


SHORT REPORT

Open Access



# EvoAug: improving generalization and interpretability of genomic deep neural networks with evolution-inspired data augmentations

Nicholas Keone Lee<sup>1</sup>, Ziqi Tang<sup>1</sup>, Shushan Toneyan<sup>1</sup> and Peter K. Koo<sup>1\*</sup> 

\*Correspondence:  
koo@cshl.edu

<sup>1</sup> Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, 1 Bungtown Road, Cold Spring Harbor, NY, USA

## Abstract

Deep neural networks (DNNs) hold promise for functional genomics prediction, but their generalization capability may be limited by the amount of available data. To address this, we propose EvoAug, a suite of evolution-inspired augmentations that enhance the training of genomic DNNs by increasing genetic variation. Random transformation of DNA sequences can potentially alter their function in unknown ways, so we employ a fine-tuning procedure using the original non-transformed data to preserve functional integrity. Our results demonstrate that EvoAug substantially improves the generalization and interpretability of established DNNs across prominent regulatory genomics prediction tasks, offering a robust solution for genomic DNNs.

**Keywords:** Deep learning, Regulatory genomics, Data augmentations, Model interpretability

## Background

Uncovering *cis*-regulatory elements and their coordinated interactions is a major goal of regulatory genomics. Deep neural networks (DNNs) offer a promising avenue to learn these genomic features *de novo* through being trained to take DNA sequences as input and predict their regulatory functions as output [1–3]. Following training, these DNNs have been employed to score the functional effect of disease-associated variants [4, 5]. Moreover, post hoc model interpretability methods have revealed that DNNs base their decisions on learning sequence motifs of transcription factor (TF) binding sites and dependencies with other TFs and sequence context [6–10].

For DNNs, generalization typically improves with more training data. However, the amount of data generated in a high-throughput functional genomics experiment is fundamentally limited by the underlying biology. For example, the extent to which certain



© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

TFs bind to DNA is constrained by the availability of high-affinity binding sites in accessible chromatin.

To expand a finite dataset, data augmentations can provide additional variations on existing training data [11, 12]. Data augmentations act as a form of regularization, guiding the learned function to be invariant to symmetries created by the data transformations [13, 14]. This approach can help prevent a DNN from overfitting to spurious features and improve generalization [15]. The main challenge with data augmentations in genomics is quantifying how the regulatory function changes for a given transformation. With image data, basic affine transformations can translate, magnify, or rotate an image without changing its label. However, in genomics, the available neutral augmentations are reverse-complement transformation [16] and small random translations of the input sequence [17, 18]. With the finite size of experimental data and a paucity of augmentation methods, strategies to promote generalization for genomic DNNs are limited.

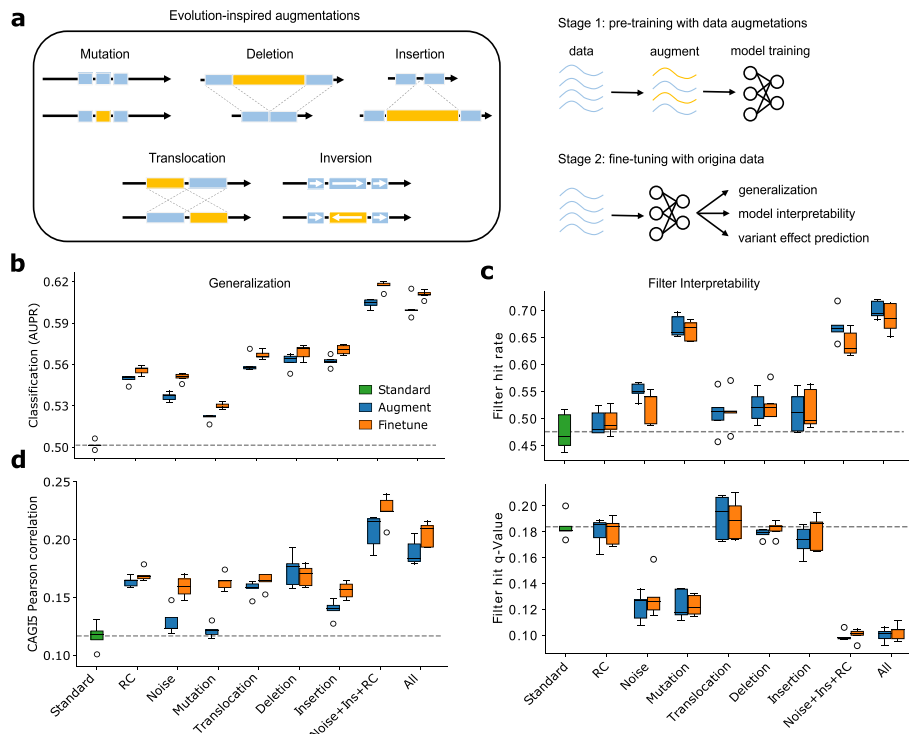
Here, we introduce EvoAug, an open-source PyTorch package that provides a suite of evolution-inspired data augmentations. We show that training DNNs with EvoAug leads to better generalization performance and improves efficacy with standard post hoc explanation methods, including filter interpretability and attribution analysis, across prominent regulatory genomics prediction tasks for well-established DNNs.

