K-nearest neighbors regressor, random forest regressor, and support vector machine regressor (SVR). We first compared training performance. For the scikit-learn based models, we evaluated both the three-fold cross validation performance during training on the held-out fold, as well as on a held-out 20% test set. We compared this performance to the best-performing BioAutoMATED models, which were TPOT models, and their automatically-calculated R^2 values during three-fold cross-validation training. With models trained on 100% of the data, we then evaluated the performance on the external validation held-out test set used in Liu et al.[31] As ESM embeddings do not support non-conventional amino acid encoding, we replaced all J amino acids (indicating isoleucine or leucine) with L (leucine). After using the scripts/extract.py code with the pre-trained esm2_t33_650M_UR50D model to extract embeddings, we utilized all defaults for training (embedding layer = 33; repr_layers = 0 32 33; include_mean_per_tok).

QUANTIFICATION AND STATISTICAL ANALYSIS

All statistical analyses are detailed in the corresponding figure legends and results section of the main text. Unless otherwise specified, significance was set at a threshold of $p < 0.005$ and denoted with an asterisk. Statistical significance was computed using scipy.stats. Randomization was performed using the Python "random" package.

Cell Systems, Volume *14*

Supplemental information

BioAutoMATED: An end-to-end automated machine learning tool for explanation and design of biological sequences

Jacqueline A. Valeri, Luis R. Soenksen, Katherine M. Collins, Pradeep Ramesh, George Cai, Rani Powers, Nicolaas M. Angenent-Mari, Diogo M. Camacho, Felix Wong, Timothy K. Lu, and James J. Collins

SUPPLEMENTAL ITEMS

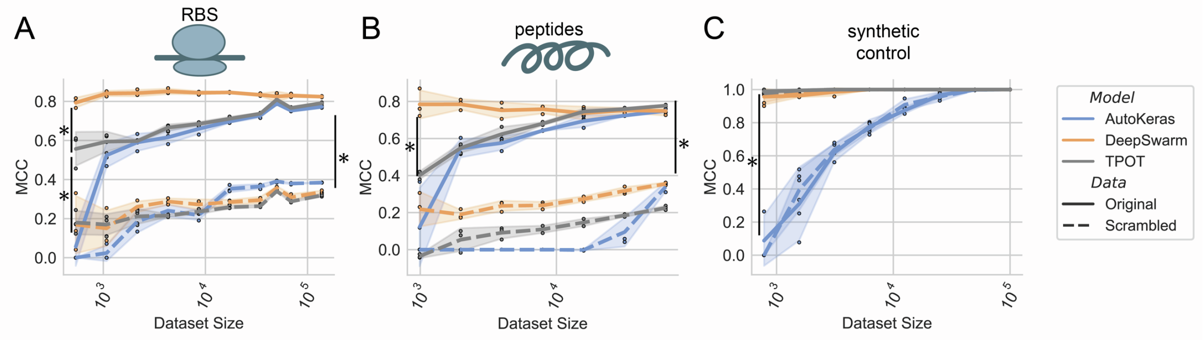


Figure S1. Data ablation experiments automatically performed with BioAutoMATED for all model types help evaluate dataset saturation and model robustness; Related to Figure 1. (A) DeepSwarm models on ribosome binding site sequences are data efficient. At the lowest dataset size, DeepSwarm achieves a significantly higher Matthews Correlation Coefficient (MCC) than both TPOT and AutoKeras, and TPOT performs better than AutoKeras. At the full dataset size, all models perform significantly better than their respective scrambled control models. (**B)** DeepSwarm is saturated on the peptide dataset while other methods can achieve higher Matthews Correlation Coefficients (MCCs) with more training data. At the lowest dataset size, DeepSwarm outperforms TPOT. At the full dataset size, all models perform significantly better than their respective scrambled control models. **(C)** Models trained on synthetic nucleic acid sequences show the necessity of training data for AutoKeras models, where performance drops off almost linearly with decreasing amounts of data. At the lowest dataset size, DeepSwarm and AutoKeras both achieve a significantly higher MCC value than TPOT. At the full dataset size, all models perform equally to their respective scrambled control models. All data robustness experiments were performed using the best binary classification models for each architecture search and each sequence type. For all panels, points correspond to one of N=3 folds and asterisks indicate significance with $p < 0.005$ using two-sided t-tests.

