

Selene: a PyTorch-based deep learning library for sequence data

Kathleen M. Chen ^{1,5}, Evan M. Cofer ^{2,3,5}, Jian Zhou ^{1,2} and Olga G. Troyanskaya ^{1,2,4*}

To enable the application of deep learning in biology, we present Selene (<https://selene.flatironinstitute.org/>), a PyTorch-based deep learning library for fast and easy development, training, and application of deep learning model architectures for any biological sequence data. We demonstrate on DNA sequences how Selene allows researchers to easily train a published architecture on new data, develop and evaluate a new architecture, and use a trained model to answer biological questions of interest.

Deep learning describes a set of machine learning techniques that use stacked neural networks to extract complicated patterns from high-dimensional data¹. These techniques are widely used for image classification and natural language processing, and have led to very promising advances in the biomedical domain, including genomics and chemical synthesis^{1–3}. In regulatory genomics, networks trained on high-throughput sequencing data (for example, ChIP-seq), or ‘sequence-based models’, have become the de facto standard for predicting the regulatory and disease impact of mutations^{4–7}. While deep-learning-related publications are often accompanied by the associated pre-trained model^{6,8,9}, a key challenge in both developing new deep learning architectures and training existing architectures on new data is the lack of a comprehensive, generalizable, and user-friendly deep learning library for biology.

Beyond regulatory genomics, sequence-level deep learning models have broad promise in a wide range of research areas, including recent advances on prediction of disease risk of missense mutations in proteins¹⁰ and potential applications to, for example, predicting target site accessibility in genome editing. We must enable the adoption and active development of deep-learning-based methods in biomedical sciences. For example, a biomedical scientist excited by a publication of a model capable of predicting the disease-associated effect of mutations should be able to train a similar model on their own ChIP-seq data focused on their disease of interest. A bioinformatician interested in developing new model architectures should be able to experiment with different architectures and evaluate all of them on the same data. Currently, this requires advanced knowledge specific to deep learning^{2,11}, substantial new code development, and associated time investment far beyond what most biomedical scientists are able to commit.

Here we present Selene, a framework for developing sequence-level deep learning networks that provides biomedical scientists with comprehensive support for model training, evaluation, and application across a broad range of biological questions. Sequence-level data refers to any type of biological sequence such as DNA, RNA, or protein sequences and their measured properties (for example, binding of transcription factors or RNA-binding proteins, or DNase sensitivity). Selene contains modules for (1) data sampling

and training for model development (Fig. 1a) and (2) prediction and visualization for analyses using the trained model (Fig. 1b,c). With Selene, researchers can run model development and analysis workflows out-of-the-box. For more advanced use cases, Selene provides templates for extending modules within each workflow so that users can adapt the library to their particular research questions.

There has been recent work to make deep learning in biology more accessible: DragoNN is a toolkit for teaching deep learning in regulatory genomics; pysster¹² is a Python package for training convolutional neural networks on biological sequence data; and Kipoi¹³ is a framework to archive, use, and build on published predictive models in genomics. These resources constitute the nascent software ecosystem for sequence-level deep learning. Selene is our contribution to this ecosystem. Selene supports general model development not constrained to a particular architecture (in contrast to pysster) or task (in contrast to DragoNN) and is designed for users with different levels of computational experience. Users are supported in tasks ranging from simply applying an existing model, to retraining it on new data (tasks also supported by Kipoi), to developing new model architectures (a task that is challenging to do with any other tool). The models developed using Selene can be shared and used through the Kipoi framework.

To demonstrate Selene’s capabilities for developing and evaluating sequence-level deep learning models, we use it to (1) train a published architecture on new data; (2) develop, train, and evaluate a new model (improving a published model); and (3) apply a trained model to data and visualize the resulting predictions in the case studies that follow.

In the first case study, a researcher wants to use the DeepSEA⁴ model architecture as a starting point and train the model on different data. Selene is completely general and a user can easily use or specify any model of their choice using modules in PyTorch.

Suppose a cancer researcher is interested in modeling the regulatory elements of the transcription factor GATA1, specifically focusing on proerythroblasts in bone marrow. This is a tissue-specific genomic feature that DeepSEA does not predict. The researcher downloads peak data from Cistrome¹⁴ and a reference genome FASTA file. Once a researcher formats the data to match the documented inputs¹⁵ and fills out the necessary training parameters (for example, batch size or learning rate), they can use Selene to train the DeepSEA architecture on their data with no new lines of Python code. In this example, they find that the model obtains an area under the curve (AUC) of 0.942 on this feature (Fig. 2a).

Selene automatically generates training, testing, and validation samples from the provided input data. The samples generated for each partition can be saved and used in subsequent model development so that comparisons can be made across models with different

¹Flatiron Institute, Simons Foundation, New York, NY, USA. ²Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ, USA.