## Results and discussion

### Evolution-inspired data augmentations for sequence-based genomic DNNs

To enhance the effectiveness of sequence-based models, data augmentations should aim to increase genetic diversity while maintaining the same biological functionality. Evolution provides a natural process to generate genetic variability, including random mutations, deletions, insertions, inversions, and translocations, among others [19]. However, these genetic changes often have functional consequences that expand phenotypic diversity and aid in natural selection. While the addition of homologous sequences to a dataset could achieve the goal of increasing sequence diversity while preserving biological function, identifying regulatory regions with similar functions throughout the genomes across species is difficult. Alternatively, synthetic perturbations that do not alter the function can be applied, but it is crucial to have prior knowledge to ensure that features such as motifs and their dependencies are not affected. Therefore, formulating new data augmentation strategies for genomics remains a significant challenge.

In this study, we present a suite of evolution-based data augmentations and a two-stage training curriculum to preserve functional integrity (Fig. 1a). In the first stage, a DNN is trained on sequences with EvoAug augmentations applied stochastically online during training, using the same training labels as the wild-type sequence. The goal is to enhance the model's ability to learn robust representations of features, such as motifs, by exposing it to expanded (albeit synthetically generated) genetic variation. While each augmentation has the potential to disrupt core motifs in any given perturbation, we expect the overall effect to preserve motifs on average. However, the specific data augmentations employed may introduce a bias in how these motif grammars are structured. Thus, in the second stage, the DNN is fine-tuned on the original, unperturbed data to refine these features and guide the function towards the observed biology, thereby removing any bias introduced by the data augmentations (see Methods).



**Fig. 1** EvoAug improves generalization and interpretability of Basset models. **a** Schematic of evolution-inspired data augmentations (left) and the two-stage training curriculum (right). **b** Generalization performance (area under the precision-recall curve) for Basset models pretrained with individual and combinations of augmentations, i.e., Noise+Ins+RC (Gaussian noise, insertion, reverse-complement) and all augmentations (Gaussian noise, reverse-complement, mutation, translocation, deletion, insertion), and fine-tuned on Basset dataset. Standard represents no augmentations during training. **c** Comparison of the average hit rate of first-layer filters to known motifs in the JASPAR database (top) and the average  $q$ -value of the filters with matches (bottom). **d** Comparison of the average Pearson correlation between model predictions and experimental data from CAGIS Challenge. **b–d** Each box-plot represents 5 trials with random initializations

EvoAug data augmentations introduce a modeling bias to learn invariances of the (un) natural symmetries generated by the augmentations. For instance, random insertions and deletions assume that the distance between motifs is not critical, whereas random inversions and translocations promote invariances to motif strand orientation and the order of motifs, respectively. Nevertheless, the bias created by the augmentations can lead to poor generalization when the introduced bias does not accurately reflect the underlying biology. Therefore, the fine-tuning stage is critical as it provides an avenue to unlearn any biases not supported by the observed data.

### EvoAug improves generalization and interpretability of genomic DNNs

To demonstrate the utility of EvoAug, we analyzed several established DNNs across three prominent types of regulatory genomic prediction tasks that span a range of complexity.

First, we applied EvoAug to the Basset model and dataset [20], which consists of a multi-task binary classification of chromatin accessibility sites across 161 cell types/tissues. We trained the Basset model with each augmentation applied independently and

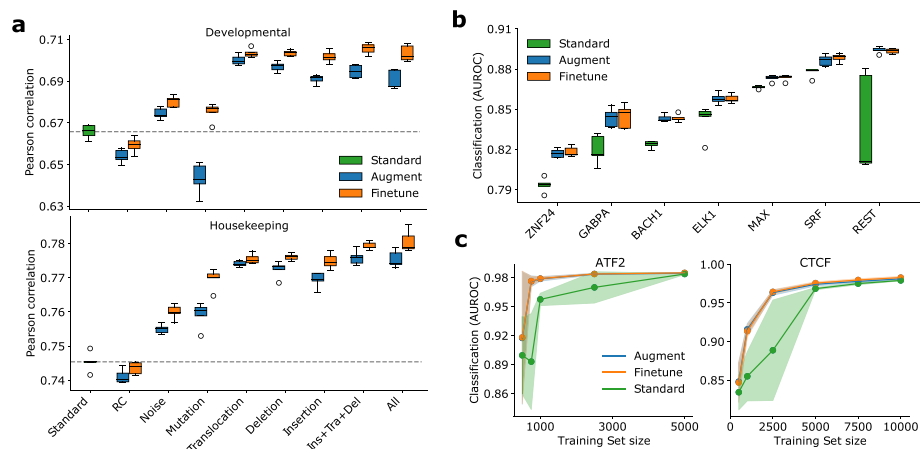
in various combinations. We conducted a hyperparameter sweep to determine the optimal settings for each augmentation (Additional file 1: Figs. S1-S5). From hyperparameter sweeps, we observed that the inversion augmentation improved performance up to the sequence length, which is essentially a reverse-complement transformation (Additional file 1: Figs. S1, S3, and S4). Hence, inversions were excluded to reduce redundancy.

