ARTICLES
https://doi.org/10.1038/s42256-021-00316-z

nature machine intelligence

An automated framework for efficiently designing deep convolutional neural networks in genomics

Zijun Zhang[1], Christopher Y. Park[1], Chandra L. Theesfeld[2] and Olga G. Troyanskaya[1,2,3] ✉

Convolutional neural networks (CNNs) have become a standard for analysis of biological sequences. Tuning of network architectures is essential for a CNN's performance, yet it requires substantial knowledge of machine learning and commitment of time and effort. This process thus imposes a major barrier to broad and effective application of modern deep learning in genomics. Here we present Automated Modelling for Biological Evidence-based Research (AMBER), a fully automated framework to efficiently design and apply CNNs for genomic sequences. AMBER designs optimal models for user-specified biological questions through the state-of-the-art neural architecture search (NAS). We applied AMBER to the task of modelling genomic regulatory features and demonstrated that the predictions of the AMBER-designed model are significantly more accurate than the equivalent baseline non-NAS models and match or even exceed published expert-designed models. Interpretation of AMBER architecture search revealed its design principles of utilizing the full space of computational operations for accurately modelling genomic sequences. Furthermore, we illustrated the use of AMBER to accurately discover functional genomic variants in allele-specific binding and disease heritability enrichment. AMBER provides an efficient automated method for designing accurate deep learning models in genomics.

Artificial neural networks, or deep learning, have become a state-of-the-art approach to solve diverse problems in biology[1,2]. Convolutional neural networks (CNNs) are especially well suited for identifying high-level features in raw input data with strong spatial structures[3] and as such are powerful at modelling raw genomic sequences and extracting functional information from billions of base pairs in the genome[1]. CNN-based approaches address the computational challenges of predicting the chromatin state and the binding state of RNA-binding proteins from a sequence[4–6], identifying RNA splice sites[7], predicting gene expression[8], prioritizing the disease relevance of variants[9] and many more[1]. Overall, CNNs have become the de facto standard for analysis of genomes—a fundamental problem both in basic understanding of biology and in enabling personalized and precision medicine approaches.

The successful applications of CNNs have been largely attributed to their corresponding architectures. Indeed, for CNN applications in genomics and biomedicine, numerous efforts have been devoted to the development of architectures, such as in DeepSEA[4], Basenji[10], SpliceAI[7] and many more[5,6,11–15]. This is similar to the extensive efforts in architecture designs for tackling computer vision problems, for example, VGG[16], Inception[17] and ResNet[18]. Each of these architectures is motivated and inspired by deep understanding of machine learning and domain knowledge; and requires substantial effort and time commitment by experts to design and implement by extensive trial-and-error processes.

In this Article, we present Automated Modelling for Biological Evidence-based Research (AMBER), an automatic framework for efficiently designing CNNs in genomics. It leverages the groundbreaking idea of automated machine learning, and the related family of algorithms for neural architecture search (NAS) previously developed in the context of computer vision[19,20]. For a given fixed set of training data, AMBER designs an optimal architecture by NAS in a pre-defined model space. We show that the AMBER-designed models significantly outperformed equivalent non-NAS models, matching or even exceeding published expert-designed models. Finally, we use two well-established benchmarks to demonstrate that the AMBER-designed optimal architectures provided significant advantages in prioritizing functional genomic variants in allele-specific binding and heritability enrichment in genome-wide association studies (GWAS). We also illustrate the use of AMBER-designed models to discover disease-relevant variants. Thus, AMBER creates accurate and informative deep learning models that can support functional genomics discoveries by biologists with and without machine learning expertise. AMBER is publicly available at https://github.com/zj-zhang/AMBER.

Overview of methods and workflow

The AMBER framework fully automates the process of training and applying deep learning to genomics, including automatic design of neural network architecture from the training data and downstream functional analyses with the AMBER-designed model (Fig. 1). Unlike existing approaches that focus on making deep learning more accessible using established model architectures[21,22], AMBER automatically designs an optimal architecture for each user-specified problem.

In general, to investigate a biological question with AMBER, a biologist would compile a compendium of functional genomics data such as profiles of transcription factor binding or histone marks along the genome. AMBER uses such sets of compiled training features and labels as input to automatically design deep learning models for the biological question or task of interest (Fig. 1a). Here we use AMBER to model transcriptional regulatory activities. For this task, the training features are one-hot encoded matrices that each represent 1,000-bp DNA sequences from the reference human genome, and the training labels are binary outcomes derived from a compendium of 919 distinct transcriptional regulatory features. These regulatory features include four main functional categories in diverse tissues and cell lines: transcription factors (TF),

[1]Flatiron Institute, Simons Foundation, New York, NY, USA. [2]Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ, USA. [3]Department of Computer Science, Princeton University, Princeton, NJ, USA. ✉e-mail: ogt@cs.princeton.edu

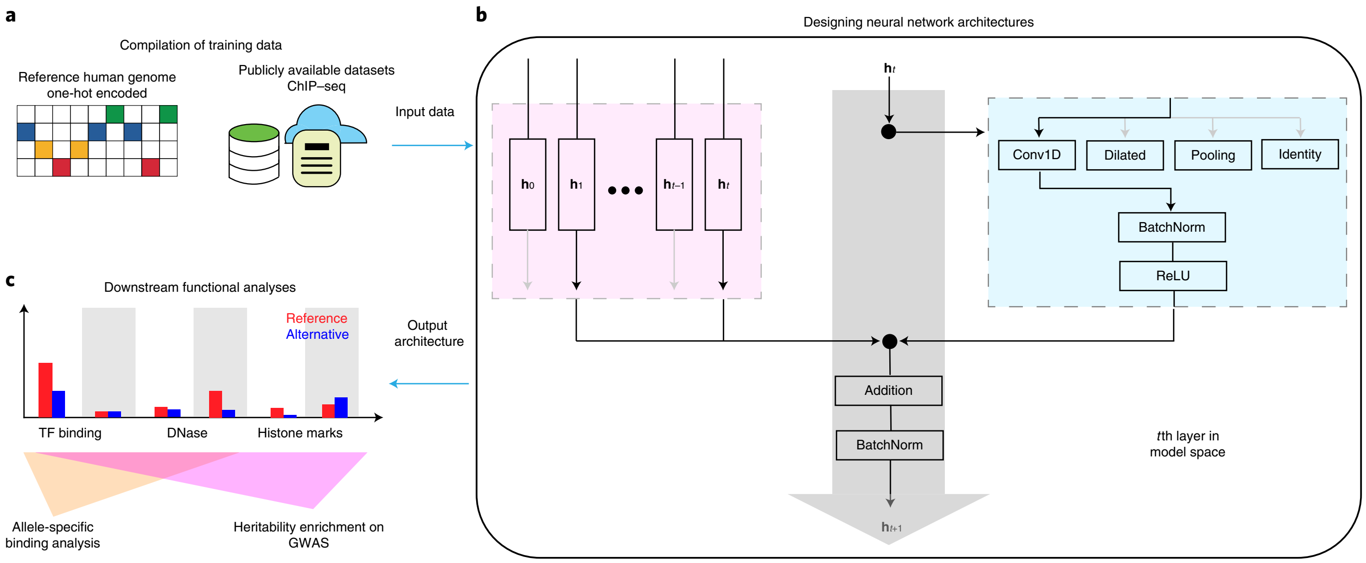
a
Compilation of training data
Reference human genome one-hot encoded
Publicly available datasets ChIP–seq
Input data
b
Designing neural network architectures
$\mathbf{h}_t$
$\mathbf{h}_0$
$\mathbf{h}_1$
$\mathbf{h}_{t-1}$
$\mathbf{h}_t$
Conv1D
Dilated
Pooling
Identity
BatchNorm
ReLU
Addition
BatchNorm
$\mathbf{h}_{t+1}$
tth layer in model space
c
Downstream functional analyses
Reference
Alternative
Output architecture
TF binding
DNase
Histone marks
Allele-specific binding analysis
Heritability enrichment on GWAS