³Graduate Program in Quantitative and Computational Biology, Princeton University, Princeton, NJ, USA. ⁴Department of Computer Science, Princeton University, Princeton, NJ, USA. ⁵These authors contributed equally: Kathleen M. Chen, Evan M. Cofer. *e-mail: ogt@cs.princeton.edu

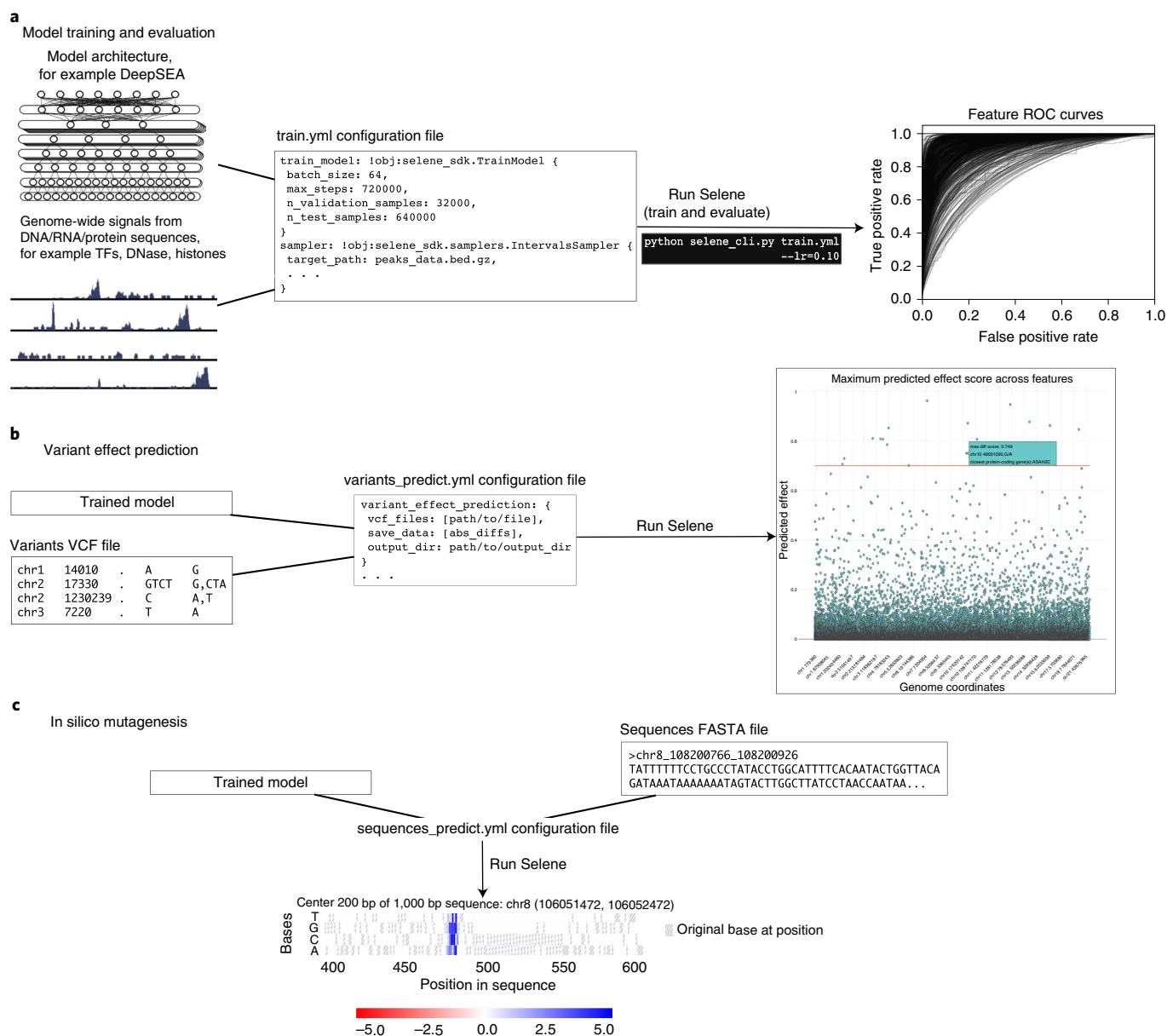


Fig. 1 | Overview of Selene. **a**, As input, the library accepts (left) the model architecture, dataset and (middle) a configuration file that specifies the necessary input data paths and training parameters. Selene automatically splits the data into training and validation/testing, trains the model, evaluates it, and (right) generates figures from the results. **b**, Selene supports variant effect prediction with the same configuration file format and includes functionality to visualize the variants and their difference scores as a Manhattan plot, where a user can hover over each point to see variant information. **c**, Selene calculates mutation effect scores and visualizes the scores as a heat map.

architectures and/or parameters. Further, Selene automatically evaluates the model on the test set after training and, in this case, generates figures to visualize the model's performance as receiver operating characteristic and average precision curves.