Remarkably, EvoAug-trained DNNs outperformed standard training with no augmentations (Fig. 1b). The best results were achieved when multiple augmentations were used together. Additionally, we found that fine-tuning on the original data further improved performance, even when augmentation hyperparameters were poorly specified (Additional file 1: Fig. S1). Notably, specific EvoAug augmentations, such as random mutations and combinations of data augmentations, had a profound impact on improving the motif representations learned by the first-layer convolutional filters (Fig. 1c). The convolutional filters capture a wider repertoire of motifs and their representations better reflect known motifs, both quantitatively and qualitatively, when compared with convolutional filters of models trained without augmentations. This suggests that EvoAug augmentations can help DNNs learn more accurate and informative representations of the sequence motifs.

A major downstream application of genomic DNNs is to score the functional consequences of non-coding mutations. By evaluating the zero-shot prediction capabilities of each DNN on saturation mutagenesis data of 15 *cis*-regulatory elements from the CAGI5 Challenge [21], we found that models trained with EvoAug outperformed their standard training counterpart (Fig. 1d). Notably, Basset's performance was comparable to other DNNs based on binary predictions [17]; however, its overall performance was lower than more sophisticated DNNs and top competitors in the CAGI5 challenge [2]. Interestingly, we observed that DNNs pretrained with Gaussian noise or random mutagenesis augmentations did not perform well. These augmentations impose flatness locally in sequence-function space, effectively reducing the effect size of nucleotide variants. However, fine-tuning these models improved their variant effect predictions beyond what was achieved with standard training, thus demonstrating the effectiveness of the two-stage training curriculum.

To further demonstrate the benefits of EvoAug, we trained DeepSTARR models as a multi-task quantitative regression to predict enhancer activity from self-transcribing active regulatory region sequencing (STARR-seq) data [9], where each task represents a different promoter from a developmental or housekeeping gene in *Drosophila* S2 cells. Most EvoAug augmentations resulted in improved performance, except for reverse-complement and random mutations (Fig. 2a and Additional file 1: Figs. S3-S5). As before, we observed additional performance gains when augmentations were used in combination. Furthermore, the attribution maps generated by EvoAug-trained models were more interpretable, with identifiable motifs and less spurious noise (Additional file 1: Fig. S6).

In addition, we found that the EvoAug-trained DNNs consistently outperformed DNNs with standard training on various single-task binary classification tasks for TF binding across multiple chromatin immunoprecipitation sequencing (ChIP-seq) datasets (Fig. 2b). Interestingly, we did not observe any significant improvement in performance after fine-tuning, suggesting that the implicit prior imposed by EvoAug augmentations



**Fig. 2** Generalization of EvoAug on additional models and datasets. **a** Box-plot of regression performance for DeepSTARR models pretrained with individual or combination of augmentations (i.e., insertion + translocation + deletion; all augmentations) and fine-tuned on original STARR-seq data for two promoters: developmental (top) and housekeeping (bottom). Standard represents no augmentations during training. **b** Box-plot of classification performance (area under the receiver-operating-characteristic curve) for DNNs trained on CHIP-seq data. **c** Average classification performance for CHIP-seq experiments downsampled to different dataset sizes. Shaded region represents the standard deviation of the mean. **a, b** Each box-plot represents 5 trials with random initializations

was appropriate for these tasks; the underlying regulatory grammars for these TFs are not complex.

To further investigate the impact of EvoAug on small datasets, we retrained each DNN on down-sampled versions of two abundant CHIP-seq datasets. We found that EvoAug-trained DNNs exhibit a greater improvement in performance for smaller datasets compared to standard training (Fig. 2c). This result suggests that EvoAug can be particularly useful in scenarios where the available training data is limited.

Training with EvoAug adds a computational cost, depending on the augmentations chosen and their settings (Additional file 1: Tables S2 and S3). Nevertheless, EvoAug stabilized training (Additional file 1: Fig. S7), leading to smoother convergence and improved generalization overall.

## Conclusion

EvoAug greatly expands the set of available data augmentations for genomic DNNs. Our study demonstrated that EvoAug's two-stage training curriculum is effective in improving generalization performance. Moreover, EvoAug-trained models learned better representations of consensus motifs, as evidenced by filter visualization and attribution analysis.

Our findings support previous arguments for using evolution as a natural source of data augmentation [22]. Interestingly, the impact of synthetic evolutionary perturbations was not excessively disruptive, and performance even improved before fine-tuning in most cases. This functional robustness appears to be a characteristic of the non-coding genome [23].

Data augmentations are a commonly used technique to balance bias and variance in machine learning models. However, their effectiveness is expected to decrease as the dataset size increases. Nevertheless, EvoAug still improved performance on the already

large Basset dataset. Other methods that can enhance generalization include multitask learning [24], contrastive learning [25, 26], and language modeling [27]. Even though Basset and DeepSTARR are already trained in a multitask framework, EvoAug improved their performance. Multitasking can introduce class imbalance, but EvoAug provides additional examples with pseudo-positive labels, which can mitigate this issue. EvoAug also provides different views of the data, which can be useful for contrastive learning. Importantly, EvoAug is a lightweight and effective strategy that only requires the original data.

The optimal combination of augmentations and their hyperparameter choices depends on the model and dataset. While we performed hyperparameter grid searches in this study, more advanced search strategies such as population-based training [28] using Ray Tune [29] could improve efficiency. In the future, we plan to investigate EvoAug's potential in cross-dataset generalization and variant effect predictions, including expression quantitative trait loci.