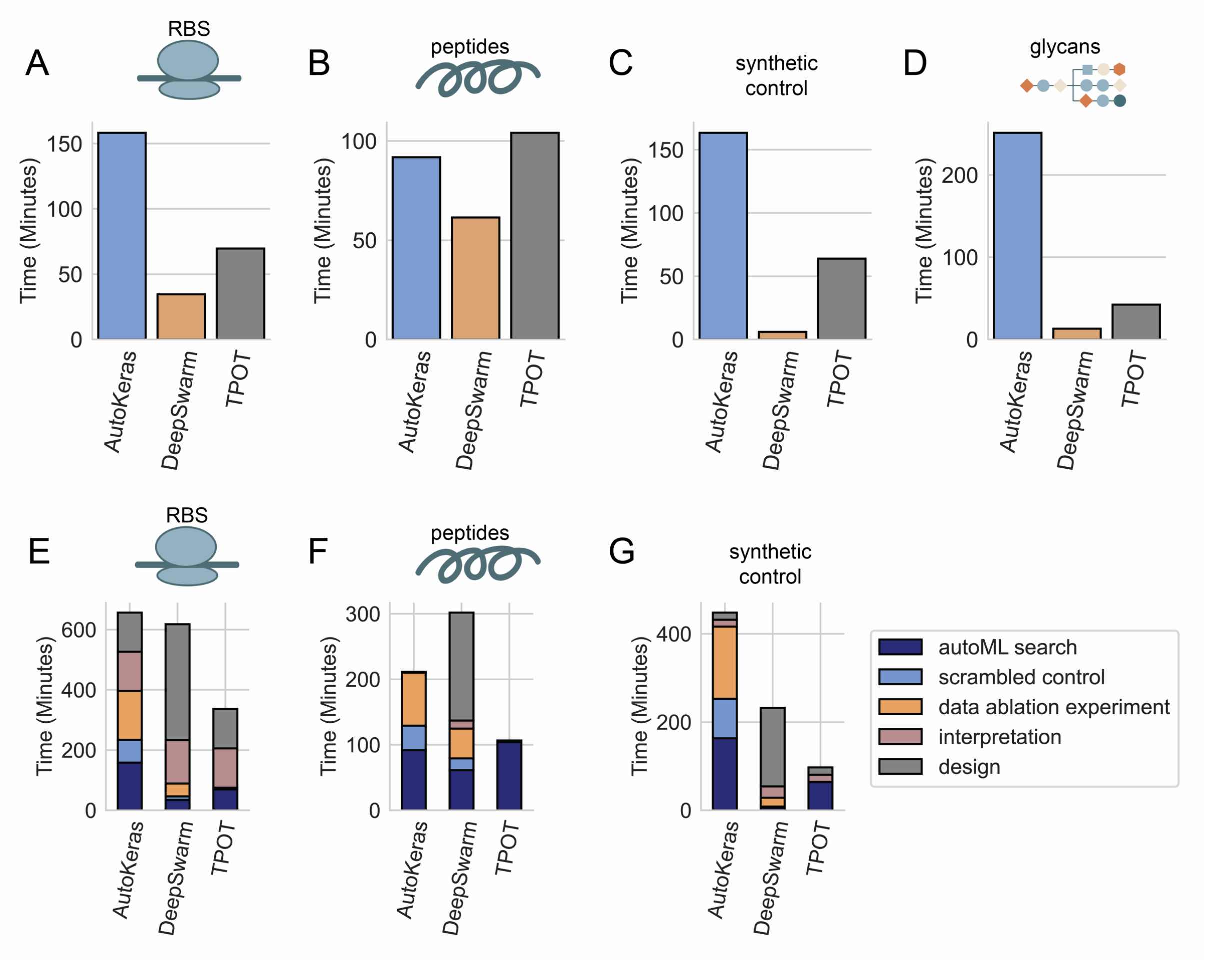


Figure S2. AutoML model search time is automatically evaluated for all model types; Related to Figure 1. The AutoML model search time alone is provided for **(A)** ribosome binding site sequences, **(B)** peptides, **(C)** synthetic nucleic acid sequences, and **(D)** glycan sequences. For all datasets, the DeepSwarm algorithm identifies an optimal model architecture in the shortest time. BioAutoMATED also provides a comprehensive time breakdown for **(E)** ribosome binding sites, **(F)** peptides, and **(G)** synthetic nucleic acid control sequences. For all sequence types, the most time-intensive step in the DeepSwarm platform is sequence design. The time breakdown was not computed for glycans because the interpretation and design modules were run separately from the AutoML model search for glycan prediction tasks. All time calculations were computed using binary classification models.