Now that the researcher has a trained model, they can use Selene to apply in silico mutagenesis—converting every position in the sequence to every other possible base⁴ (DNA and RNA) or amino acid (protein sequences)—to a set of GATA1 sequences drawn from the test set and examine the consequences of these 'mutations'. Selene supports visualization of the outputs of in silico mutagenesis as a heat map and/or motif plot. By visualizing the log₂ fold change for these sequences in a heat map, the researcher can see that the model detects disruptions in binding at the GATA motif (Fig. 2b).

We provide the code and results for this example in Selene's GitHub repository (<https://github.com/FunctionLab/selene>; see case 1 in the 'manuscript' folder).

In another use case, a researcher may want to develop and train a new model architecture. For example, a bioinformatician might want to modify a published model architecture to see how that affects performance. First, the researcher uses modules in PyTorch to specify the model architecture they are interested in evaluating; in this case study, they try to enhance the DeepSEA architecture with batch normalization and three additional convolutional layers. The researcher specifies parameters for training and the paths to the model architecture and data in a configuration file and passes this as input to the library's command-line interface (CLI). Training is automatically completed by Selene; afterward,

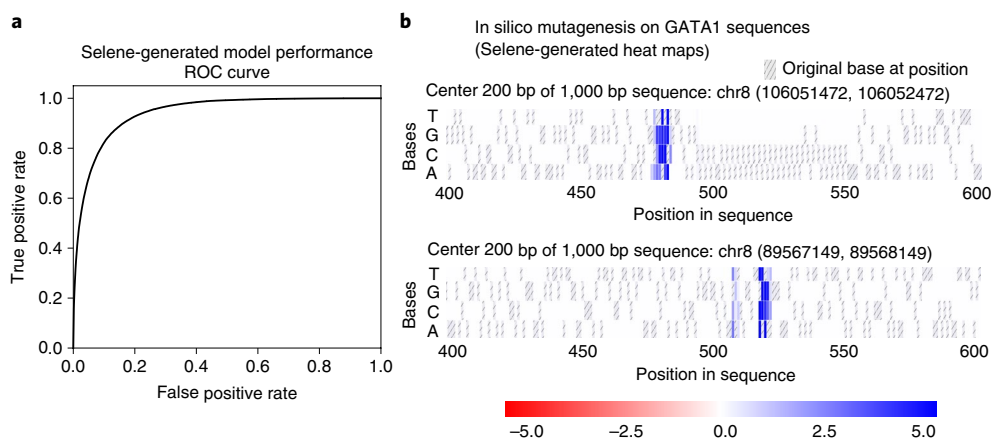


Fig. 2 | Visualizations generated by using Selene to train and apply a model to sequences. **a**, Selene visualization of the performance of the model trained in the first case study. **b**, Selene visualization of in silico mutagenesis on the case-study-trained model for 20 randomly selected GATA1 sequences in the test set (two representative plots displayed here; all heat maps generated are displayed in the example Jupyter notebook, https://github.com/FunctionLab/selene/blob/master/manuscript/case1/3_visualize_ism_outputs.ipynb). Bases in the original sequence are distinguished by the gray stripes in the heat map cells.