EvoAug is a PyTorch package that is open-source [30], easy to use, extensible, and accessible via pip (<https://pypi.org/project/evoaug>) and GitHub (<https://github.com/pkoo/evoaug>), with full documentation provided on ReadtheDocs.org (<https://evoaug.readthedocs.io>). In time, we plan to extend EvoAug functionality to TensorFlow [30] and JAX [31]. We anticipate that EvoAug will have broad utility in improving the efficacy of sequence-based DNNs for regulatory genomics.

## Methods

### Models and datasets

#### *Basset*

The Basset dataset [20] consists of a multi-task binary classification of chromatin accessibility sites across 161 cell types/tissues. The inputs are genomic sequences of length 600 nt and the output are binary labels (representing accessible or not accessible) for 161 cell types measured experimentally using DNase I hypersensitive sites sequencing (DNase-seq). We filtered sequences that contained at least one N character and the data splits (training; validation; test) reduced from (1,879,982; 70,000; 71,886) to (437,478; 16,410; 16,703). This “cleaned” dataset [32] was analyzed using a Basset-inspired model, which is given according to:

- Input  $x \in \{0, 1\}^{600 \times 4}$  (one-hot encoding of 600 nt sequence)
- 1D convolution (300 filters, size 19, stride 1)
- BatchNorm + ReLU
- Max-pooling (size 3, stride 3)
- 1D convolution (200 filters, size 11, stride 1)
- BatchNorm + ReLU
- Max-pooling (size 4, stride 4)
- 1D convolution (200 filters, size 7, stride 1)
- BatchNorm + ReLU
- Max-pooling (size 2, stride 2)
- Fully-connected (1000 units)
- BatchNorm + ReLU

- Dropout (0.3)
- Fully-connected (1000 units)
- BatchNorm + ReLU
- Dropout (0.3)
- Fully-connected output (161 units, sigmoid)

BatchNorm represents batch normalization [33], and dropout [34] rates set the probability that neurons in a given layer are temporarily removed during each mini-batch of training.

### **DeepSTARR**

The DeepSTARR dataset [9] consists of a multi-task regression of enhancer activity for two promoters, well-known developmental and housekeeping transcriptional programs in *D. melanogaster* S2 cells. The inputs are genomic sequences of length 249 nt and the output is 2 scalar values representing the activity of developmental enhancers and housekeeping enhancers measured experimentally using STARR-seq. Sequences with N characters were also removed, but this minimally affected the size of the dataset (i.e., reduced it by approximately 0.005%). This dataset [32] was analyzed using the original DeepSTARR model, given according to:

- Input  $x \in \{0, 1\}^{249 \times 4}$
- 1D convolution (256 filters, size 7, stride 1)
- BatchNorm + ReLU
- Max-pooling (size 2, stride 2)
- 1D convolution (60 filters, size 3, stride 1)
- BatchNorm + ReLU
- Max-pooling (size 2, stride 2)
- 1D convolution (60 filters, size 5, stride 1)
- BatchNorm + ReLU
- Max-pooling (size 2, stride 2)
- 1D convolution (120 filters, size 3, stride 1)
- BatchNorm + ReLU
- Max-pooling (size 2, stride 2)
- Fully-connected (256 units)
- BatchNorm + ReLU
- Dropout (0.4)
- Fully-connected (256 units)
- BatchNorm + ReLU
- Dropout (0.4)
- Fully-connected output (2 units, linear)

### **ChIP-seq**

Transcription factor (TF) chromatin immunoprecipitation sequencing (ChIP-seq) data was processed and framed as a binary classification task. The inputs are genomic

sequences of length 200 nt and the output is a single binary label representing TF binding activity, with positive-label sequences indicating the presence of a ChIP-seq peak and negative-label sequences indicating a peak for a DNase I hypersensitive site from the same cell type but one that does not overlap with any ChIP-seq peaks. Nine representative TF ChIP-seq experiments in a GM12878 cell line and a DNase-seq experiment for the same cell line were downloaded from ENCODE [35]; for details, see Additional file 1: Table S1. Negative sequences (i.e., DNase-seq peaks that do not overlap with any positive peaks) were randomly down-sampled to match the number of positive sequences, keeping the classes balanced. The dataset was split randomly into training, validation, and test set according to the fractions 0.7, 0.1, and 0.2, respectively [32].

A custom convolutional neural network was employed to analyze these datasets, given according to:

- Input  $x \in \{0, 1\}^{200 \times 4}$
- 1D convolution (64 filters, size 7, stride 1)
- BatchNorm + ReLU
- Dropout (0.2)
- Max-pooling (size 4, stride 4)
- 1D convolution (96 filters, size 5, stride 1)
- BatchNorm + ReLU
- Dropout (0.2)
- Max-pooling (size 4, stride 4)
- 1D convolution (128 filters, size 5, stride 1)
- BatchNorm + ReLU
- Dropout (0.2)
- Max-pooling (size 2, stride 2)
- Fully-connected layer (256 units)
- BatchNorm + ReLU
- Dropout (0.5)
- Fully-connected output layer (1 unit, sigmoid)

### Evolution-inspired data augmentations

EvoAug is comprised of a set of data augmentations given by the following:

- Mutation: a transformation where single nucleotide mutations are randomly applied to a given wild-type sequence. This is implemented as follows: (1) given the hyperparameter of the fraction of nucleotides in each sequence to mutate (`mutate_frac`), the number of mutations for a given sequence length is calculated; (2) a position along the sequence is randomly sampled (with replacement) for each number of mutations; and (3) the selected positions are mutagenized to a random nucleotide. Since our implementation does not guarantee that a nucleotide selected will be mutated to a different nucleotide than it originally was, we take approximate account for silent mutations by dividing the user-defined



`mutate_frac` by 0.75 so that on average the fraction of nucleotides in each sequence mutated to a different nucleotide is equal to `mutate_frac`.