Fig. 1 | Method and workflow overview of AMBER. a, AMBER uses a compendium of training data to design deep learning models in functional genomics. In this application, we applied AMBER to the task of predicting transcriptional regulation on DNA sequences. The features are one-hot encoded matrices representing DNA sequences from the reference human genome, and the labels are functional annotations derived from a large set of ChIP–seq data. **b**, AMBER designs network architecture by searching for optimal combinations of computational operations (blue box) and residual connections (pink box) for each layer, to construct a child model that maps training features to training labels. $\mathbf{h}_t$, the output matrix from the *t*th layer. **c**, Taking the optimal architecture as output, AMBER performs downstream functional analyses. For the transcriptional regulation model, we analysed the functional variant prioritization by AMBER-designed models to predict allele-specific binding and heritability enrichment in GWAS. Shaded areas represent individual transcriptional regulation prediction tasks in the multitasking regime.

polymerases (Pol), histone modifications (Histone) and DNA accessibility (DNase). The task aims to predict whether one or more of the 919 transcriptional regulatory features are active for any 1,000-bp human DNA sequences. In total, the training dataset spans more than 500 million base pairs of the human genome, with 4,400,000, 8,000 and 455,024 samples for training, validation and testing, respectively. Conditioned on this dataset, the target model for AMBER to design is a CNN with multitasking consisting of 919 individual tasks.

To more formally define the neural architecture search problem, the target CNN architecture can be divided into two interconnected components: the computational operations used in each layer (blue box, Fig. 1b) and the residual connections from previous layers (pink box, Fig. 1b). Residual connections have been demonstrated to enable the training of much deeper neural networks with superior performances[18], while greatly expanding the model search space (7.4×10^{19} times more viable architectures in our model space; Methods). Thus, it is essential that residual connection search is considered when AMBER searches for architectures, and the search needs to be efficient. AMBER searches for both of the two components jointly using the efficient neural architecture search (ENAS) controller model[20]. The controller model is parameterized as a recurrent neural network (or RNN; for details, see Methods). Briefly, for each layer in the model search space, the probability of selecting a computational operation is computed by a multivariate classification dependent on the current RNN hidden state; and the probability of selecting the residual connections from a previous layer is a function of the RNN hidden states of the current layer as well as the previous layer of interest. The RNN hidden states were subsequently updated by the operations or residual connections sampled from the output probabilities. To train the controller RNN, we employed reinforcement learning to maximize a reward of area under the receiver operating characteristic curve (AUROC) on the validation dataset.

The output of AMBER is an optimized architecture that performs better than architectures uniformly sampled from the same model search space (Methods). This architecture is then trained from scratch to convergence. Similar to conventional hyperparameter tuning processes, the final training, given the optimal architecture, can be further fine-tuned for other hyperparameters, such as learning rate and batch size (for example, with grid search); in this study, we did not fine-tune these, to ensure fair comparisons between architectures and to demonstrate a use case by non-machine learning experts. Furthermore, we show that AMBER-designed models provide significant advantages over baseline models in multiple practical scenarios, including allele-specific binding and heritability enrichment in GWAS. In the following sections, we describe each part of the AMBER pipeline as well as the downstream analyses in detail.

AMBER designs accurate and efficient models

In our example AMBER application, we defined the model search space of 12 layers, each layer with 7 commonly used computational operations. We chose to use a 12-layer model space because this was the maximum hardware memory limit for a single Nvidia-V100 graphics processing unit (GPU), and shallower models can be attained by an identity operator that in effect removed one layer. In total, this model space hosts 5.1×10^{30} distinct model architectures (Methods).

We benchmarked the computational efficiency of AMBER designing CNNs for genomic sequences. We first demonstrated the quality of AMBER-generated CNN architectures improved over time, by retraining the bootstrapped architectures to evaluate the testing performance every 50 AMBER search steps (Supplementary Fig. 1a). Notably, this retrain analysis alone took ~1,400 GPU hours (= average 10 hours per model × 4 GPUs × 7 steps × 5 models per step), even at a sparse re-sampling rate (that is, per 50 steps); AMBER search bypassed this computational overhaul by the