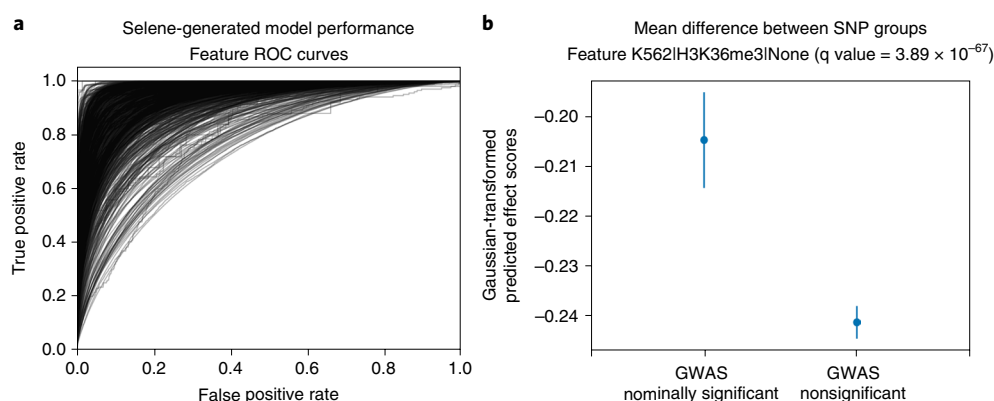


Fig. 3 | Using Selene to train a model and obtain model predictions for variants in an Alzheimer's GWAS study. **a**, Selene visualization of the performance of the trained six-convolutional-layer model. **b**, We visualize the mean and 95% confidence intervals of the quantile-normalized (against the Gaussian distribution) predicted effect scores of the two variant groups for the genomic feature H3K36me3 in K562 cells, the feature in the model with the most significant difference (one-sided Wilcoxon rank-sum test, adjusted P value using Benjamini-Hochberg of 3.89×10^{-67}). After applying the multiple testing correction, 914 of the 919 genomic features that the model predicts showed a significant difference ($\alpha < 0.05$) between the groups. SNP, single-nucleotide polymorphism.

the researcher can easily use Selene to compare the performance of their new model to the original DeepSEA model on the same chromosomal holdout dataset.

In this case study, the researcher finds that the deeper architecture achieves an average AUC of 0.938 (Fig. 3a) and an average area under the precision recall curve (AUPRC) of 0.362, which is an improvement over the average AUC of 0.933 and AUPRC of 0.342 of the original three-convolutional-layer model. The researcher can share this model with a collaborator and upload it to the Kipoi¹³ model zoo, a repository of trained models for regulatory genomics, with which Selene-trained models are fully compatible (an example is available in the GitHub repository at https://github.com/FunctionLab/selene/tree/master/manuscript/case2/3_kipoi_export).

In the final case study, a human geneticist studying Alzheimer's wants to apply the six-convolutional-layer model developed in the previous case study, so they first assess its ability to prioritize potential disease-associated variants. Specifically, they use Selene to make variant effect predictions for nominally significant variants ($P < 0.05$, $n = 422,398$) and non-significant variants ($P > 0.50$, $n = 3,842,725$) reported in the International Genomics of Alzheimer's Project¹⁶ Alzheimer's disease GWAS¹⁷. The

researcher finds that the predicted effect is significantly higher for GWAS nominally significant variants than for non-significant variants, indicating that the new model is indeed able to prioritize potential disease-associated variants (one-sided Wilcoxon rank-sum test; the most significant feature, H3K36me3 in K562 cells, has an adjusted P value, by Benjamini-Hochberg correction, of 3.89×10^{-67}) (Fig. 3b).

Selene's modeling capability extends far beyond the case studies described here. The library can be applied to not only DNA but also RNA and protein sequences, and not only chromatin data but any current genome-, transcriptome-, or even proteome-wide measurements. We developed Selene to increase the accessibility of deep learning in biology and facilitate the creation of reproducible workflows and results. Furthermore, Selene is open-source software that will continue to be updated and expanded on the basis of community and user feedback.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, statements of data availability and associated accession codes are available at <https://doi.org/10.1038/s41592-019-0360-8>.

Received: 8 October 2018; Accepted: 20 February 2019;
Published online: 28 March 2019

References

1. LeCun, Y., Bengio, Y. & Hinton, G. *Nature* **521**, 436–444 (2015).
2. Ching, T. et al. *J. R. Soc. Interface* **15**, 20170387 (2018).
3. Segler, M. H. S., Preuss, M. & Waller, M. P. *Nature* **555**, 604–610 (2018).
4. Zhou, J. & Troyanskaya, O. G. *Nat. Meth.* **12**, 931–934 (2015).
5. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. *Nat. Biotechnol.* **33**, 831–838 (2015).
6. Kelley, D. R., Snoek, J. & Rinn, J. L. *Genome Res.* **26**, 990–999 (2016).
7. Angermueller, C., Lee, H. J., Reik, W. & Stegle, O. *Genome. Biol.* **18**, 67 (2017).
8. Kelley, D. R. et al. *Genome Res.* **28**, 739–750 (2018).
9. Quang, D. & Xie, X. *Nucleic Acids Res.* **44**, e107 (2016).
10. Sundaram, L. et al. *Nat. Genet.* **50**, 1161–1170 (2018).
11. Min, S., Lee, B. & Yoon, S. *Brief. Bioinform.* **18**, 851–869 (2017).
12. Budach, S. & Marsico, A. *Bioinformatics* **34**, 3035–3037 (2018).
13. Avsec, Z. et al. *bioRxiv* Preprint at <https://www.biorxiv.org/content/10.1101/375345v1> (2018).
14. Mei, S. et al. *Nucleic Acids Res.* **45**, D658–D662 (2017).
15. Troyanskaya, O. G. et al. Selene CLI operations and outputs. *Selene* <https://selene.flatironinstitute.org/overview/cli.html> (2018).
16. Ruiz, A. et al. *Transl. Psychiatry* **4**, e358 (2014).
17. Huang, K.-L. et al. *Nat. Neurosci.* **20**, 1052–1061 (2017).