- **Translocation:** a transformation that randomly selects a break point in the sequence (thereby creating two segments) and then swaps the order of the two sequence segments. An equivalent statement of this transformation is a “roll”—shifting the sequence forward along its length a randomly specified distance and then reintroducing the part of the sequence shifted beyond the last position back at the first position. This is implemented as follows: (1) given the hyperparameters of the minimum distance (`shift_min`, default 0) and maximum distance (`shift_max`) of the shift, the integer-valued shift length is chosen randomly from the interval  $[-\text{shift\_max}, -\text{shift\_min}] \cup [\text{shift\_min}, \text{shift\_max}]$ , where a negative value simply denotes a backward shift rather than a forward shift, and (2) the shift is applied to the sequence with a `roll()` function in PyTorch.
- **Insertion:** a transformation where a random DNA sequence (of random length) is inserted randomly into a wild-type sequence. This is implemented as follows: (1) given the hyperparameters of the minimum length (`insert_min`, default 0) and maximum length (`insert_max`) of the insertion, the integer-valued insertion length is chosen randomly from the interval between `insert_min` and `insert_max` (inclusive), and (2) the insertion is inserted at a random position within the original sequence. Importantly, to maintain a constant input sequence length to the model (i.e., original length plus `insert_max`), the remaining amount of length between the insertion length and `insert_max` is split evenly and placed on the 5' and 3' flanks of the sequence, with the remainder from odd lengths going to the 3' end. Whenever an insertion augmentation is employed in combination with other augmentations, all sequences without an insertion are padded with a stretch of random DNA of length `insert_max` at the 3' end to ensure that the model processes sequences with a constant length for both training and inference time.
- **Deletion:** a transformation where a random, contiguous segment of a wild-type sequence is removed, and the shortened sequence is then padded with random DNA sequence to maintain the same length as wild-type. This is implemented as follows: (1) given the hyperparameters of the minimum length (`delete_min`, default 0) and maximum length (`delete_max`) of the deletion, the integer-valued deletion length is chosen randomly from the interval between `delete_min` and `delete_max` (inclusive); (2) the starting position of the deletion is chosen randomly from the valid positions in the sequence that can encapsulate the deletion; (3) the deletion is performed on the designated stretch of the sequence; (4) the remaining portions of the sequence are concatenated together; and (5) random DNA is used to pad the 5' and 3' flanks to maintain a constant input sequence length, similar to the procedure for insertions.
- **Inversion:** a transformation where a random subsequence is replaced by its reverse-complement. This is implemented as follows: (1) given the hyperparameters of the minimum length (`invert_min`, default 0) and maximum length (`invert_max`) of the inversion, the integer-valued inversion length is chosen randomly from the interval between `invert_min` and `invert_max` (in-

sive); (2) the starting position of the inversion is chosen randomly from the valid position indices in the sequence; and (3) the inversion (i.e., a reverse-complement transformation) is performed on the designated subsequence while the remaining portions of the sequence remain untouched.

- Reverse-complement: a transformation where a full sequence is replaced with some probability `rc_prob` by its reverse-complement.
- Gaussian noise: a transformation where Gaussian noise (with distribution parameters `noise_mean = 0` and `noise_std`) is added to the input sequence; a random value drawn independently and identically from the specified distribution is added to each element of the one-hot input matrix.

### ***Pretraining with data augmentations***

Training with augmentations requires two main hyperparameters: first, a set of augmentations to sample from; and second, the maximum number of augmentations to be applied to a sequence. For each mini-batch during training, each sequence is randomly augmented independently. The number of augmentations to be applied to a given sequence has two possible settings in EvoAug: (1) `hard`, always equal to the maximum number of augmentations, or (2) `soft`, randomly select the number of augmentations for a sequence from 1 to the maximum number. Our experiments with Basset and DeepSTARR use the former setting, while our experiments with ChIP-seq datasets use the latter setting. Then, the subset of augmentations to be applied to the sequence is sampled randomly without replacement from the user-defined set of augmentations. After a subset of augmentations is chosen, the order in which multiple augmentations are applied to a single sequence is given by the following priority: inversion, deletion, translocation, insertion, reverse-complement, mutation, noise addition. Each augmentation is then applied stochastically for each sequence.