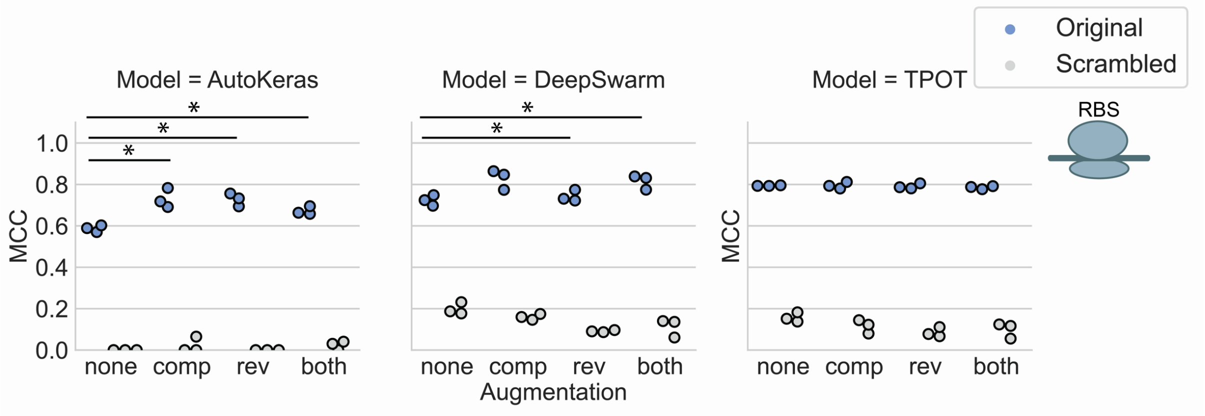


Figure S3. An augmentation option is available for nucleotide sequence-based datasets; Related to Figure 1. Here, with a reduced sample of RBS sequences (N=2,000), we see that the synthetically complemented dataset yields an AutoKeras model with a higher MCC value than the original dataset ($p < 0.005$). Reverse complementation achieves a higher MCC than the original dataset ($p < 0.05$) and both complementation techniques achieve a higher MCC than the original dataset ($p < 0.005$). For DeepSwarm models, the reverse complementation and both complementation techniques yield higher MCCs ($p < 0.05$). Asterisks indicate significance as specified using two-sided t-tests. Points correspond to one of N=3 folds.