Acknowledgements

The authors acknowledge all members of the Troyanskaya lab for helpful discussions. In addition, the authors thank D. Simon for setting up the website and automating

updates to the site. The authors are pleased to acknowledge that this work was performed using the high-performance computing resources at Simons Foundation and the TIGRESS computer center at Princeton University. This work was supported by NIH grants R01HG005998, U54HL117798, R01GM071966, and T32HG003284; HHS grant HHSN272201000054C; and Simons Foundation grant 395506, all to O.G.T. O.G.T. is a CIFAR fellow.

Author contributions

K.M.C. and J.Z. conceived the Selene library. K.M.C. and E.M.C. designed, implemented, and documented Selene. K.M.C. performed the analyses described in the manuscript. O.G.T. supervised the project. K.M.C., E.M.C., and O.G.T. wrote the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at <https://doi.org/10.1038/s41592-019-0360-8>.

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence and requests for materials should be addressed to O.G.T.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© The Author(s), under exclusive licence to Springer Nature America, Inc. 2019

Methods

Overview of Selene. Selene consists of two components: a Python library for developing sequence-level neural networks, and a command-line interface (CLI) for prototypical use cases of the library (that is, training a new model, evaluating an existing model, and analyzing sequence data and variants with a trained model). We herein refer to these components as the software development kit (SDK) and the CLI, respectively. All functionality provided by the CLI is also available to the user through the SDK. Rather than supplanting the SDK, the CLI is intended to maximize code reuse and minimize user time spent learning SDK by heavily reducing the configuration tasks left to the user (for example, when GPU usage is specified, the CLI ensures all appropriate computations are performed on the GPU). When appropriate, the SDK does deliver functionality beyond that of the CLI. For instance, the SDK includes several data visualization methods that would be too unwieldy as executables run from the command line.

Thorough documentation for the SDK is available at <https://selene.flatironinstitute.org>, and tutorials for both the CLI and SDK can be found on the GitHub page (<https://github.com/FunctionLab/selene>). Notably, one tutorial demonstrates how to use Selene to train a deep neural network regression model (https://github.com/FunctionLab/selene/blob/master/tutorials/regression_mpra_example/regression_mpra_example.ipynb). This tutorial illustrates Selene's use outside of the models of transcriptional regulation shown in the case studies.

Selene software development kit. The Selene SDK, formally known as *selene_sdk*, is an extensible Python package intended to ease the development of new programs that leverage sequence-level models through code reuse. The Selene CLI is built entirely on the functionality provided by the SDK, but it is probable that users will use the SDK outside this context. For example, after training a new sequence-level model with the CLI, one could use the SDK in conjunction with a Python-based web application framework (e.g., Flask, Django) to build a web server so that other researchers can submit sequences or variants and get the trained model's predictions as output.

Leveraging the SDK in a user's Python project is no different from using any other Python module. That is, one only needs to import the *selene_sdk* module or any of its members and supply them with the correct parameters. The runtime behavior of each component of *selene_sdk*, as well as the required parameters for all members of *selene_sdk*, is described in detail in the online documentation (<https://selene.flatironinstitute.org/overview/overview.html>).

Selene CLI. The Selene CLI is a usable program to be run from the command line by the user. It encapsulates the configuration, execution, and logging of Selene's most common use cases. These use cases are embodied by the CLI's three commands: *train*, *evaluate* and *analyze*. These commands are used to train new models, evaluate the performance of trained models, and analyze model predictions (perform in silico mutagenesis or variant effect prediction), respectively. Each command configures its specific runtime environment with a combination of command line arguments and parameters drawn from user-provided configuration files. The flexibility of these configuration files allows them to leverage user-developed code as well, and further extends the usability of the CLI. We provide a step-by-step tutorial that describes the CLI configuration file format and shows some example configuration keys and values (https://github.com/FunctionLab/selene/blob/master/tutorials/getting_started_with_selene/getting_started_with_selene.ipynb); examples of CLI configuration code files are available at https://github.com/FunctionLab/selene/tree/master/config_examples. Finally, comprehensive documentation detailing all possible configurations supported by Selene can be found on Selene's documentation website (<https://selene.flatironinstitute.org/overview/cli.html>). Users can reference any of these resources when creating their own configuration files.