For the Basset and DeepSTARR models, each augmentation has an optimal setting that was determined from a hyperparameter search independently using the validation set (Additional file 1: Figs. S1, S3, and S4). For the Basset models, the hyperparameters were set to:

- mutation: `mutate_frac = 0.15`
- translocation: `shift_min = 0, shift_max = 30`
- insertion: `insert_min = 0, insert_max = 30`
- deletion: `delete_min = 0, delete_max = 30`
- reverse-complement: `rc_prob = 0.5`
- noise: `noise_mean = 0, noise_std (standard deviation) = 0.3`

For the DeepSTARR models, the hyperparameters were set to:

- mutation: `mutate_frac = 0.05`
- translocation: `shift_min = 0, shift_max = 20`
- insertion: `insert_min = 0, insert_max = 20`
- deletion: `delete_min = 0, delete_max = 30`

- reverse-complement: `rc_prob = 0`
- noise: `noise_mean = 0, noise_std = 0.3`

When augmentations were used in combinations, the maximum number of augmentations was set to 3 for Basset and 2 for DeepSTARR. The same hyperparameter settings used in DeepSTARR analyses with all augmentations were used for the ChIP-seq analysis. For models trained with combinations of augmentations, the hyperparameters intrinsic to augmentations were set at the values identified above and the maximum number of augmentations per sequence was also determined through a hyperparameter sweep for each dataset (Additional file 1: Figs. S2 and S5).

Unless otherwise specified, all models were trained (with or without data augmentations) for 100 epochs using the Adam optimizer [36] with an initial learning rate of  $1 \times 10^{-3}$  and a weight decay ( $L_2$  penalty) term of  $1 \times 10^{-6}$ ; additionally, we employed early stopping with a patience of 10 epochs and a learning rate decay that decreased the learning rate by a factor of 0.1 when the validation loss did not improve for 5 epochs. For each model trained, the version of the model with the highest-performing weights during its training, as measured by validation loss, is the version of the model whose performance is reported here.

### ***Fine-tuning***

Models that completed training with data augmentations were subsequently fine-tuned on the original dataset without augmentations. Fine-tuning employs the Adam optimizer with a learning rate of  $1 \times 10^{-4}$  and a weight decay ( $L_2$  penalty) term of  $1 \times 10^{-6}$  for 5 epochs. The model that yields the lowest validation loss was used for test time evaluation.

### ***Evaluation***

When evaluating models on validation or test sets, no data augmentations were used on input sequences. For models trained with an insertion augmentation (alone or in combination with other augmentations), each sequence is padded at the 3' end with a stretch of random DNA of length `insert_max`.

### **Interpretability analysis**

#### ***Filter interpretability***

We visualized the first-layer filters of various Basset models according to activation-based alignments [37] and compared how well they match motifs in the 2022 JASPAR nonredundant vertebrates database [38] using Tomtom [39], a motif search comparison tool. Matrix profiles MA1929.1 and MA0615.1 were excluded from filter matching to remove poor quality hits; low information content filters tend to have a high hit rate with these two matrix profiles. Hit rate is calculated by measuring how many filters matched to at least one JASPAR motif. Average  $q$ -value is calculated by taking the average of the smallest  $q$ -values for each filter among its matches.

### Attribution analysis

SHAP-based [40] attribution maps (implemented with GradientShap from the Captum package [41]) were used to generate sequence logos (visualized by Logomaker [42]) for sequences that exhibited high experimental enhancer activity for the Developmental promoter (i.e., task 0 in the DeepSTARR dataset). One thousand random DNA sequences were synthesized to serve as references for each GradientShap-based attribution map. A gradient correction [43] was applied to each attribution map. For comparison, this analysis was repeated for a DeepSTARR model that was trained without any augmentations and a fine-tuned DeepSTARR model that was pretrained with all augmentations (excluding inversions) with two augmentations per sequence.

### CAGI5 challenge analysis

The CAGI5 challenge dataset [21] was used to benchmark model performance on variant effect predictions. This dataset contains massively parallel reporter assays (MPRAs) that measure the effect size of single-nucleotide variants through saturation mutagenesis of 15 different regulatory elements ranging from 187 nt to 600 nt in length. We extracted 600 nt sequences from the reference genome centered on each regulatory region of interest and converted it into a one-hot representation. Alternative alleles were then substituted correspondingly to construct the CAGI test sequences.

For a given Basset model, the output predictions of two input sequences, one with a centered reference allele and the other with an alternative allele, are made. The cell type-agnostic approach employed in this study uses the mean across these values to calculate a single scalar value, functional activity across cell types. The effect size is then calculated with the log-ratio of this single value for the alternative allele and reference allele, according to:  $\log(\text{alternative value}/\text{reference value})$ .

To evaluate the variant effect prediction performance, Pearson correlation was calculated within each CAGI5 experiment between the experimentally measured and predicted effect sizes. The average of the Pearson correlation across all 15 experiments represents the overall performance of the model.

### Supplementary Information

The online version contains supplementary material available at <https://doi.org/10.1186/s13059-023-02941-w>.

**Additional file 1.** Supplementary Tables S1-S3 and Figures S1-S7.

**Additional file 2.** Review history.

### Acknowledgements

This work was supported in part by funding from the NIH grant R01HG012131 and the Simons Center for Quantitative Biology at Cold Spring Harbor Laboratory. This work was performed with assistance from the US National Institutes of Health Grant S10OD028632-01.

### Peer review information

Andrew Cosgrove was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

### Review history

The review history is available as Additional file 2.

### Authors' contributions

NKL and PKK conceived of the method, designed the experiments, and wrote the majority of the code base. NKL, ST, and ZT conducted experiments and analyzed the data. PKK oversaw the project. All authors interpreted the results and contributed to the manuscript. The authors read and approved the final manuscript.

### Availability of data and materials

EvoAug Python package is deposited on the Python Package Index (PyPI) repository with documentation hosted on <https://evoaug.readthedocs.io>. The open-source project repository is available under the MIT license at GitHub [30] (<https://github.com/p-koo/evoaug>). The code to reproduce analyses in this paper is available under the MIT license at GitHub, [https://github.com/p-koo/evoaug\\_analysis](https://github.com/p-koo/evoaug_analysis). Processed data, including DeepSTARR [9], Basset [20] and ChIP-seq analysis, are available at Zenodo [32] ([doi.org/10.5281/zenodo.7265991](https://doi.org/10.5281/zenodo.7265991)). Model weights and code [44] are also available at Zenodo ([doi.org/10.5281/zenodo.7767325](https://doi.org/10.5281/zenodo.7767325)).

### Declarations

#### Ethics approval and consent to participate

Not applicable.

#### Competing interests

The authors declare that they have no competing interests.

Received: 12 December 2022 Accepted: 17 April 2023

Published online: 05 May 2023

### References

- Chen KM, Wong AK, Troyanskaya OG, Zhou J. A sequence-based global map of regulatory activity for deciphering human genetics. *Nat Genet.* 2022;54:1–10.
- Avsec Ž, Agarwal V, Visentin D, Ledsam JR, Grabska-Barwinska A, Taylor KR, Assael Y, Jumper J, Kohli P, Kelley DR. Effective gene expression prediction from sequence by integrating long-range interactions. *Nat Methods.* 2021;18(10):1196–203.
- Zhou J. Sequence-based modeling of three-dimensional genome architecture from kilobase to chromosome scale. *Nat Genet.* 2022;54(5):725–34.
- Hoffman GE, Bendl J, Girdhar K, Schadt EE, Roussos P. Functional interpretation of genetic variants using deep learning predicts impact on chromatin accessibility and histone modification. *Nucleic Acids Res.* 2019;47(20):10597–611.
- Dey KK, Van de Geijn B, Kim SS, Hormozdiari F, Kelley DR, Price AL. Evaluating the informativeness of deep learning annotations for human complex diseases. *Nat Commun.* 2020;11(1):1–9.
- Koo PK, Ploenzke M. Improving representations of genomic sequence motifs in convolutional networks with exponential activations. *Nat Mach Intell.* 2021;3(3):258–66.
- Avsec Ž, Weilert M, Shrikumar A, Krueger S, Alexandari A, Dalal K, Fropf R, McAnany C, Gagneur J, Kundaje A, et al. Base-resolution models of transcription-factor binding reveal soft motif syntax. *Nat Genet.* 2021;53(3):354–66.
- Koo PK, Majdandzic A, Ploenzke M, Anand P, Paul SB. Global importance analysis: an interpretability method to quantify importance of genomic features in deep neural networks. *PLoS Comput Biol.* 2021;17(5):1008925.
- de Almeida BP, Reiter F, Pagani M, Stark A. Deepstarr predicts enhancer activity from DNA sequence and enables the de novo design of synthetic enhancers. *Nat Genet.* 2022;54(5):613–24.
- Horton CA, Alexandari AM, Hayes MG, Schaepe JM, Marklund E, Shah N, Aditham AK, Shrikumar A, Afek A, Greenleaf WJ, et al. Short tandem repeats recruit transcription factors to tune eukaryotic gene expression. *Biophys J.* 2022;121(3):287–8.
- Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. *J Big Data.* 2019;6(1):1–48.
- Fort S, Brock A, Pascanu R, De S, Smith SL. Drawing multiple augmentation samples per image during training efficiently decreases test error. 2021. arXiv preprint [arXiv:2105.13343](https://arxiv.org/abs/2105.13343)
- Zhu S, An B, Huang F. Understanding the generalization benefit of model invariance from a data perspective. *Adv Neural Inf Process Syst.* 2021;34:4328–41.
- Geiping J, Goldblum M, Somepalli G, Shwartz-Ziv R, Goldstein T, Wilson AG. How much data are augmentations worth? An investigation into scaling laws, invariance, and implicit regularization. 2022. arXiv preprint [arXiv:2210.06441](https://arxiv.org/abs/2210.06441)
- Puli A, Zhang LH, Oermann EK, Ranganath R. Out-of-distribution generalization in the presence of nuisance-induced spurious correlations. 2021. arXiv preprint [arXiv:2107.00520](https://arxiv.org/abs/2107.00520)
- Zhou H, Shrikumar A, Kundaje A. Towards a better understanding of reverse-complement equivariance for deep learning models in genomics. In: *Machine Learning in Computational Biology*, PMLR; 2022. p. 1–33
- Toneyan S, Tang Z, Koo PK. Evaluating deep learning for predicting epigenomic profiles. *Nat Mach Intell.* 2022;4:1–13.
- Kelley DR. Cross-species regulatory sequence activity prediction. *PLoS Comput Biol.* 2020;16(7):1008050.
- Frazer KA, Murray SS, Schork NJ, Topol EJ. Human genetic variation and its contribution to complex traits. *Nat Rev Genet.* 2009;10(4):241–51.
- Kelley DR, Snoek J, Rinn JL. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res.* 2016;26(7):990–9.
- Shigaki D, Adato O, Adhikari AN, Dong S, Hawkins-Hooker A, Inoue F, Juven-Gershon T, Kenlay H, Martin B, Patra A, Penzar DD, Schubach M, Xiong C, Yan Z, Boyle AP, Kreimer A, Kulakovskiy IV, Reid J, Unger R, Yosef N, Shendure J, Ahituv N, Kircher M, Beer MA. Integration of multiple epigenomic marks improves prediction of variant impact in saturation mutagenesis reporter assay. *Hum Mutat.* 2019;40(9):1280–91.
- Lu, A.X, Lu, A.X, Moses, A. Evolution is all you need: phylogenetic augmentation for contrastive learning. 2020. arXiv preprint [arXiv:2012.13475](https://arxiv.org/abs/2012.13475)

23. Kryukov GV, Schmidt S, Sunyaev S. Small fitness effect of mutations in highly conserved non-coding regions. *Hum Mol Genet.* 2005;14(15):2221–9.
24. Crawshaw, M. Multi-task learning with deep neural networks: a survey. 2020. arXiv preprint [arXiv:2009.09796](https://arxiv.org/abs/2009.09796)
25. Zbontar J, Jing L, Misra I, LeCun Y, Deny S, Barlow twins: Self-supervised learning via redundancy reduction. In: International Conference on Machine Learning, PMLR; 2021. p. 12310–12320
26. Hjelm RD, Fedorov A, Lavoie-Marchildon S, Grewal K, Bachman P, Trischler A, Bengio Y. Learning deep representations by mutual information estimation and maximization. 2018. arXiv preprint [arXiv:1808.06670](https://arxiv.org/abs/1808.06670)
27. Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
28. Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, Vinyals O, Green T, Dunning I, Simonyan K, et al. Population based training of neural networks. 2017. arXiv preprint [arXiv:1711.09846](https://arxiv.org/abs/1711.09846)
29. Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez JE, Stoica I. Tune: a research platform for distributed model selection and training. 2018. arXiv preprint [arXiv:1807.05118](https://arxiv.org/abs/1807.05118)
30. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. <https://www.tensorflow.org/>. Accessed 31 Oct 2022.
31. Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Necula G, Paszke A, VanderPlas J, Wanderman-Milne S, Zhang Q. JAX: Composable transformations of Python+NumPy programs. [http://github.com/google/jax](https://github.com/google/jax). Accessed 31 Oct 2022.
32. Lee NK, Toneyan S, Tang Z, Koo PK. EvoAug Data [Data set]. Zenodo. 2022. <https://doi.org/10.5281/zenodo.7265991>. Accessed 31 Oct 2022.
33. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning, PMLR; 2015. p. 448–456
34. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res.* 2014;15(1):1929–58.
35. Luo Y, Hitz BC, Gabdank I, Hilton JA, Kagda MS, Lam B, Myers Z, Sud P, Jou J, Lin K, et al. New developments on the encyclopedia of DNA elements (encode) data portal. *Nucleic Acids Res.* 2020;48(D1):882–9.
36. Kingma D, Ba J. Adam: A method for stochastic optimization. 2014. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
37. Koo PK, Ploenzke M. Deep learning for inferring transcription factor binding sites. *Curr Opin Syst Biol.* 2020;19:16–23.
38. Castro-Mondragon JA, Riudavets-Puig R, Rauluseviciute I, Lemma RB, Turchi L, Blanc-Mathieu R, Lucas J, Boddie P, Khan A, Pérez NM, Fornes O, Leung TY, Aguirre A, Hammal F, Schmelter D, Baranasic D, Ballester B, Sandelin A, Lenhard B, Vandepoele K, Wasserman WW, Parcy F, Mathelier A. JASPAR 2022: the 9th release of the open-access database of transcription factor binding profiles. *Nucleic Acids Res.* 2021;50(D1):165–73.
39. Gupta S, Stamatoyannopoulos JA, Bailey TL, Noble WS. Quantifying similarity between motifs. *Genome Biol.* 2007;8(2):1–9.
40. Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. *Adv Neural Inf Process Syst.* 2017;30. [https://papers.nips.cc/paper\\_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html](https://papers.nips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html).
41. Kokhlikyan N, Miglani V, Martin M, Wang E, Alsallakh B, Reynolds J, Melnikov A, Kliushkina N, Araya C, Yan S, Reblitz-Richardson O. Captum: a unified and generic model interpretability library for pytorch. 2020. arXiv preprint [arXiv:2009.07896](https://arxiv.org/abs/2009.07896)
42. Tareen A, Kinney JB. Logomaker: beautiful sequence logos in python. *Bioinformatics.* 2020;36(7):2272–4.
43. Majdandzic A, Rajesh C, Koo PK. Statistical correction of input gradients for black box models trained with categorical input features. 2022. bioRxiv preprint. [biorxiv.org/content/10.1101/2022.04.29.490102v2](https://doi.org/10.1101/2022.04.29.490102v2).
44. Lee NK, Toneyan S, Tang Z, Koo PK. EvoAug reproducibility code. Github. 2022. [https://github.com/p-koo/evoaug\\_analysis](https://github.com/p-koo/evoaug_analysis). Accessed 31 Oct 2022.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Ready to submit your research? Choose BMC and benefit from:**

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

**At BMC, research is always in progress.**

Learn more [biomedcentral.com/submissions](https://biomedcentral.com/submissions)