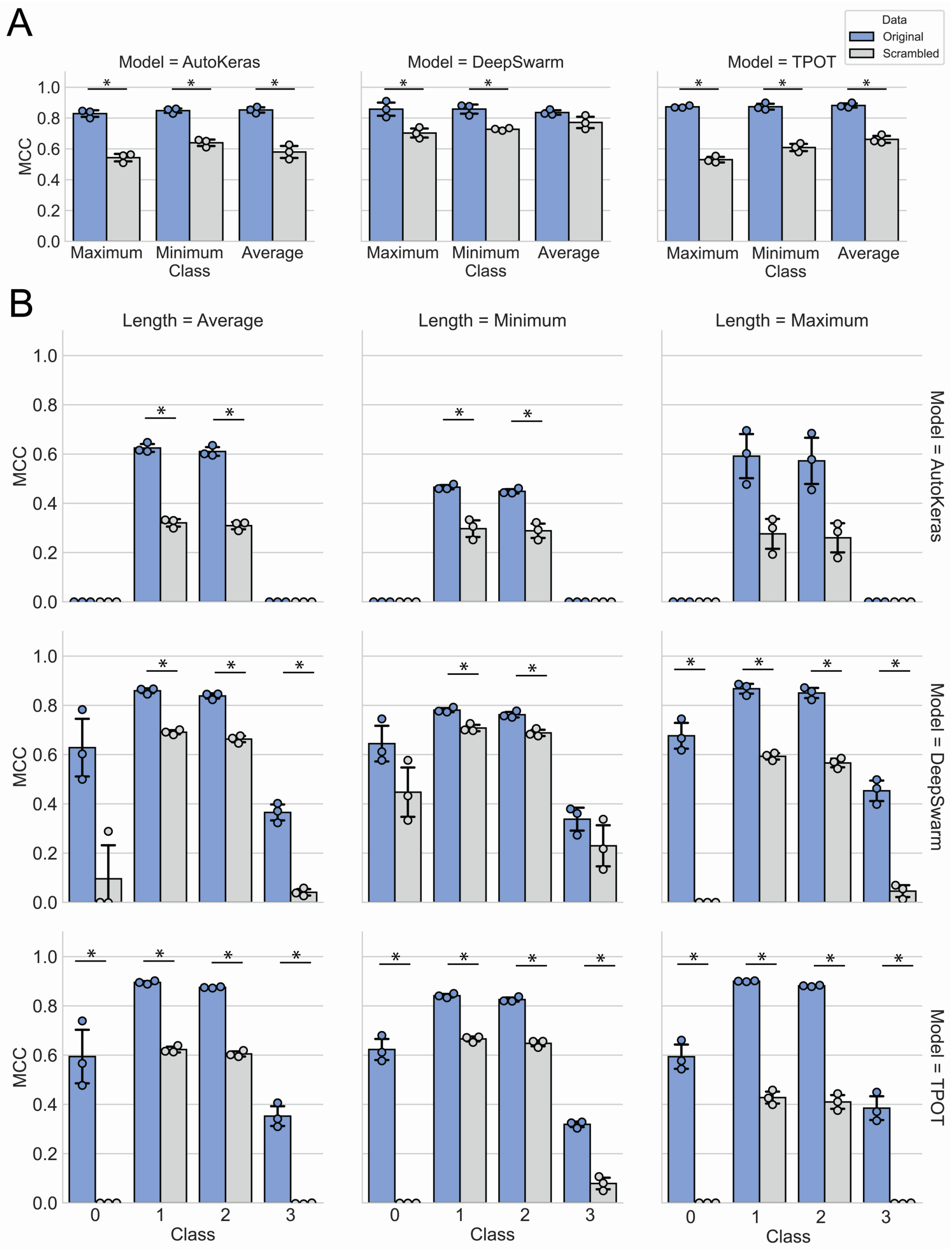


Figure S4. For sequences of heterogeneous lengths, BioAutoMATED employs length standardization options; Related to Figure 1. (A) Users are provided with the options for truncation to the shortest sequence length, padding to the longest sequence length, or truncation and padding to the average sequence length. For an immunogenicity binary classification task on glycan sequences, most length standardization options result in roughly equivalent model performance. For all models, the minimum and maximum padding options result in significantly better performance for models trained on the original sequences compared to scrambled sequences. The average padding option in DeepSwarm models fails to significantly outperform the scrambled control models, indicating that information about meaningful sequence patterns may be lost when averaging sequence lengths. **(B)** For a multi-class phylogenetic domain classification task, models trained on maximum length sequences appear to have the strongest performance for original sequences and poorest performance for scrambled sequences. The TPOT models outperform DeepSwarm and AutoKeras models and are the only AutoML search algorithm to result in significantly higher performance than models trained on scrambled sequences for all padding options and all classes. For all panels, points correspond to one of N=3 folds and asterisks indicate significance with $p < 0.005$ using two-sided t-tests.