Model architectures. DeepSEA architecture used in case 1 (from the supplementary note in the DeepSEA publication⁴):

- (1) Convolutional layer (320 kernels; window size, 8; step size, 1)
- (2) Pooling layer (window size, 4; step size, 4)
- (3) Convolutional layer (480 kernels; window size, 8; step size, 1)
- (4) Pooling layer (window size, 4; step size, 4)
- (5) Convolutional layer (960 kernels; window size, 8; step size, 1)
- (6) Fully connected layer (919 genomic features)
- (7) Sigmoid output layer

Dropout proportion (proportion of outputs randomly set to 0):

- Layer 2: 20%
- Layer 4: 20%
- Layer 5: 50%
- All other layers: 0%

Architecture used in cases 2 and 3:

- (1) Convolutional layer (320 kernels; window size, 8; step size, 1)
- (2) Convolutional layer (320 kernels; window size, 8; step size, 1)
- (3) Pooling layer (window size, 4; step size, 4)
- (4) Convolutional layer (480 kernels; window size, 8; step size, 1)

- (5) Convolutional layer (480 kernels; window size, 8; step size, 1)
- (6) Pooling layer (window size, 4; step size, 4)
- (7) Convolutional layer (960 kernels; window size, 8; step size, 1)
- (8) Convolutional layer (960 kernels; window size, 8; step size, 1)
- (9) Fully connected layer (919 genomic features)
- (10) Sigmoid output layer

Dropout proportion:

- Layer 5: 20%
- Layer 8: 50%

Batch normalization applied after layers 2, 5, and 8 and before dropout.

Both architectures use the binary cross-entropy loss function and stochastic gradient descent optimizer (momentum, 0.9; weight decay, 10^{-6}).

Reproducing the case studies. Below, we have described the steps taken for each of the case studies. The code required to reproduce each case study is included in the GitHub repository (<https://github.com/FunctionLab/selene/tree/master/manuscript>) and was run with Selene version 0.2.0. We have also created Zenodo records for each case that contain all the input data, data processing scripts and output files generated from Selene:

- Case 1: <https://doi.org/10.5281/zenodo.1442433>
- Case 2: <https://doi.org/10.5281/zenodo.1442437>
- Case 3: <https://doi.org/10.5281/zenodo.1445555>

Case 1: training a state-of-the-art architecture on a different dataset. Steps to train DeepSEA on new data.

- (1) Download the data from Cistrome. In this case, we are only working with one dataset for one specific genomic feature. Cistrome ID 33545, measurements from GSM970258.
- (2) Format the data. We use tools from Samtools¹⁸ (specifically, *tabix*¹⁹ and *bgzip* from HTSLib, <https://www.htslib.org/>). Create a .bed file of chromosome, start, end and the genomic feature name (useful when there is more than one feature). Sort this file and compress it into a .gz file. Tabix index this file. Specific commands:
 - (i) Only use the columns [chr, start, end]: `cut -f 1-3 <peaks-file> > <peak-coordinates-file>`. Note: Eventually, we will add support for parsing BED files with strand specific features and/or continuous values that quantify these features
 - (ii) Add the genomic feature name as the fourth column of the file: `sed -i 's/$/t<feature-name>/' <peak-coordinates-file>`
 - (iii) Sort the file by [chr, start, end]: `sort -k1V -k2n -k3n <peak-coordinates-file> > <sorted-coordinates-file>`
 - (iv) Compress the file: `bgzip <sorted-coordinates-file>` This compresses the file to a .gz file in place. To separately generate the .gz file, run `bgzip -c <sorted-coordinates-file> > <sorted-coordinates-file>.gz`
 - (v) Tabix index the file: `tabix -p bed <sorted-coordinates-file>.gz`
- (3) Create a file of distinct features that the model will predict, where each feature is a single line in the file. This can easily be created from the .bed file in step 2 by running `cut -f 4 <peak-coordinates-file> | sort -u > <distinct-features>`.
- (4) Download the GRCh38/hg38 FASTA file. We downloaded the reference sequences GRCh37/hg19 and GRCh38/hg38 used in our analyses from ENCODE²⁰.
- (5) Specify the model architecture, loss and optimizer as a Python file. An example of this is available for DeepSEA at <https://github.com/FunctionLab/selene/blob/master/models/deepsea.py>.
- (6) Fill out the configuration file with the appropriate file paths and training parameters. We recommend starting from one of the example training files in the GitHub tutorials (for example, https://github.com/FunctionLab/selene/blob/master/tutorials/getting_started_with_selene/getting_started_with_selene.ipynb) or in the 'config_examples' directory (https://github.com/FunctionLab/selene/tree/master/config_examples). You can also review the documentation for the configuration parameters on Selene's website¹⁵.
- (7) Run Selene.

Steps to apply and visualize the results of in silico mutagenesis.

- (1) Collect sequences you want to visualize as a FASTA file. For this particular case, we provide a script to do so (https://github.com/FunctionLab/selene/blob/master/manuscript/case1/data/get_test_regions.py).
- (2) Fill out the configuration file with the appropriate file paths (for example, the path to the FASTA file and the trained model weights file).
- (3) Run Selene. You will get the raw predictions and the log₂ fold change scores as output files.