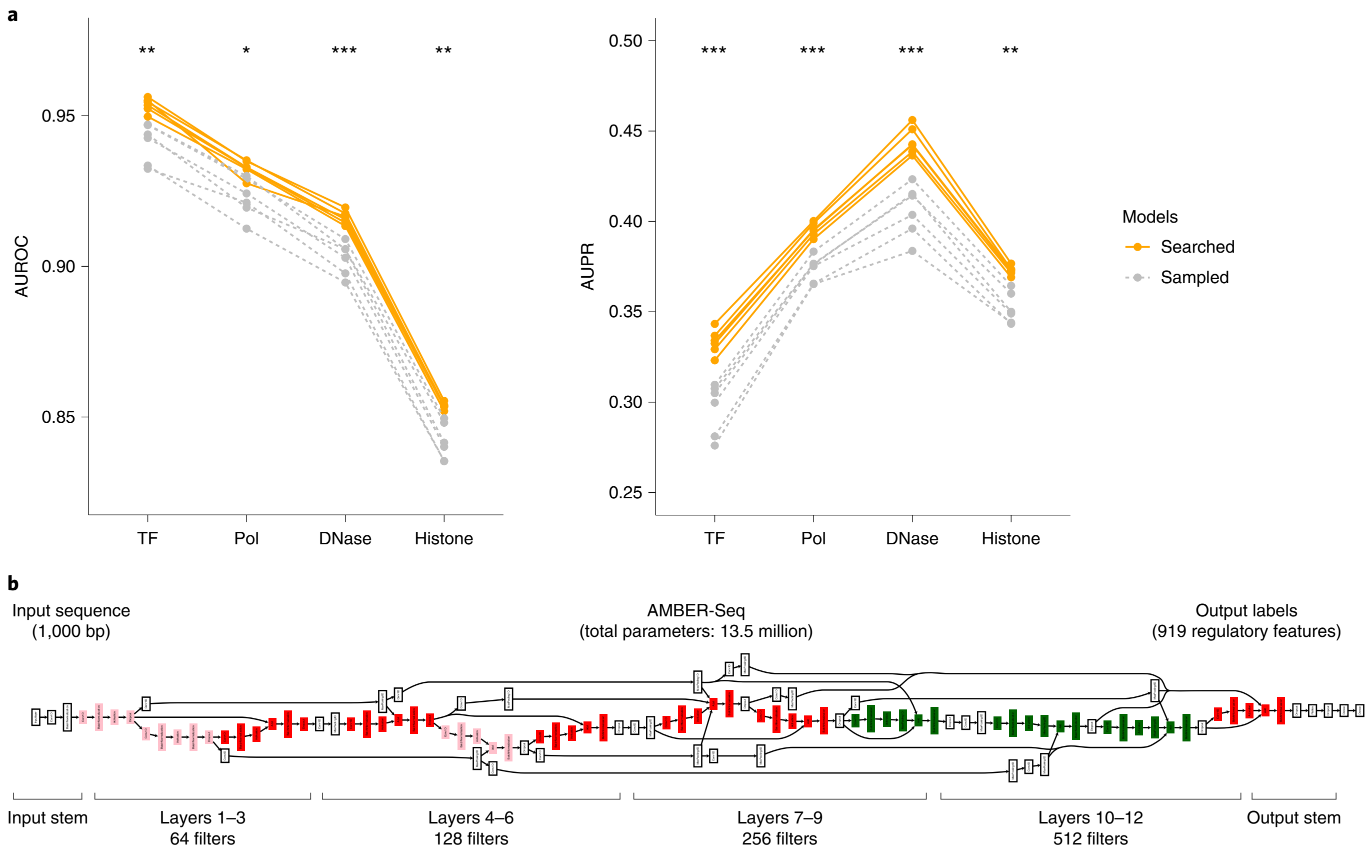

Fig. 2 | AMBER searched architectures outperform sampled architectures. a, The average testing AUROC and area under the precision-recall curve (AUPR) in each functional category were compared for 12 models with distinct architectures either generated by AMBER searched (orange) or uniformly sampled from model space (grey). Each model, represented by a line, was identically trained to convergence. **b**, An illustration of the optimal model architecture, AMBER-Seq, used for downstream analyses. AMBER-Seq is an AMBER-designed deep CNN that outputs a multi-label binary classification for 919 transcriptional regulatory features using 1,000-bp DNA sequences as input. Computational operations are colour-coded as: red, convolution with kernel size 8; pink, convolution with kernel size 4; green, max pooling. Statistical significance (*t*-test): $^{*}P<0.05$, $^{**}P<0.01$, $^{***}P<0.001$.

employment of a parameter-sharing scheme, which substantially reduced the GPU time to 72 hours to design an accurate CNN in our study. Comparing the GPU time used by AMBER search phase to other architecture search algorithms aimed to design CNNs for CIFAR-10 (an established computer-vision dataset) image classifications (Supplementary Fig. 1b), AMBER search phase is orders of magnitude more efficient than reinforcement learning (RL)-NAS[19] and AmoebaNet[23] and comparable to Differentiable Architecture Search (DARTS)[24] and ENAS[20]. Thus, AMBER architecture search enabled highly efficient NAS in genomics and was capable of designing more accurate CNN models as its training progressed.

To robustly evaluate the accuracy of AMBER-designed models, we performed six independent runs of AMBER architecture search, generating six 'searched models'. We compared these searched models with uniformly sampled residual network architectures from the same model space ('sampled models'). Furthermore, we included a set of three-layer fully connected CNNs, representing an ordinal, un-optimized version of CNNs without the residual network architecture (Methods). Given the architectures, the final training steps for AMBER architectures and the sampled residual network architectures were identical, with all models trained to convergence (Methods).

The average testing AUROC and AUPR for each functional category of 919 regulatory feature prediction tasks (that is TF, Pol, DNase and Histone) were compared for the six searched and six sampled model architectures. AMBER-designed architectures significantly outperformed the sampled architectures for all categories (Fig. 2a). The prediction accuracies of different models were more alike within a given functional category than across different categories, indicating that the inherent characteristics of the training data play an essential role in the model's prediction performance, regardless of its model architecture. This is expected, because the training data determined the upper bound of model performance[25], while the searched architectures better approximated this bound. Of course, with unlimited time and resources to enable complete sampling, the optimal architecture is theoretically reachable by sampling as well; however, the time and resource consumption will be tremendous in a model space of 5.1×10^{30} potential architectures. The AMBER architecture search by far speeds up this process and yields model architectures in a narrow high-performance region. Moreover, compared with fully connected CNNs without residual connections (average testing AUROC = 0.860), the performance for both searched (Fig. 2a; $P=1.1\times10^{-9}$, *t*-test) and sampled ($P=4.3\times10^{-7}$, *t*-test) models were significantly better, demonstrating the importance of residual architecture in the CNN model's predictive power. Detailed performances for each model can be found in Supplementary Table 1. Hence, AMBER robustly designs high-performance CNN architectures.

Theoretically, the superior performance from searched model architectures could be achieved by higher relative model complexity. However, no significant differences were observed between the two groups of architectures ($P=0.69$, *t*-test). When we examined

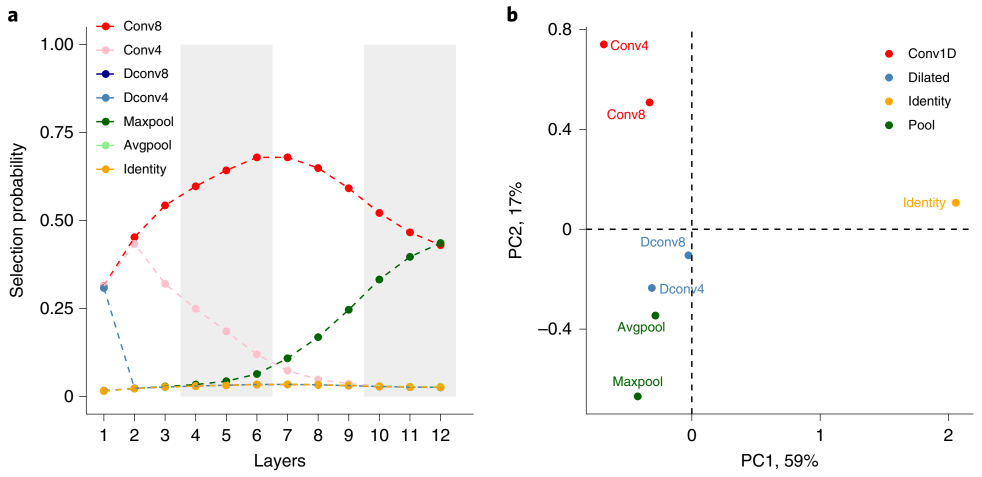

Fig. 3 | Illustration of AMBER architecture search logistics. a, Selection probabilities for distinct computational operations in each layer of the AMBER-Seq controller. For this architecture, convolutional operations were preferred in the bottom to middle layers, while the likelihood of selecting max pooling increased in the top layers. The 12 layers were divided into four blocks (shaded areas) based on the number of their convolutional filters. **b**, PCA of the embedding vectors for different computational operations. PC1 separated identity from computational operations; PC2 separated vanilla convolution, dilated convolution and pooling. Abbreviations: conv8/4, 1D convolution with kernel size 8/4; dconv8/4, dilated convolution with kernel size 8/4; maxpool/avgpool, max/average pooling.

the total number of parameters in each child architecture (dot sizes, Supplementary Fig. 2), the average number of parameters is 12.9 million for searched architectures and 13.3 million for sampled architectures, respectively. Furthermore, we did not observe correlations between the model complexities and their testing performances (Spearman correlation = 0.06, $P = 0.87$). This indicates that the superior performance from searched model architectures is not explicitly linked to model complexities, and that AMBER-designed models are parameter efficient.

For the rest of the analyses in this study, we used the AMBER-designed architecture with the best testing performance, referred to as AMBER-Seq (Fig. 2b), and compared it with the sampled architecture with the best testing performance, referred to as AMBER-Base (Supplementary Fig. 3). Starting with the 1,000-bp one-hot encoded input, we use the input stem of one convolutional layer to expand the 4-channel DNA sequence into 64 channels. The input stem is identical for all child networks. Similarly, the output stem flattens the convolutional feature maps, followed by a dense layer of 925 hidden units to predict the 919 regulatory outputs. The middle 12 layers are variable and grouped into four blocks, each with 3 layers. The total number of parameters in AMBER-Seq is 13.5 million, which is substantially fewer than the original expert-based implementation (52.8 million) in ref. [4] and a model of a similar task (22.8 million) in ref. [10]. With fewer total parameters, AMBER-Seq matched and even exceeded the previously expert-designed implementation in prediction accuracy (AUROC and AUPR; Supplementary Table 1).