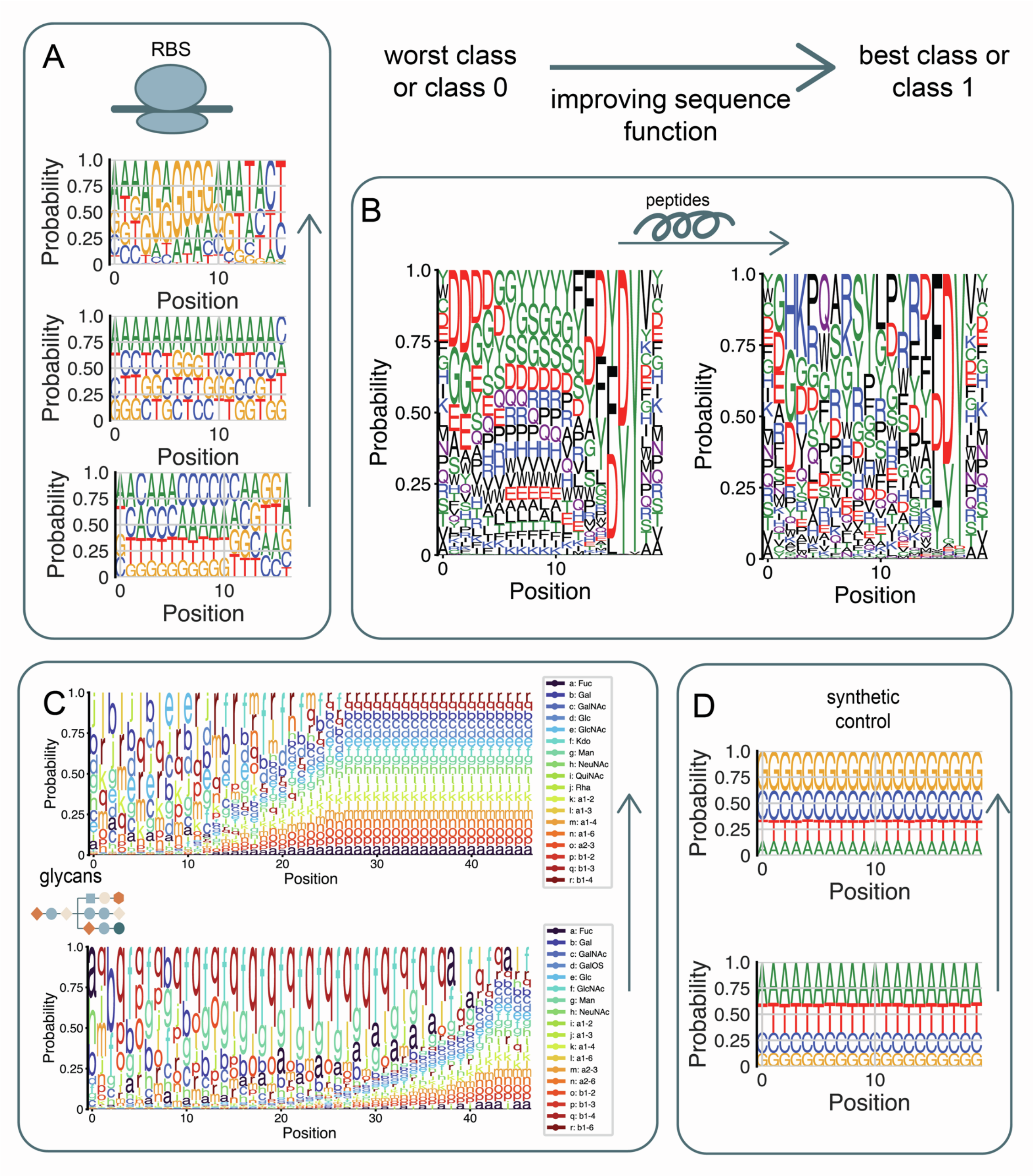


Figure S5: Raw sequences logos from experimental data contextualize model interpretation; Related to Figures 2, 3, 4, and S6. Arrows signify improving sequence function, where the left end of an arrow corresponds to the worst (or 0) class sequences, and the right end of an arrow corresponds to the best (or 1) class sequences. Sequence logos are computed for **(A)** RBS sequences, **(B)** peptide sequences, **(C)** immunogenic glycan sequences, and **(D)** synthetic nucleic acid sequences. All sequence logos are computed with samples of N=50 sequences.

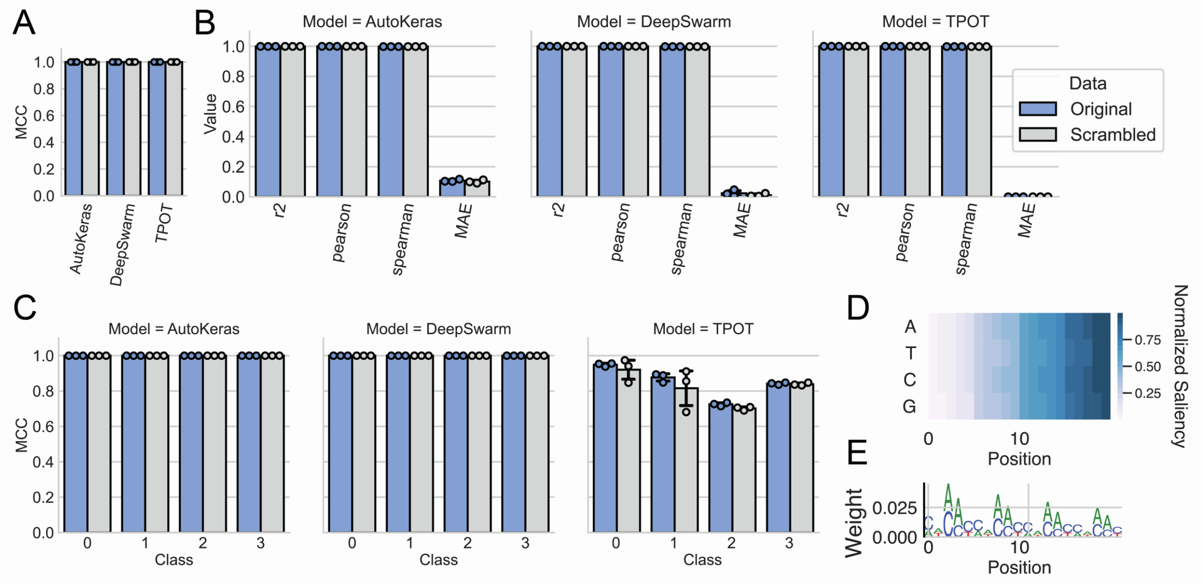


Figure S6. A synthetic nucleic acid control dataset benchmarks BioAutoMATED under ideal conditions; Related to Table 1. (A) A synthetic control set of 20-nucleotide sequences was scored according to a simple sum of their nucleotides with the arbitrary scoring regime: A = 1, T = 2, C = 3, G = 4. Models trained on these synthetic sequences as well as scrambled sequences have equivalent performance, serving as a successful positive control. **(B)** The synthetic nucleic acid dataset served as a successful positive control for regression models trained on both original and scrambled sequences. **(C)** To test the multi-class classification capabilities of BioAutoMATED, synthetic nucleic acids were split into four quartiles and labeled as “a”, “b”, “c”, and “d” (i.e., Class 0, 1, 2, and 3, respectively). The synthetic nucleic acid positive control was accurately classified with a multi-class classification model for both original and scrambled sequences. For panels **(A)**, **(B)**, and **(C)**, points correspond to one of N=3 folds. **(D)** A saliency map of the synthetic nucleic acid model and **(E)** its corresponding sequence logo validates that the adenine nucleotide, the lowest-scoring nucleotide, receives the highest model attention for N=100 randomly selected sequences. Normalized saliency represents arbitrary model attention units.

	Held out test set RBS sequences from Höllerer et al.[30]	**Held out test set peptide sequences from Liu et al.[31]**
Number of positives	13,836	207
Total number in dataset	27,654	471
R^2	0.867	0.435
Pearson R	0.931	0.659
Spearman R	0.879	0.678
auROC	0.939	0.868
Matthews Correlation Coefficient	0.725	0.523

Table S1: The top-performing models successfully validate on external datasets; Related to Figure 2 and Figure 3. The AutoKeras regression model trained on the ribosome binding site sequences obtained from Höllerer et al.[30] performs well on held-out test set sequences, displayed in the first column. In the second column, the top-performing TPOT regression model trained on the Liu et al.[31] peptide dataset successfully validates on the held-out test set. To obtain the auROC and MCC values, the target values were binarized on the median of the dataset.

	Free-trigger toehold switch sequences from Green et al.[47]	**Zika virus-sensing toehold switch sequences from Pardee et al.[48]**
Number of positives	42	N/A – ranked list
Total number in dataset	168	24
R^2	0.108	0.097
Pearson R	0.329	0.311
Spearman R	0.337	0.307
auROC	0.725	N/A

Table S2: The top-performing toehold switch models successfully validate on external datasets; Related to Figure 5. The top-performing AutoKeras toehold switch regression model performs equivalently to Valeri, Collins, Ramesh, et al.[29] for two external validation datasets, respectively: the 168 gold-standard free-trigger toehold switch sequences reported in Green et al.[47] and the 24 ranked toehold switches built to detect the Zika virus reported in Pardee et al.[48] Performance is evaluated with R^2 values, Pearson R coefficients, Spearman R coefficients, and area-under-ROC curve (auROC).

Dataset	Validation Set	DNABERT processing	k-mer	auROC			MCC		
				Bio Auto MATED	DNABERT		Bio Auto MATED	DNABERT	
					LR $2x10^{-4}$	LR $5x10^{-5}$		LR $2x10^{-4}$	LR $5x10^{-5}$
Ribosome binding sites (N=275,849)	External test set (N=27,654)	Binarized on median of train set (0.125)	k=3	0.915	0.502	0.961	0.659	0.0	0.811
			k=6		0.498	0.962		0.0	0.821
Toeholds (90%, N=82,380)	10% test set (N=9,154)	Binarized on median of dataset (0.446)	k=3	0.894	0.499	0.932	0.648	0.0	0.738
			k=6		0.503	0.932		0.0	0.740
Synthetic nucleic acids (90%, N=90,000)	10% test set (N=10,000)	Binarized on median of dataset (50)	k=3	1.0	0.505	1.0	1.0	0.0	1.0
			k=6		0.510	1.0		0.0	0.998

Table S4: BioAutoMATED and DNABERT[52] comparison on structured nucleic acid datasets; Related to Table 1. Generic DNABERT nucleic acid models were fine-tuned on RBS, toehold, and synthetic nucleic acid benchmark datasets as previously described. The k-mer size parameter influences the choice of pre-trained language model and the number of nucleotides being considered at a time by the model. Two different learning rates (LR) were evaluated: 2e-4 is the example learning rate given by DNABERT GitHub example and 5e-5 is the default learning rate within the DNABERT code.

Model	Cross Validation Results (N=3 folds)		External Validation Results (N=471 sequences)	
	R^2 on 80% folds during training	R^2 on 20% test set during training	Spearman R	R^2
ESM embeddings into K Neighbors Regressor	0.527 +/- 0.031	0.128 +/- 0.065	0.664	0.411
ESM embeddings into Random Forest Regressor	0.909 +/- 0.006	0.136 +/- 0.035	0.834	0.730
ESM embeddings into SVR	0.178 +/- 0.047	0.043 +/- 0.016	0.626	0.491
BioAutoMATED (best performing TPOT model)	0.425		0.678	0.434

Table S5: BioAutoMATED and ESM[53] comparison on a structured peptide dataset; Related to Table 1. Generic ESM protein models were used to generate embeddings that were then used as input into three types of scikit-learn models. The performance of the model depends on the architecture type.