- (4) Follow one of the Jupyter notebook tutorials for in silico mutagenesis (<https://github.com/FunctionLab/selene/tree/master/tutorials>) to generate visualizations for the sequences. We have done this at https://github.com/FunctionLab/selene/blob/master/manuscript/case1/3_visualize_ism_outputs.ipynb.

Case 2: developing a new architecture and making model comparisons. Steps to train 'deeper DeepSEA' on the same exact data as DeepSEA.

- (1) Download the code and data bundle from the DeepSEA website (http://deepsea.princeton.edu/media/code/deepsea_train_bundle.v0.9.tar.gz). You only need the .mat files in this directory. We also include a file listing the 919 genomic features that the model predicts. This is from the resources directory in the standalone version of DeepSEA (<http://deepsea.princeton.edu/media/code/deepsea.v0.94b.tar.gz>). Zenodo record: https://zenodo.org/record/2214970/files/DeepSEA_data.tar.gz.
- (2) Fill out the configuration file for Selene's MultiFileSampler (<https://selene.flatironinstitute.org/overview/cli.html#multiple-file-sampler>) and specify the path to each .mat file for training, validation and testing.
- (3) Run Selene.

Please see the DeepSEA publication⁴ for details about data processing and training.

In the main text, we report test performance for the model trained using the online sampler. When training on the same exact data (the .mat files) as DeepSEA, we achieve an average AUC of 0.934 and an average AUPRC of 0.361.

Steps to download and format all the peak data from ENCODE and Roadmap Epigenomics.

- (1) Download all chromatin feature profiles used for training DeepSEA, specified in Supplementary Table 1 of the DeepSEA manuscript⁴ (https://zenodo.org/record/2214970/files/chromatin_profiles.tar.gz).
- (2) For each file, keep the chromosome, start, and end columns. In addition, create a fourth column with the feature's name. Concatenate all these files and create the distinct features file. We provide a Python script for this step on the GitHub page (https://github.com/FunctionLab/selene/blob/master/manuscript/case2/1_train_with_online_sampler/data/process_chromatin_profiles.py).
- (3) Format the data according to the instructions in the 'Getting started' tutorial:
 - i. Sort the file by [chr, start, end]: `sort -k1V -k2n -k3n <peak-coordinates-file> > <sorted-coordinates-file>`.
 - ii. Compress the file: `bgzip <sorted-coordinates-file>`. This compresses the file to a .gz file in place. To separately generate the .gz file, run `bgzip -c <sorted-coordinates-file> > <sorted-coordinates-file>.gz`.
 - iii. Tabix index the file: `tabix -p bed <sorted-coordinates-file>.gz`.

A script containing these steps can be downloaded from https://github.com/FunctionLab/selene/blob/master/manuscript/case2/1_train_with_online_sampler/data/process_data.sh.

- (4) Download the hg19 FASTA file (<https://www.encodeproject.org/files/male.hg19/@download/male.hg19.fasta.gz>).
- (5) Specify the model architecture, loss, and optimizer as a Python file: https://github.com/FunctionLab/selene/blob/master/selene_sdk/utis/example_model.py.

- (6) Fill out the configuration file with the appropriate file paths and training parameters. We set the training parameters (number of steps, batches and so on) so that they matched how DeepSEA was originally trained.
- (7) Run Selene.

Case 3: applying a new model to variants.

- (1) Download the single-nucleotide polymorphisms from the International Genomics of Alzheimer's Project. (<https://www.niagads.org/igap-age-onset-survival-analyses-p-value-only>).
- (2) Group the variants into those with *P* values below 0.05 (significant) and those with *P* values above 0.50 (non-significant).
- (3) Fill out the configuration file with the paths to the two variant files and the trained model weights file from case 2.
- (4) Run Selene.
- (5) Follow the script provided for this case to analyze the variant predictions (https://github.com/FunctionLab/selene/blob/master/manuscript/case3/2_variant_groups_comparison.sh).

Statistical analysis. Details of the statistical test used for case study 3 are specified in the associated text and figure legend (Fig. 3b).

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Code availability

Selene is open-source software (license BSD 3-Clause Clear). Project homepage: <https://selene.flatironinstitute.org>. GitHub: <https://github.com/FunctionLab/selene>. Archived version: <https://github.com/FunctionLab/selene/archive/0.2.0.tar.gz>.