Deciphering the logic of AMBER architecture search

Unbiased architecture search performed by AMBER provides insight into which computational operations and architectures are most suited for particular problems in genomics. This can diagnose whether the controller RNN model has learned meaningful representations and help design better model search spaces for future applications.

For this analysis, we analysed the average probability of all computational operations in the last step of the AMBER-Seq controller training across the 12 layers (Fig. 3a). The likelihood of using convolutions (vanilla and dilated convolution) was the highest in the bottom to middle layers; in particular, convolution with kernel size 8 was universally preferred, which is consistent with the choice in expert-based architectures[4]. Interestingly, in higher layers, the likelihood of max pooling starts to increase as the layers are closer to the output. While max-pooling and identity layers both regularize the model complexity, the selection of max pooling appears to maximize the testing performance. When we changed the last three max-pooling layers (all after layer 7 where selection of max pooling started to increase; Fig. 3b) to identity layers in the AMBER-Seq architecture, followed by retraining each model to convergence five times independently, the testing performance decreased 0.1% for AUROC and 0.3% for AUPR on average for all 919 regulatory features ($P = 0.007$ for AUROC and 0.029 for AUPR). In light of hierarchical representation learning of CNNs in computer vision[26], we speculate this is because more high-level features with biological semantic meanings are constituted in the top layers of convolutions, after extensive usage of convolution operations in the bottom layers. Subsequently, by using max pooling as the computational operation in top layers, the model performs feature selections that regularizes model complexity and encourages the usage of high-level semantic features in predicting the final regulatory outcomes. We anticipate that this AMBER architecture design pattern can be further generalized and transferable to other related tasks[27].

The controller's ability to distinguish distinct and similar computational operations is critical for searching high-performance architectures. The differential selection likelihood of operations across layers is a function of previous RNN hidden states and the embedding vectors for each operation, which are learned during the AMBER search phase (Methods). We performed principal component analysis (PCA) on the embedding vectors and analysed how AMBER distinguishes operations (Fig. 3b and Supplementary Fig. 4). We found that the first principal component (PC1) separates identity from all other computational operations, as the identity layer does not involve any computations. In the second principal component (PC2), convolution and pooling were separated with dilated convolution as an intermediate between vanilla convolution and pooling layers. Indeed, dilated convolution enlarges the receptive field similar to pooling layers, while also performing convolution computations[28]. The third principal component further separated computational operations by their corresponding operation types (Supplementary Fig. 4). Overall, the AMBER controller RNN can

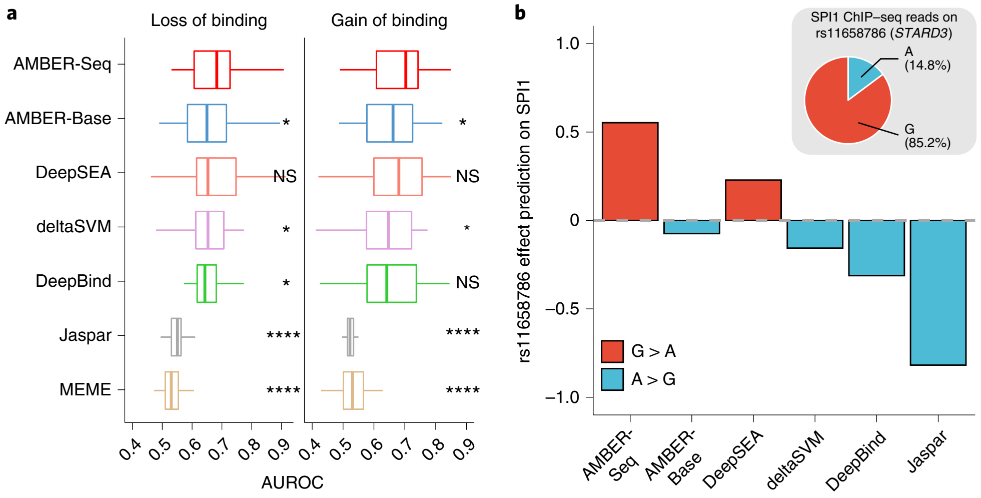

Fig. 4 | Benchmarking variant effect prediction with allele-specific binding. a, Performance of distinguishing loss- and gain-of-binding variants from different models and methods evaluated by AUROC. AMBER-Seq outperformed AMBER-Base on the compendium of allele-specific transcription factor binding sites, matching or even exceeding previous expert-designed machine learning methods. In each boxplot, the centre line marks the median, and the top and bottom lines mark the first and third quartiles. Whiskers are drawn up to the largest and smallest values within the distance of 1.5 times the interquartile range. **b**, A biological case study of variant effect prediction of human genomic variant rs11658786. This variant was predicted to alter an SPI1 binding site in the gene *STARD3*. Among different methods, only AMBER-Seq and DeepSEA predicted the loss-of-binding effect (G > A) of this variant. The A allele significantly reduces SPI1 binding, as illustrated by an independent ChIP–seq experiment (inset). Statistical significance of results of AMBER-Seq versus each of the other models (Wilcoxon test): NS, $P > 0.05$, *$P < 0.05$, **$P < 0.01$, ***$P < 0.001$, ****$P < 0.0001$.

distinguish between similar but distinct operations in building the target architecture.

Variant effect prediction on allele-specific binding

A key application of CNNs in genomics is to predict functional effects of genomic variants, that is, a variant's potential to disrupt an existing molecular mechanism or generate a new one. To investigate the variant effect prediction of different neural network architectures, we compared their ability to correctly predict allele-specific binding for 52,413 variants in 83 distinct transcription factors generated by chromatin immunoprecipitation followed by sequencing (ChIP–seq) experiments[29]. These experiments measure the effect of specific alleles on binding of transcription factors, providing an independent evaluation set for our predictions. For comparison, in addition to AMBER-Seq and AMBER-Base, we included a set of commonly used models and motifs for scoring variant effects: expert-designed CNNs DeepSEA[4] and DeepBind[6], deltaSVM[30], Jaspar[31] and MEME[32] (Fig. 4a). For comparison across different models, variant scores were rank transformed to the range of [−1, 1] and AUROC was computed for each method's ability to distinguish loss-/gain-of-binding alleles versus neutral alleles (Methods). In general, machine learning methods (AMBER-Seq/-Base, DeepSEA, DeepBind, deltaSVM) predicted variant effects significantly better than the motif-based methods (that is, Jaspar and MEME). Next, we extracted the subset of allele-specific variants (both loss- and gain-of-binding alleles) whose predicted effect directions were not consistent among the five machine learning models, to assess each method's accuracy in predicting the direction of change (Supplementary Table 2). This subset consists of arguably harder cases because the predicted effects are highly variable between different machine learning methods. AMBER-Seq achieved the highest accuracy of 60.4% on this set, outperforming 57.5% by AMBER-Base (Supplementary Table 2). Importantly, AMBER-Seq's performance matched or exceeded all other methods, including expert-designed architectures and the AMBER-Base model, demonstrating the power of automated architecture search (asterisks, Fig. 4a).

As a biological case study, we focused on the effect of genomic variant rs11658786 on binding of the SPI1 transcription factor (Fig. 4b). SPI1 (also known as PU.1) is a transcription activator with important functions in haematopoiesis[33], leukemogenesis[34] and adipogenesis[35,36]. AMBER-Seq predicted that the alternative allele at this position reduces SPl1 binding, a prediction supported by independent experimental data—in an independent ChIP–seq dataset, SPI1 predominantly binds to the G allele (85.2%) than the A allele (14.8%; Fig. 4b, inset). Interestingly, all other models except DeepSEA predicted that the alternative allele enhances SPl1 binding, contradicting experimental results. Moreover, rs1165876 is an expression quantitative trait locus (eQTL) for its target gene, *STARD3* (Supplementary Fig. 5a), where the gene expression for the G genotype is the highest and the A genotype is the lowest. The eQTL effect for gene expression is consistent with the AMBER-Seq predicted effect of SPI1 binding and its transcription activation function. Finally, *STARD3* is a gene that encodes a member of a subfamily of lipid trafficking proteins that is involved in cholesterol metabolism. By querying the GWAS catalogue[37], we confirmed that rs11658786 is in strong linkage disequilibrium (LD) with significant GWAS loci in high cholesterol, its interaction terms, as well as smoking status (Supplementary Fig. 5b). Overall, this case study illustrates how variant effects accurately predicted by the automatically generated AMBER-Seq model can be useful for prioritizing functional variants of interest.

Heritability enrichment analysis of GWAS

Finally, we assessed the utility of automatic architecture search by comparing AMBER-Seq with the uniformly sampled AMBER-Base model for explaining disease heritability in GWAS from UK Biobank[38]. Using AMBER-Seq and AMBER-Base models, variant annotations for each of the 919 transcriptional regulatory features of each model were generated, followed by stratified LD-score regression[39] to evaluate their heritability enrichment for a given GWAS (Methods). We analysed the GWAS summary statistics of disease phenotypes previously reported[40] (Methods). The union of the significantly enriched variant annotations (false discovery rate

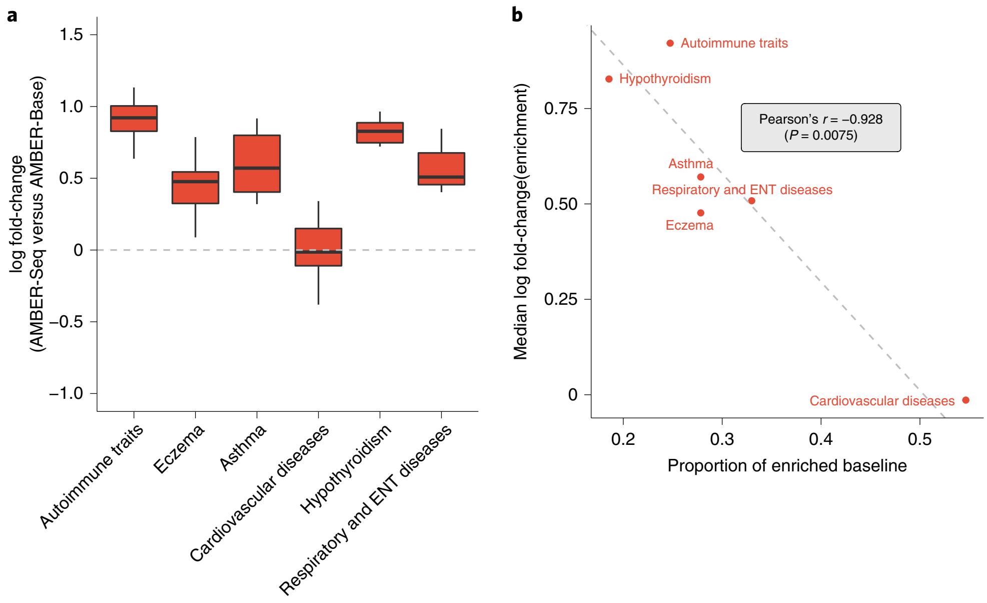

Fig. 5 | Benchmarking heritability enrichment in disease GWAS. a, Comparison of heritability enrichment of AMBER-Seq and AMBER-Base's variant annotations for six disease GWAS. On average, AMBER-Seq annotations were 1.81 times more enriched in disease heritability than the annotations of AMBER-Base. In each boxplot, the centre line marks the median, and the top and bottom lines mark the first and third quartiles. Whiskers are drawn up to the largest and smallest values within the distance of 1.5 times the interquartile range. ENT, ear, nose and throat. **b**, The median magnitude of enrichment fold-change between AMBER-Seq and AMBER-Base was negatively correlated with the proportion of enriched baseline annotations in various diseases, indicating that AMBER can deliver more informative variant annotations in diseases with poor baseline annotations.

(FDR) < 0.05) from both models were used for downstream comparisons and were subsequently examined for overlapping between the AMBER-Seq and AMBER-Base models, or unique to either one of the models (Supplementary Fig. 6). Of the six GWAS diseases we studied, five have significantly more enriched heritability in AMBER-Seq variant annotations (Fig. 5a and Methods). To extend our observations, average heritability enrichment for disease GWAS was computed for each searched and sampled model, and their fold-change of enrichment over AMBER-seq was calculated (Supplementary Table 3). We observed a significant positive correlation between a model's testing AUROC and its fold-change of heritability enrichment (Pearson's correlation = 0.802, $P = 0.002$), with AMBER-Seq having more heritability enrichment over models with poor testing performance. On average, AMBER-Seq variant annotations were 1.81 times more enriched in heritability compared with their counterparts in AMBER-Base across all diseases, indicating that the AMBER-designed model produced more informative variant effect predictions for interpreting disease-associated genomic loci.

Moreover, the variant annotations from AMBER-Seq were particularly useful where baseline annotations[39] fail to explain heritability (Fig. 5b). Baseline annotations are a collection of 97 functional annotations previously curated[39] that cover major known regulatory patterns for human genome. Specifically, to quantify how well the baseline annotations alone explained heritability, we regressed baseline annotations for each GWAS phenotype and calculated the proportion of baseline annotations that were significantly enriched in heritability. We observed a significant negative correlation between the median log fold-change of heritability enrichment of annotations from AMBER-Seq over AMBER-Base, versus the proportion of baseline annotations that are significant (Fig. 5b). This demonstrates that for disease where only a few baseline annotations were significantly enriched in heritability, AMBER-Seq provides the most improvement over AMBER-Base in variant annotation. Conversely, when AMBER-Seq and AMBER-Base heritability enrichment was comparable, the majority of the heritability was largely explained by baseline annotations. Therefore, the automated model design pipeline of AMBER is able to deliver more informative variant annotations in the cases where they are arguably most needed, that is, for diseases that are poorly annotated by baseline annotations.

Discussion

The past decade has witnessed a revolutionary transformation in genomics and exponential accumulation of high-throughput sequencing data. These data enable the study of diverse molecular mechanisms and biological systems through a quantitative lens. Deep learning models have been especially powerful in modelling biological sequences, transforming our ability to interpret genomes[4–6]. These methods generally employ CNNs to extract features from raw genomic sequences, but such an approach comes with a price: a convolutional layer has more hyperparameters than a regular fully connected layer, making the hyperparameter tuning a significantly harder problem. At present, the vast majority of the deep learning models are manually tuned by computational biologists through trial and error, which is time consuming and imposes a substantial barrier for applications of such models by biomedical researchers. To address this challenge, we developed an automatic architecture search framework, AMBER, for efficiently designing optimal deep learning models in genomics. In this study, we have applied AMBER to predicting genomic regulatory features, including downstream analyses such as variant effect prediction and heritability enrichment in GWAS. We found that AMBER matched or exceeded the performance of baseline models, including both expert-designed and uniformly sampled architectures, and is computationally efficient. We anticipate that AMBER will provide a useful tool for biomedical researchers, with and without machine learning expertise, to rapidly develop deep learning models for their specific biological questions.

An important additional application of AMBER is for upgrading existing models with advanced model architectures or updating models when additional data become available. Compared with the original implementation of DeepSEA in 2015, it is interesting to observe that all six runs of AMBER searched models performed better (Supplementary Table 1). This is especially relevant as new and powerful architectures are being developed continuously (for example, residual connections[18] that probably contribute to AMBER-Seq's high performance), yet it is non-trivial to adapt models with the latest deep learning techniques, and such adoption is time and effort consuming. AMBER enables readily integrating such modern approaches into existing expert-designed models. With AMBER, researchers can easily build and apply modern deep learning techniques to find the optimal neural architecture, thereby accelerating the scientific discoveries in biology.

Finally, an important future direction for architecture search in biology is to jointly optimize the prediction accuracy as well as model interpretability. For example, elucidating the decision logic behind variant prediction can help identify molecular pathways that likely led to the predicted effects, shedding new light on molecular mechanisms of transcriptional regulation[41]. In general, an interpretable model is particularly desirable when practitioners need explicit evidence for decision-making and/or for knowledge discovery, such as in hypothesis testing and variant prioritization in genetics studies. Moving forward, we hope frameworks like AMBER can be further developed to identify neural network architectures that are balanced in predictive power and interpretability.

Methods

Designing model search space. The AMBER neural architecture search framework consists of two components to design a child model for specific tasks: (1) a model search space with a large number of different child model architectures; and (2) a controller model that samples architectures from the model search space. For simplicity, we start by illustrating the design of model search space.

The model search space is a sequential collection of layers for the child model, where each layer has a number of candidate computational operations. More concretely, in this study, we aimed to design a one-dimensional (1D) CNN with 12 candidate convolutional layers. Each layer had 6 distinct computational operations: 1D convolution with filter size 4 or 8 (conv4, conv8), dilated 1D convolution with rate 10 and filter size 4 or 8 (dconv4, dconv8), max pooling or average pooling with size 4 (maxpool, avgpool). These hyperparameters for computational operations were selected based on previous studies[4,10]. Moreover, we added an identity mapping to each layer that maps input identically to output without any computations (identity), for potentially reducing the child model complexity. The 12 convolutional layers were connected to fixed input and output stem layers for inputs and outputs, respectively. We divided the 12 convolutional layers into 4 blocks of layers, where each block had doubled the number of filters from the previous block while reduced the size of the feature map by a factor of four. Layers within each block had identical number of filters. We set the first block to have 32 filters for searching architectures.

Formally, let the model space of $T=12$ layers be $\Omega=\{\Omega_1, \Omega_2, \ldots \Omega_t\}$, where Ω_t is the tth layer. Under the current setup, $\Omega_t=\{$conv8, conv4, dconv8, dconv4, maxpool, avgpool, identity$\}$, $\forall\, t$. Let the selection of computational operations at the tth layer be a sparse categorical encoder, that is, $a_t^{\mathrm{o}} \in \{1, 2, \ldots, |\Omega_t|\}$. For example, $a_2^{\mathrm{o}} = 1$ describes the operation for the second hidden layer of the child model is conv8. Therefore, child model computational operations are fully specified by a sequence of integers $\{a_1^{\mathrm{o}}, a_2^{\mathrm{o}}, \ldots, a_{12}^{\mathrm{o}}\}$; in total, different combinations of computational operations constitute $8^{12} \approx 6.9 \times 10^{10}$ viable child models in the model space. The task of finding the child model computational operations can be subsequently considered as a multiclass classification problem with autoregressive characteristics.

In addition to searching operations, we also incorporated the residual connections in the model search space. For the tth layer, the residual connections from layers $1, 2, \ldots, t-1$ are binary encoded by $a_{t,k}^{\mathrm{r}}, \forall\, k \in \{1, 2, .., t-1\}$. If $a_{t,k}^{\mathrm{r}} = 1$, the residual connection is added from the output of the kth layer to the tth layer[18]. Having residual connections is essential for training deeper neural networks, but also significantly increases the complexity in architecture searching. For our 12-layer model space, residual connection search increased the search space by around $2^{12\times 11/2} \approx 7.4 \times 10^{19}$. Now with the residual connections, a full child model can be specified by a sequence of integers $\left\{a_1^{\mathrm{o}}, \ldots, a_t^{\mathrm{o}}, a_{t,1}^{\mathrm{r}}, \ldots, a_{t,t-1}^{\mathrm{r}}, \ldots\right\}$; for brevity, we use a_t to denote both the operations and residual connections in the same layer and use $\{a_1, \ldots, a_t\}$ to represent the child model architecture.

Efficient neural architecture search. We adopted ENAS as the optimization method for searching the child network architectures in the model space[20]. ENAS employs an RNN as the controller model to sequentially predict the child model architecture from the model space. Briefly, the controller RNN, parameterized by θ, generates the child model architectures a with log-likelihood $\pi(a;\theta)$ and is trained by REINFORCE[42]. The policy gradient to maximize the reward R_k over a batch of m sampled architectures is obtained by:

$$\tfrac{1}{m}\sum_{k=1}^{m}\left(R_k - b\right)\sum_{t=1}^{T}\nabla_\theta \log P\left(a_{(t-1):1};\theta\right) =$$

$$\tfrac{1}{m}\sum_{k=1}^{m}\nabla_\theta \pi\left(a;\theta\right)\left(R_k - b\right)$$

We set the reward R_k to be the validation AUROC of the kth child model architecture; b is an exponential moving average of previous rewards to reduce the high variance of the policy gradient. For detailed mathematical formulations, we direct readers to Zoph and Le[19].

Another important feature that enables efficient sampling of child architectures is the parameter-sharing scheme among child models[20]. The computational graph for a child model is a directed acyclic graph (DAG). Under the parameter-sharing scheme, we build a large computational graph, named child DAG with parameters ω, which hosts all possible combinations of child model architectures. The key observation of ENAS is that each child model architecture is a subgraph of the child DAG; therefore, the training of child model parameters is shared and significantly faster. The gradient for the child model parameters ω with respect to its loss function L is obtained through Monte Carlo estimate by the expectation on a set of M architecture samples a:

$$\nabla_\omega \underset{a\sim\pi(a;\theta)}{\mathbb{E}}\left[L\left(\omega;a\right)\right] = \frac{1}{M}\sum_{i=1}^{M}\nabla_\omega L(\omega;a)$$

In this study, we made the following specifications and modifications in training the controller RNN parameters θ and the child DAG parameters ω. The controller RNN was parameterized as a 1-layer long short-term memory (LSTM) of 64 hidden units. Following the original ENAS implementation[20], we set $M=1$ for updating ω; and regularized the proportion of residual connections if it deviated from 0.4. This regularization strength reflects the prior belief of the sparsity of skip connections and has been shown to generate high-performance CNNs in a previous report[20]. The child DAG was set according to the model space described in the previous section. The child DAG was first trained for a whole pass of the training data with a batch size of 1,000 as a warm-up process. Next, the controller RNN sampled 100 child architectures from the child DAG and evaluated their rewards. The child architectures and the rewards were used to train the controller RNN parameters θ. Then we trained the child DAG with architectures sampled from updated $\pi(a;\theta)$. Both controller and child models were trained by Adam optimizer with a learning rate of 0.001. These two training processes were alternated for 300 iterations, and the child architecture with the best reward in the last controller step was extracted.

Sampled architectures were generated by sampling the computational operations uniformly and sampling the residual connections at the proportion of 0.4 as used in searched models. Finally, the child models of searched and sampled were trained from scratch. Both searched and sampled architectures were scaled in width (that is, the number of convolutional kernels) by a factor of 2 and added dropout layers with rate of 0.1 after each pooling layer, a heuristic used in a previous report[20]. We did not fine-tune these to ensure fair comparisons between architectures. In addition to the sampled models, we included three-layer fully connected CNNs without residual connections for comparison. The architectures and hyperparameters for the three-layer fully connected CNNs were matched with the top layers in searched and sampled models, consisting of a convolution layer with 512 filters, a max-pooling layer with size 4 and a dense layer with 925 units. To mitigate overfitting, we fine-tuned the dropout rates in {0.1, 0.3, 0.5, 0.7, 0.9} after the max-pooling layer and reported the best models with dropout rate of 0.7. Each model was trained using identical, default optimization configuration (for example, learning rate, momentum, batch size) until convergence. Convergence was defined as validation AUROC not increasing for at least ten epochs. To more robustly measure the accuracy of AMBER, we ran the search and sample processes six times, respectively. Throughout the study, all processing and analysis of searched and sampled models were strictly identical, except for how we derived their corresponding architectures. We referred to the searched model with the best testing performance as AMBER-Seq and referred to the sampled model with the best testing performance as AMBER-Base.

Dataset for transcriptional regulatory activity prediction. The generic tasks of interest in this study were to predict transcriptional regulatory activity for a given DNA sequence. We aimed to design an end-to-end CNN model that takes raw one-hot encoded DNA as input. Following previous work[4], we used the pre-compiled training, validation and testing dataset downloaded from http://deepsea.princeton.edu/help/. The inputs were one-hot encoded matrices of DNA sequences built on the hg19 reference human genome assembly. The training labels were compiled from a large compendium of publicly available ChIP–seq datasets, which measure the genome-wide molecular profiles such as protein binding or chemical modifications using high-throughput sequencing. In total, there are 919 distinct labels for ChIP–seq profiles of transcription factor binding, histone modification and DNase accessibility assays in diverse human cell lines and tissues; and there are 4,400,000 training samples, 8,000 validation samples and 455,024 testing samples, each of 1,000 bp (1,000 × 4 when one-hot encoded) in length.

Allele-specific binding analysis. A compendium of allele-specific transcription factor binding sites reported previously[29] were compiled for benchmarking the variant effect predictions of the AMBER searched models. Briefly, ChIP–seq data were collected that measured genome-wide binding profiles for 83 unique transcription factors. For each binding site, a binomial test was performed to test allelic imbalance and Benjamini–Hochberg FDR was used to correct for multiple testing. The baseline machine learning methods and the motif scorings were computed previously[29]. We further divided the variants into loss-of-binding alleles (reference reads ratio > 0.6 and FDR < 0.01), gain-of-binding alleles (reference reads ratio < 0.4 and FDR < 0.01) and neutral alleles (FDR > 0.9).

The transcription factors were then mapped to the corresponding cell lines in the multitasking model. To benchmark the models of AMBER-Seq and AMBER-Base with other baseline models, we computed the variant effect scores as the log fold-change between reference allele prediction and alternative allele prediction, as previously described[4]. Then the AUROCs for distinguishing loss-of-function and gain-of-function alleles against the neural alleles were computed for each transcription factor from each model/motif, respectively. To compare the variant effect scores across different methods, we further rank transformed the scores to the range of [−1, 1] while preserving scores at 0 for each method.

For the biological case study of variant effect prediction on single nucleotide polymorphisms (SNP) rs11658786, we reported its variant effect predictions from AMBER-Seq and AMBER-Base along with available baseline variant scoring methods[29]. Variants in high LD with the allele-specific variant of interest were queried from LDlink webserver[43] (https://ldlink.nci.nih.gov/) using the EUR/CEU population and $R^2 > 0.9$. Then the set of variants were processed by plink[44] (v1.90) and plotted using the R package gaston[45]. The eQTLs for allele-specific variants were queried using the Genotype-Tissue Expression (GTEx) web portal[46] (https://www.gtexportal.org/home/).

GWAS analysis. To evaluate the informativeness of the variant annotations from different model architectures, we used stratified LD-score regression[39] to assess the heritability enrichment for variant annotations. First, we downloaded the summary statistics files from UK Biobank for disease phenotypes reported previously[40]. Selene[22] (v0.4.2) was employed to process the genome-wide variant effect predictions for SNPs from the 1000 Genome Project (European cohort) for each transcriptional regulatory feature in both AMBER-designed AMBER-Seq model and uniformly sampled AMBER-Base model. Then the variant effect predictions were subsequently converted to LD scores and regressed on the χ^2 statistics using ldsc v1.0.1 Python implementation (https://github.com/bulik/ldsc), conditioned on a set of 97 baseline LD annotations from baselineLD v2.2 (https://data.broadinstitute.org/alkesgroup/LDSCORE/). We restricted our analyses to phenotypes with ratio statistics less than 10% to avoid potential model misspecifications[39]. The enrichment P values were computed by ldsc and corrected for multiple testing by Benjamini–Hochberg FDR. Regulatory features whose variant annotations were significant (FDR < 0.05) in either the searched AMBER-Seq or the sampled AMBER-Base models were analysed for their overlapping statistics and enrichment fold-changes across models.

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

Data availability

All data used in this study are publicly available and the URLs are provided in the corresponding sections in Methods. Training data for the genomic regulatory features were downloaded from http://deepsea.princeton.edu/help/ as described in ref. [4]. The ground-truth data for allele-specific binding analysis were obtained from the supplementary data of ref. [29]. The UK Biobank GWAS summary statistics data are reported in ref. [40] and downloaded from https://alkesgroup.broadinstitute.org/UKBB/.

Code availability

The AMBER package is available on GitHub at https://github.com/zj-zhang/AMBER; the analysis presented in this study is available on GitHub at https://github.com/zj-zhang/AMBER-Seq. The AMBER code is publicly available on Zenodo at https://zenodo.org/record/4384777[47].

Received: 14 August 2020; Accepted: 8 February 2021;
Published online: 15 March 2021

References

1. Eraslan, G., Avsec, Ž., Gagneur, J. & Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.* https://doi.org/10.1038/s41576-019-0122-6 (2019).
2. Ching, T. et al. Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* **15**, 20170387 (2018).
3. LeCun, Y. & Bengio, Y. in *The Handbook of Brain Theory and Neural Networks* (ed. Arbib, M. A.) 3361(10) (MIT Press, 1995).
4. Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat. Methods* **12**, 931–934 (2015).
5. Kelley, D. R., Snoek, J. & Rinn, J. L. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res* **26**, 990–999 (2016).
6. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.* **33**, 831–838 (2015).
7. Jaganathan, K. et al. Predicting splicing from primary sequence with deep learning. *Cell* **176**, 535–548.e24 (2019).
8. Zhou, J. et al. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nat. Genet.* **50**, 1171–1179 (2018).
9. Zhou, J. et al. Whole-genome deep-learning analysis identifies contribution of noncoding mutations to autism risk. *Nat. Genet.* **51**, 973–980 (2019).
10. Kelley, D. R. et al. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome Res* **28**, 739–750 (2018).
11. Ma, J. et al. Using deep learning to model the hierarchical structure and function of a cell. *Nat. Methods* **15**, 290–298 (2018).
12. Zhang, Z. et al. Deep-learning augmented RNA-seq analysis of transcript splicing. *Nat. Methods* **16**, 307–310 (2019).
13. Quang, D. & Xie, X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Res.* **44**, e107 (2016).
14. Angermueller, C., Lee, H. J., Reik, W. & Stegle, O. DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biol.* **18**, 67 (2017).
15. Agarwal, V. & Shendure, J. Predicting mRNA abundance directly from genomic sequence using deep convolutional neural networks. *Cell Rep.* **31**, 107663 (2020).
16. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations* 1–14 (ICLR, 2014).
17. Chollet, F. Xception: deep learning with depthwise separable convolutions. In *Proc. 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 1800–1807 (IEEE, 2017).
18. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 770–778 (IEEE, 2016).
19. Zoph, B. & Le, Q. V. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations* (ICLR, 2017).
20. Pham, H., Guan, M. Y., Zoph, B., Le, Q. V. & Dean, J. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning* 4095–4104 (PMLR, 2018).
21. Avsec, Ž. et al. The Kipoi repository accelerates community exchange and reuse of predictive models for genomics. *Nat. Biotechnol.* **37**, 592–600 (2019).
22. Chen, K. M., Cofer, E. M., Zhou, J. & Troyanskaya, O. G. Selene: a PyTorch-based deep learning library for sequence data. *Nat. Methods* **16**, 315–318 (2019).
23. Real, E., Aggarwal, A., Huang, Y. & Le, Q. V. Regularized evolution for image classifier architecture search. In *Proc. AAAI Conference on Artificial Intelligence* Vol. 33, 4780–4789 (2019).
24. Liu, H., Simonyan, K. & Yang, Y. Darts: differentiable architecture search. In *International Conference on Learning Representations* (ICLR, 2019).
25. He, X., Zhao, K. & Chu, X. AutoML: a survey of the state-of-the-art. *Knowl. Based Syst.* **212**, 106622 (2021).
26. Lee, H., Grosse, R., Ranganath, R. & Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. 26th International Conference On Machine Learning, ICML 2009* 609–616 (ACM, 2009); https://doi.org/10.1145/1553374.1553453
27. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 8697–8710 (IEEE, 2018).
28. Yu, F. & Koltun, V. Multi-scale context aggregation by dilated convolutions. In *4th International Conference on Learning Representations* (ICLR, 2016).
29. Wagih, O., Merico, D., Delong, A. & Frey, B. J. Allele-specific transcription factor binding as a benchmark for assessing variant impact predictors. Preprint at *bioRxiv* https://doi.org/10.1101/253427 (2018).
30. Lee, D. et al. A method to predict the impact of regulatory variants from DNA sequence. *Nat. Genet.* **47**, 955–961 (2015).
31. Bryne, J. C. et al. JASPAR, the open access database of transcription factor-binding profiles: new content and tools in the 2008 update. *Nucleic Acids Res.* **36**, D102–D106 (2008).
32. Machanick, P. & Bailey, T. MEME-ChIP: motif analysis of large DNA datasets. *Bioinformatics* **27**, 1696–1697 (2011).

33. Zhang, P. et al. Negative cross-talk between hematopoietic regulators: GATA proteins repress PU.1. *Proc. Natl Acad. Sci. USA* **96**, 8705–8710 (1999).
34. Metcalf, D. et al. Inactivation of PU.1 in adult mice leads to the development of myeloid leukemia. *Proc. Natl Acad. Sci. USA* **103**, 1486–1491 (2006).
35. Wang, F. & Tong, Q. Transcription factor PU.1 is expressed in white adipose and inhibits adipocyte differentiation. *Am. J. Physiol. Physiol.* **295**, C213–C220 (2008).
36. Lin, L. et al. Adipocyte expression of PU.1 transcription factor causes insulin resistance through upregulation of inflammatory cytokine gene expression and ROS production. *Am. J. Physiol. Endocrinol. Metab.* **302**, E1550 (2012).
37. Buniello, A. et al. The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic Acids Res.* **47**, D1005–D1012 (2019).
38. Bycroft, C. et al. The UK Biobank resource with deep phenotyping and genomic data. *Nature* **562**, 203–209 (2018).
39. Finucane, H. K. et al. Partitioning heritability by functional annotation using genome-wide association summary statistics. *Nat. Genet.* **47**, 1228–1235 (2015).
40. Loh, P. R., Kichaev, G., Gazal, S., Schoech, A. P. & Price, A. L. Mixed-model association for biobank-scale datasets. *Nat. Genet.* **50**, 906–908 (2018).
41. Zhang, Z., Zhou, L., Gou, L. & Wu, Y. N. Neural architecture search for joint optimization of predictive power and biological knowledge. Preprint at https://arxiv.org/abs1909.00337 (2019).
42. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**, 229–256 (1992).
43. Machiela, M. & Chanock, S. LDlink: a web-based application for exploring population-specific haplotype structure and linking correlated alleles of possible functional variants. *Bioinformatics* **31**, 3555–3557 (2015).
44. Purcell, S. et al. PLINK: A tool set for whole-genome association and population-based linkage analyses. *Am. J. Hum. Genet.* **81**, 559–575 (2007).
45. Claire Dandine-Roulland, C. & Perdry, H. Genome-wide data manipulation, association analysis and heritability estimates in R with Gaston 1.5. In *46th European Mathematical Genetics Meeting* (EMGM, 2018).
46. Lonsdale, J. et al. The Genotype-Tissue Expression (GTEx) project. *Nat. Genet.* **45**, 580–585 (2013).
47. Zhang, Z. Code for 'An automated framework for efficiently designing deep convolutional neural networks in genomics'. *Zenodo* https://doi.org/10.5281/ZENODO.4384777 (2020).

Acknowledgements

We acknowledge all members of the Troyanskaya laboratory for helpful discussions. We acknowledge that the work in this paper was performed at the high-performance computing resources at Simons Foundation. O.G.T. is a CIFAR fellow.

Author contributions

Z.Z. and O.G.T. conceived the study. Z.Z. implemented the experiments. C.Y.P. and C.L.T. contributed research materials and analytic tools. Z.Z. and O.G.T. wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-021-00316-z.

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

Peer review information *Nature Machine Intelligence* thanks the anonymous reviewers for their contribution to the peer review of this work.

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

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

nature research

Corresponding author(s): Olga Troyanskaya

Last updated by author(s): Jan 26, 2021

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 our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the 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
☐	☒	A statement on 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 statistical parameters 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

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Training data for the genomic regulatory features were downloaded from http://deepsea.princeton.edu/help/ . The ground-truth data for allele-specific binding analysis is obtained from the supplementary data of Waigh et al., 2015. The UK-Biobank GWAS summary statistics data is reported in Loh et al., 2018; and downloaded from https://alkesgroup.broadinstitute.org/UKBB/ .
Data analysis	The AMBER package is available at GitHub: https://github.com/zj-zhang/AMBER ; the analysis presented in this study is available at https://github.com/zj-zhang/AMBER-Seq . Selene (v0.4.2) was employed to process the genome-wide variant effect predictions for SNPs from the 1000 Genome Project (European cohort). LDscore regression was performed using ldsc v1.0.1 Python implementation (https://github.com/bulik/ldsc), conditioned on a set of 97 baseline LD annotations from baselineLD v2.2 (https://data.broadinstitute.org/alkesgroup/LDSCORE/). Variants in high LD with the allele-specific variant of interest were queried from LDlink webserver (https://ldlink.nci.nih.gov/) using the EUR/CEU population and R2>0.9. Then the set of variants were processed by plink (v1.90) and plotted by R package gaston.

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 and reviewers. 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

All data used in this manuscript were publicly available and the corresponding web links were provided in the Method section. No restrictions were imposed on data availability.

Field-specific reporting

Please select the one below that is 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/documents/nr-reporting-summary-flat.pdf

Life sciences study design

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

Sample size	We used all publicly available data where applicable and no sample size calculation was performed.
Data exclusions	No data were excluded from the analyses.
Replication	The computational experiments were ran independently for 12 times in total to assess the replication.
Randomization	Randomization is not applicable to this study because it does not involve experimental subject allocation.
Blinding	Blinding is not applicable to this study because no experimental design applicable.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study	
☒	☐	Antibodies
☒	☐	Eukaryotic cell lines
☒	☐	Palaeontology and archaeology
☒	☐	Animals and other organisms
☒	☐	Human research participants
☒	☐	Clinical data
☒	☐	Dual use research of concern

Methods

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