Data availability

Cistrome¹⁴, Cistrome file ID 33545, measurements from GSM970258: <http://dc2.cistrome.org/api/downloads/eyJpZCI6IjMzNTQ1In0%3A1fujCu%3ArNvWLCNoET6o9SdK8fEv13uRu4b/>. ENCODE²¹ and Roadmap Epigenomics²² chromatin profiles: files listed in Supplementary Table 1 of ref. ⁴. IGAP age at onset survival^{16,17}: <https://www.niagads.org/datasets/ng00058> (*P*-values-only file). The case studies used processed datasets from these sources. They can be downloaded at the following Zenodo links: Cistrome, <https://zenodo.org/record/2214130/files/data.tar.gz>; ENCODE and Roadmap Epigenomics chromatin profiles, https://zenodo.org/record/2214970/files/chromatin_profiles.tar.gz; IGAP age at onset survival, https://zenodo.org/record/1445556/files/variant_effect_prediction_data.tar.gz. Source data for Figs. 2 and 3 are available online.

References

18. Li, H. et al. *Bioinformatics* **25**, 2078–2079 (2009).
19. Li, H. *Bioinformatics* **27**, 718–719 (2011).
20. ENCODE Project. Reference sequences. *ENCODE: Encyclopedia of DNA Elements* <https://www.encodeproject.org/data-standards/reference-sequences/> (2016).
21. ENCODE Project Consortium. *Nature* **489**, 57–74 (2012).
22. Kundaje, A. et al. *Nature* **518**, 317–330 (2015).

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see [Authors & Referees](#) and the [Editorial Policy Checklist](#).

Statistical parameters

When statistical analyses are reported, confirm that the following items are present in the relevant location (e.g. figure legend, table legend, main text, or Methods section).

n/a Confirmed

- ☐ ☒ The exact sample size (n) for each experimental group/condition, given as a discrete number and unit of measurement
- ☒ ☐ An indication of whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
- ☐ ☒ The statistical test(s) used AND whether they are one- or two-sided
Only common tests should be described solely by name; describe more complex techniques in the Methods section.
- ☐ ☒ A description of all covariates tested
- ☐ ☒ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
- ☐ ☒ A full description of the statistics including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
- ☒ ☐ For null hypothesis testing, the test statistic (e.g. F , t , r) with confidence intervals, effect sizes, degrees of freedom and P value noted
Give P values as exact values whenever suitable.
- ☒ ☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
- ☒ ☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
- ☒ ☐ Estimates of effect sizes (e.g. Cohen's d , Pearson's r), indicating how they were calculated
- ☐ ☒ Clearly defined error bars
State explicitly what error bars represent (e.g. SD, SE, CI)

Our web collection on [statistics for biologists](#) may be useful.

Software and code

Policy information about [availability of computer code](#)

Data collection

No software was used to collect data in this study (that is, no data was collected).

Data analysis

Project homepage: <https://selene.flatironinstitute.org>
 GitHub: <https://github.com/FunctionLab/selene>
 Archived version: <https://github.com/FunctionLab/selene/archive/0.2.0.tar.gz>

Additional software used:
 Samtools (version 1.9). Specifically, tabix and bgzip in the HTSlib (version 1.9) package.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers upon request. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research [guidelines for submitting code & software](#) for further information.

Data

Policy information about [availability of data](#)

All manuscripts must include a [data availability statement](#). This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

Data sources

Cistrome

Cistrome file ID: 33545, measurements from GSM970258 (Xu et al., 2012)

<http://dc2.cistrome.org/api/downloads/eyJpZCI6IjMzNTQ1In0%3A1fujCu%3ArNvWLCNoET6o9SdKl8fEv13uRu4b/>

ENCODE and Roadmap Epigenomics chromatin profiles

Files listed in <https://media.nature.com/original/nature-assets/nmeth/journal/v12/n10/extref/nmeth.3547-S2.xlsx>

IGAP age at onset survival

<https://www.niagads.org/datasets/ng00058> (p-values only file)

Processed datasets from these sources are available at the following Zenodo links:

Cistrome:

<https://zenodo.org/record/2214130/files/data.tar.gz>

ENCODE and Roadmap Epigenomics chromatin profiles:

https://zenodo.org/record/2214970/files/chromatin_profiles.tar.gz

IGAP age at onset survival:

https://zenodo.org/record/1445556/files/variant_effect_prediction_data.tar.gz

Field-specific reporting

Please select the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see [nature.com/authors/policies/ReportingSummary-flat.pdf](https://www.nature.com/authors/policies/ReportingSummary-flat.pdf)

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	Only previously published data were used.
Data exclusions	Only previously published data were used.
Replication	This study does not present experimental findings.
Randomization	This study does not present experimental findings.
Blinding	We did not need blinding since we had no randomized control trials.

Reporting for specific materials, systems and methods

Materials & experimental systems

n/a	Involved in the study
☒	☐ Unique biological materials
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☒	☐ Animals and other organisms
☒	☐ Human research participants

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging