

# 1 **An automated framework for efficiently designing deep** 2 **convolutional neural networks in genomics**

3  
4 Zijun Zhang<sup>1</sup>, Christopher Y. Park<sup>1</sup>, Chandra L. Theesfeld<sup>2</sup>, Olga G. Troyanskaya<sup>1,2,3,\*</sup>

5  
6 <sup>1</sup>Flatiron Institute, Simons Foundation, New York City, New York, United States of America

7 <sup>2</sup>Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, New Jersey,  
8 United States of America

9 <sup>3</sup>Department of Computer Science, Princeton University, Princeton, New Jersey, United States of  
10 America

11 \* Correspondence to: [ogt@cs.princeton.edu](mailto:ogt@cs.princeton.edu)

## 12 13 14 **Abstract**

15 Convolutional neural networks (CNNs) have become a standard for analysis of biological  
16 sequences. Tuning of network architectures is essential for CNN's performance, yet it requires  
17 substantial knowledge of machine learning and commitment of time and effort. This process thus  
18 imposes a major barrier to broad and effective application of modern deep learning in genomics.  
19 Here, we present AMBER, a fully automated framework to efficiently design and apply CNNs  
20 for genomic sequences. AMBER designs optimal models for user-specified biological questions  
21 through the state-of-the-art Neural Architecture Search (NAS). We applied AMBER to the task  
22 of modelling genomic regulatory features and demonstrated that the predictions of the AMBER-  
23 designed model are significantly more accurate than the equivalent baseline non-NAS models

24 and match or even exceed published expert-designed models. Interpretation of AMBER  
25 architecture search revealed its design principles of utilizing the full space of computational  
26 operations for accurately modelling genomic sequences. Furthermore, we illustrated the use of  
27 AMBER to accurately discover functional genomic variants in allele-specific binding and  
28 disease heritability enrichment. AMBER provides an efficient automated method for designing  
29 accurate deep learning models in genomics.

30

31

## 32 **Main**

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

43

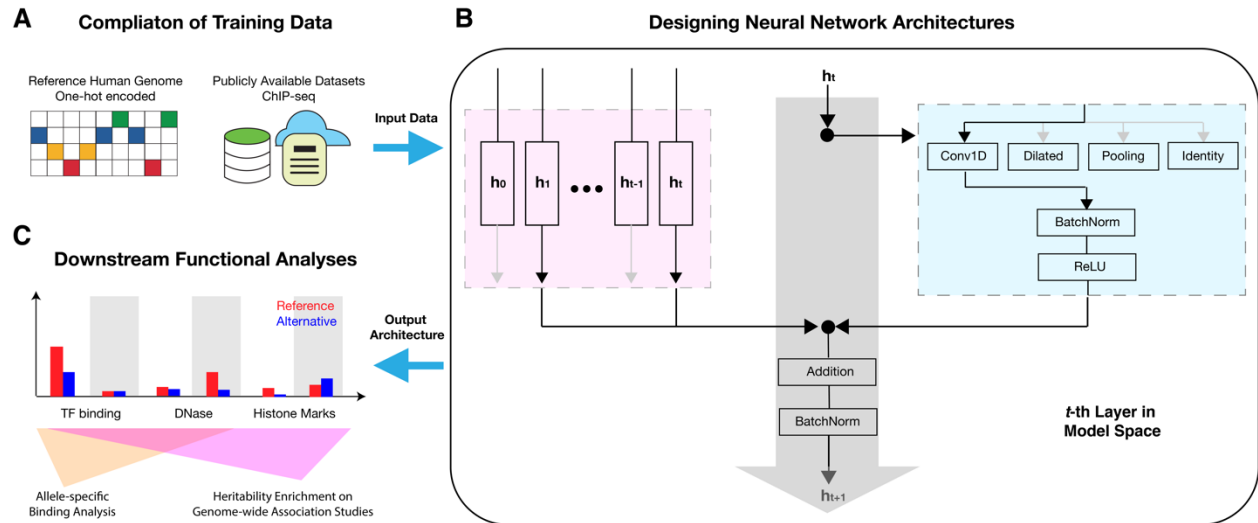
44 The successful applications of CNNs have been largely attributed to their corresponding  
45 architectures. Indeed, for CNN applications in genomics and biomedicine, numerous efforts have  
46 been devoted to the development of architectures, such as in DeepSEA<sup>4</sup>, Basenji<sup>10</sup> and SpliceAI<sup>7</sup>.

47 This is similar to the extensive efforts in architecture designs for tackling computer vision  
48 problems, for example VGG<sup>11</sup>, Inception<sup>12</sup>, and ResNet<sup>13</sup>. Each of these architectures is  
49 motivated and inspired by deep understanding of machine learning and domain knowledge; and  
50 requires substantial effort and time commitment by experts to design and implement by  
51 extensive trial-and-error processes.

52

53 Here, we present Automated Modelling for Biological Evidence-based Research (AMBER), an  
54 automatic framework for efficiently designing convolutional neural networks in genomics. To  
55 our knowledge, AMBER is the first automated approach specifically designed for modelling  
56 genomic sequences. It leverages the groundbreaking idea of Automated Machine Learning (or  
57 AutoML), and the related family of algorithms for Neural Architecture Search (NAS) previously  
58 developed in the context of computer vision<sup>14,15</sup>. For a given fixed set of training data, AMBER  
59 designs an optimal architecture by NAS in a pre-defined model space. We show that the  
60 AMBER-designed models significantly outperformed equivalent non-NAS models, matching or  
61 even exceeding published expert-designed models. Finally, we use two well-established  
62 benchmarks to demonstrate that the AMBER-designed optimal architectures provided significant  
63 advantages in prioritizing functional genomic variants in allele-specific binding and heritability  
64 enrichment in Genome-Wide Association Studies (GWAS). We also illustrate the use of  
65 AMBER-designed models to discover disease-relevant variants. Thus, AMBER creates accurate  
66 and informative deep-learning models that can support functional genomics discoveries by  
67 biologists with and without machine learning expertise. AMBER is publicly available at  
68 <https://github.com/zj-zhang/AMBER>.

69



70

71

72 **Figure 1. Method and workflow overview of AMBER.**

73 **A)** AMBER uses a compendium of training data to design deep learning models in functional

74 genomics. In this application, we applied AMBER to the task of predicting transcriptional

75 regulation on DNA sequences. The features are one-hot encoded reference human genome, and

76 the labels are functional annotations derived from a large set of ChIP-seq data. **B)** AMBER

77 designs network architecture by searching for optimal combinations of computational operations

78 (blue box) and residual connections (red box) for each layer, to construct a child model that maps

79 training features to training labels. **C)** Taking the optimal architecture as output, AMBER

80 performs downstream functional analyses. For the transcriptional regulation model, we analyzed

81 the functional variant prioritization by AMBER-designed models to predict allele-specific

82 binding and heritability enrichment in GWAS.

83

## 84 **Overview of methods and workflow**

85 The AMBER framework fully automates the process of training and applying deep learning to  
86 genomics, including automatic design of neural network architecture from the training data and  
87 downstream functional analyses with the AMBER-designed model (**Figure 1**). Unlike existing  
88 approaches that focus on making deep learning more accessible using established model  
89 architectures<sup>16,17</sup>, AMBER automatically designs an optimal architecture for each user-specified  
90 problem.

91

92 In general, to investigate a biological question with AMBER, a biologist would compile a  
93 compendium of functional genomics data such as profiles of transcription factor binding or  
94 histone marks along the genome. AMBER uses such sets of compiled training features and labels  
95 as input to automatically design deep learning models for the biological question or task of  
96 interest (**Figure 1A**). Here, we use AMBER to model transcriptional regulatory activities. For  
97 this task, the training features are one-hot encoded matrices that each represent 1000-bp DNA  
98 sequences from the reference human genome, and the training labels are binary outcomes  
99 derived from a compendium of 919 distinct transcriptional regulatory features. These regulatory  
100 features include four main functional categories in diverse tissues and cell lines: transcription  
101 factors (TF), polymerases (Pol), histone modifications (Histone), and DNA accessibility  
102 (DNase). The task aims to predict whether one or more of the 919 transcriptional regulatory  
103 features are active for any 1000-bp human DNA sequences. In total, the training dataset spans  
104 more than 500 million base-pairs of the human genome, with 4400000, 8000, and 455024  
105 samples for training, validation and testing, respectively. Conditioned on this dataset, the target

106 model for AMBER to design is a convolutional neural network with multi-tasking consisting of  
107 919 individual tasks.

108

109 To more formally define the neural architecture search problem, the target convolutional neural  
110 network architecture can be divided into two interconnected components: the computational  
111 operations used in each layer (blue box, **Figure 1B**), and the residual connections from previous  
112 layers (red box, **Figure 1B**). Residual connections have been demonstrated to enable the training  
113 of much deeper neural networks with superior performances<sup>13</sup>, while greatly expanding the  
114 model search space ( $7.4 \times 10^{19}$  times more viable architectures in our model space; see  
115 **Methods**). Thus, it's essential that residual connection search is considered when AMBER  
116 searches for architectures, and the search needs to be efficient. AMBER searches for both of the  
117 two components jointly using the Efficient Neural Architecture Search (ENAS) controller  
118 model<sup>15</sup>. The controller model is parameterized as a Recurrent Neural Network (or RNN; for  
119 details, see **Methods**). Briefly, for each layer in the model search space, the probability of  
120 selecting a computational operation is computed by a multivariate classification dependent on the  
121 current RNN hidden state; and the probability of selecting the residual connections from a  
122 previous layer is a function of the RNN hidden states of the current layer as well as the previous  
123 layer of interest. The RNN hidden states were subsequently updated by the operations or residual  
124 connections sampled from the output probabilities. To train the controller RNN, we employed  
125 reinforcement learning to maximize a reward of AUROC on the validation dataset.

126

127 The output of AMBER is an optimized architecture that performs better than architectures  
128 uniformly sampled from the same model search space (**Methods**). Furthermore, we show that

129 AMBER-designed models provide significant advantages over baseline models in multiple  
130 practical scenarios, including allele-specific binding and heritability enrichment in GWAS. In the  
131 following sections, we describe each part of the AMBER pipeline as well as the downstream  
132 analyses in detail.

133

134

### 135 **AMBER designs accurate and efficient models**

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

141

142 We benchmarked the computational efficiency of AMBER by comparing the GPU time used by  
143 the AMBER search phase to other architecture search algorithms (**Table 1**). The time of  
144 AMBER search phase is orders of magnitude more efficient than RL-NAS<sup>14</sup> and AmoebaNet<sup>18</sup>  
145 and comparable to DARTS<sup>19</sup> and ENAS<sup>15</sup>.

146

147 To robustly evaluate the accuracy of AMBER-designed models, we performed six independent  
148 runs of AMBER architecture search, generating six “searched models”. We compared these  
149 searched models with uniformly sampled residual network architectures from the same model  
150 space (“sampled models”). Given the architectures, the final training step for AMBER

151 architectures and the sampled residual network architectures were identical, with all models  
152 trained to convergence (**Methods**).

153

154

**Table 1. Runtime comparison in GPU hours**

Method	Time (in GPU hours)
AMBER	72
AmoebaNet	75600
DARTS	96
ENAS	10.8
RL-NAS	537600

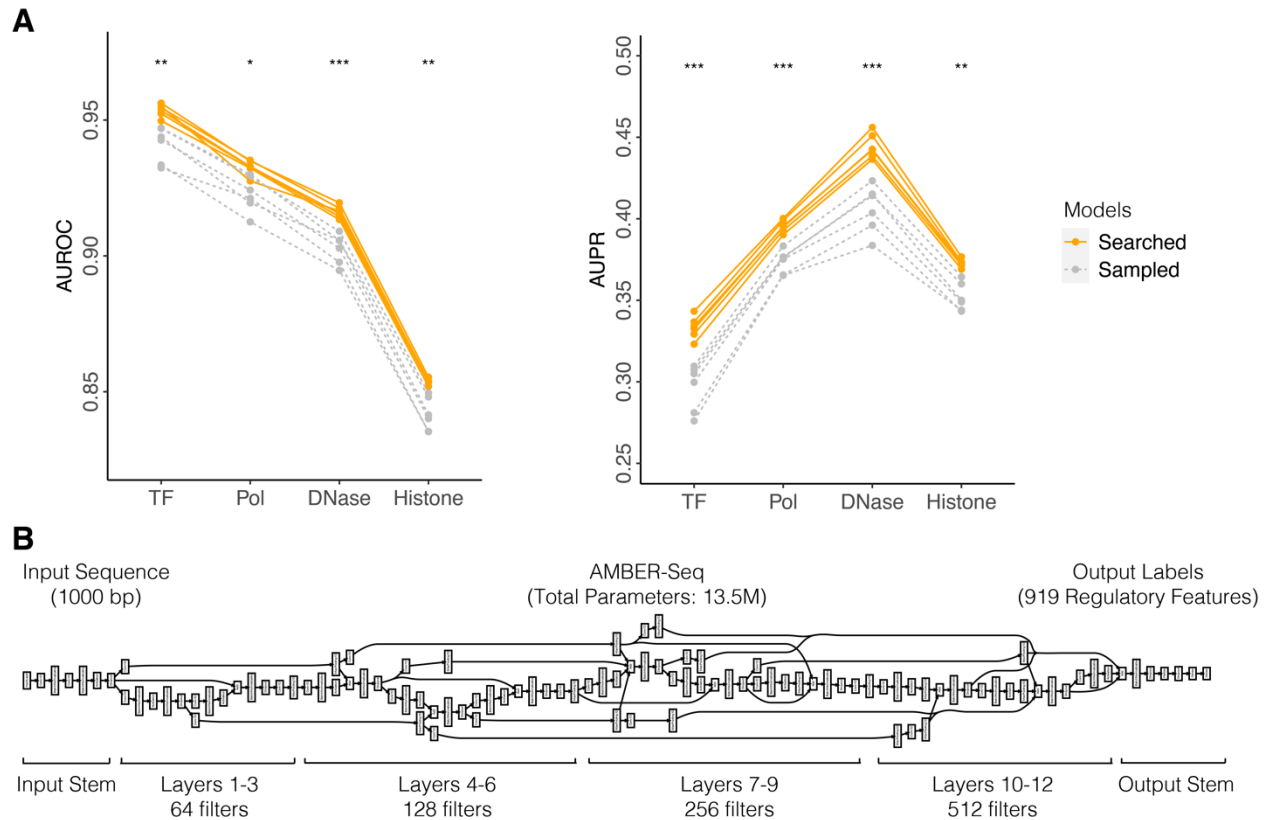
155

156

157 The average testing AUROC and AUPR for each functional category of 919 regulatory feature  
158 prediction tasks (i.e. TF, Pol, DNase and Histone) were compared for the six searched and six  
159 sampled model architectures. AMBER-designed architectures significantly outperformed the  
160 sampled architectures for all categories (**Figure 2A**). The prediction accuracies of different  
161 models were more alike within a given functional category than across different categories,  
162 indicating that the inherent characteristics of the training data play an essential role in the  
163 model's prediction performance, regardless of its model architecture. This is expected, because  
164 the training data determined the upper bound of model performance<sup>20</sup>, while the searched  
165 architectures better approximated this bound. Of course, with unlimited time and resources to  
166 enable complete sampling, the optimal architecture is theoretically reachable by sampling as  
167 well; however, the time and resource consumption will be tremendous in a model space of



168  $5.1 \times 10^{30}$  potential architectures. The AMBER architecture search by far speeds up this process  
169 and yields model architectures in a narrow high-performance region. Detailed performances for  
170 each model can be found in **Supplementary Table 1**. Hence, AMBER robustly designs high-  
171 performance convolutional neural network architectures.  
172



173

174 **Figure 2. AMBER searched architectures outperform sampled architectures.**

175 **A)** The average testing AUROC and AUPR in each functional category were compared for  
176 twelve models with distinct architectures either generated by AMBER searched (orange) or  
177 uniformly sampled from model space (grey). Each model, represented by a line, was identically  
178 trained to convergence. **B)** An illustration of the optimal model architecture, AMBER-Seq, used  
179 for downstream analyses. AMBER-Seq is an AMBER-designed deep convolutional neural

180 networks that outputs a multi-label binary classification for 919 transcriptional regulatory  
181 features using 1000-bp DNA sequences as inputs.

182 Statistical significance (t-test) \*:  $p < 0.05$ , \*\*:  $p < 0.01$ , \*\*\*:  $p < 0.001$

183

184

185

186 Theoretically, the superior performance from searched model architectures could be achieved by  
187 higher relative model complexity. However, no significant differences were observed between  
188 the two groups of architectures (p-value=0.69, t-test). When we examined the total number of  
189 parameters in each child architecture (dot sizes, **Supplementary Figure 1**), the average number  
190 of parameters is 12.9 million for searched architectures and 13.3 million for sampled  
191 architectures, respectively. Furthermore, we did not observe correlations between the model  
192 complexities and their testing performances (spearman correlation=0.06, p-value=0.87). This  
193 indicates that the superior performance from searched model architectures is not explicitly linked  
194 to model complexities, and that AMBER-designed models are parameter-efficient.

195

196 For the rest of the analyses in this study, we used the AMBER-designed architecture with the  
197 best testing performance, referred to as AMBER-Seq (**Figure 2B**); and compared it to the  
198 sampled architecture with the best testing performance, referred to as AMBER-Base  
199 (**Supplementary Figure 2**). Starting with the 1000-bp one-hot encoded input, we use the input  
200 stem of one convolutional layer to expand the 4-channel DNA sequence into 64 channels. The  
201 input stem is identical for all child networks. Similarly, the output stem flattens the convolutional  
202 feature maps, followed by a dense layer of 925 hidden units to predict the 919 regulatory outputs.

203 The middle 12 layers are variable and grouped into four blocks, each with 3 layers. The total  
204 number of parameters in AMBER-Seq is 13.5 million, which is substantially fewer than the  
205 original expert-based implementation (52.8 million) in ref.<sup>4</sup> and a model of a similar task (22.8  
206 million) in ref.<sup>10</sup>. With fewer total parameters, AMBER-Seq matched and even exceeded the  
207 previously expert-designed implementation in prediction accuracy (AUROC and AUPR; see  
208 **Supplementary Table 1**).

209

210

### 211 **Deciphering the logic of AMBER architecture search**

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

216

217 For this analysis, we analyzed the average probability of all computational operations in the last  
218 step of the AMBER-Seq controller training across the 12 layers (**Figure 3A**). The likelihood of  
219 using convolutions (vanilla and dilated convolution) was the highest in the bottom- to middle-  
220 layers; in particular, convolution with kernel size 8 was universally preferred, which is consistent  
221 with the choice in expert-based architectures<sup>4</sup>. Interestingly, in higher layers, the likelihood of  
222 max pooling starts to increase as the layers are closer to the output. In light of CNN's  
223 hierarchical representation learning in computer vision<sup>21</sup>, we speculate this is because more high-  
224 level features with biological semantic meanings are constituted in the top layers of  
225 convolutions, after extensive usage of convolution operations in the bottom layers. Subsequently,

226 by using max pooling as the computational operation in top layers, the model performs feature  
227 selections that regularizes model complexity and encourages the usage of high-level semantic  
228 features in predicting the final regulatory outcomes. We anticipate this AMBER architecture  
229 design pattern can be further generalized and transferable to other related tasks<sup>22</sup>.

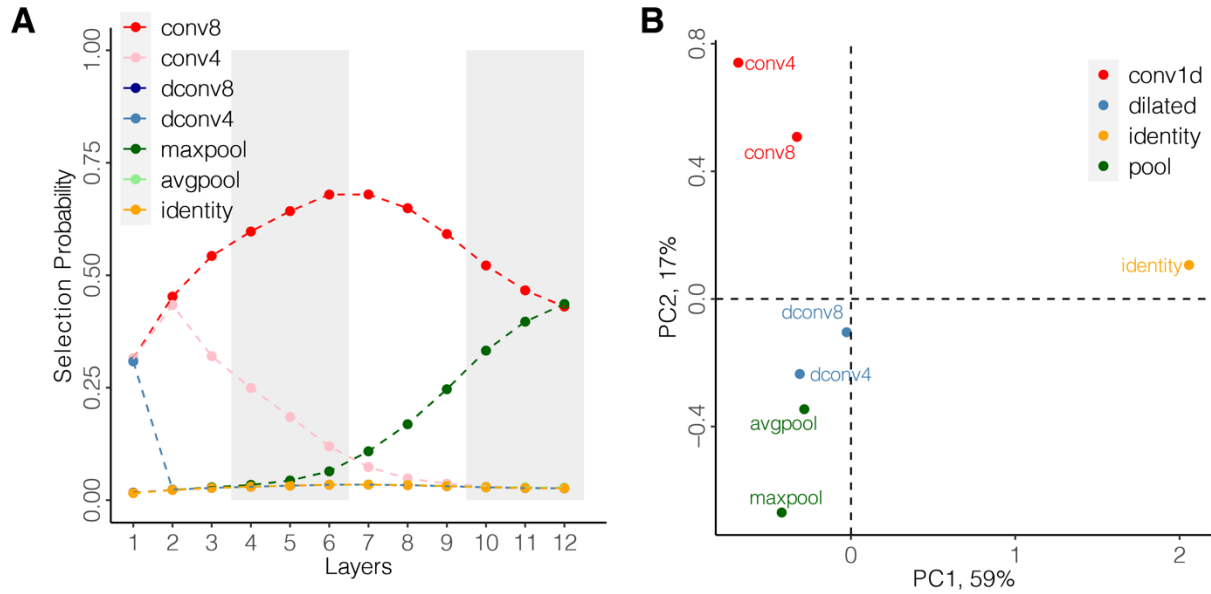
230

231 The controller's ability to distinguish distinct and similar computational operations is critical for  
232 searching high-performance architectures. The differential selection likelihood of operations  
233 across layers is a function of previous RNN hidden states and the embedding vectors for each  
234 operation, which are learned during the AMBER search phase (**Methods**). We performed  
235 Principal Component Analysis (PCA) on the embedding vectors and analyzed how AMBER  
236 distinguishes operations (**Figure 3B** and **Supplementary Figure 3**). We found that the first  
237 principal component separates identity from all other computational operations, as the identity  
238 layer does not involve any computations. In the second principal component, convolution and  
239 pooling were separated with dilated convolution as an intermediate between vanilla convolution  
240 and pooling layers. Indeed, dilated convolution enlarges the receptive field similar to pooling  
241 layers, while also performs convolution computations<sup>23</sup>. The third principal component further  
242 separated computational operations by their corresponding operation types (**Supplementary**  
243 **Figure 3**). Overall, AMBER controller RNN can distinguish between similar but distinct  
244 operations in building the target architecture.

245

246

247



248

249 **Figure 3. Illustration of AMBER architecture search logistics.**

250 **A)** Selection probabilities for distinct computational operations in each layer of the AMBER-Seq  
251 controller. For this architecture, convolutional operations were preferred in bottom to middle  
252 layers, while the likelihood of selecting max pooling increased in top layers. **B)** Principal  
253 component analysis of the embedding vectors for different computational operations. PC1  
254 separated identity from computational operations; PC2 separated vanilla convolution, dilated  
255 convolution and pooling.

256 Abbreviations: *conv8/4*: 1D convolution with kernel size 8/4; *dconv8/4*: dilated convolution with  
257 kernel size 8/4; *max/avgpool*: max/average pooling.

258

## 259 **Variant effect prediction on allele-specific binding**

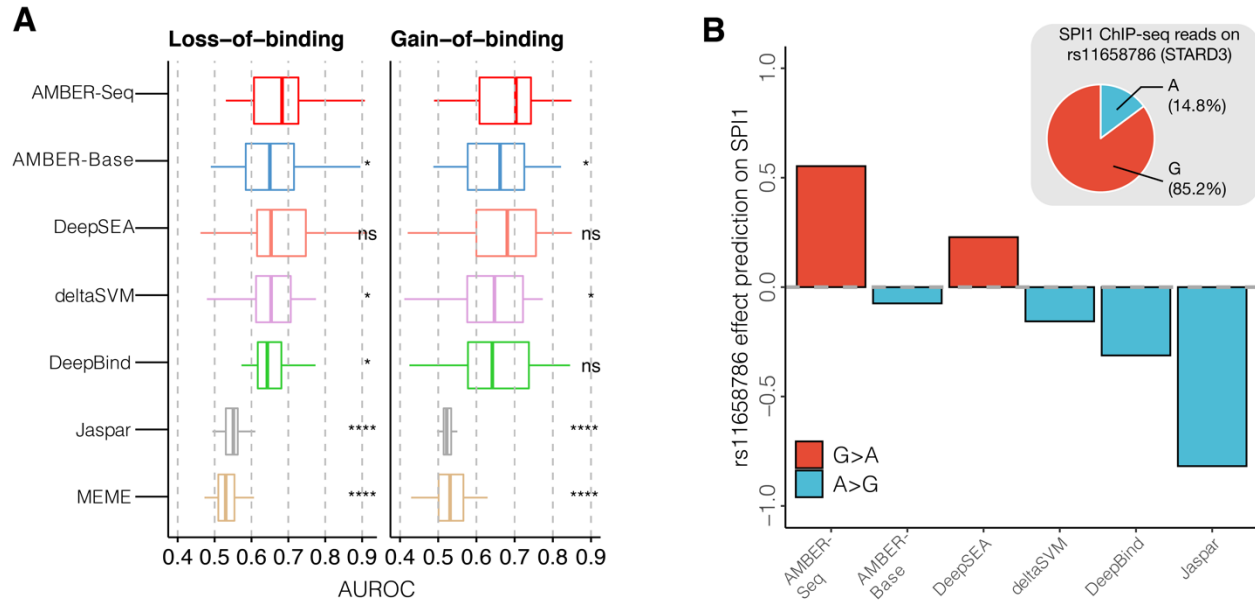
260 A key application of convolutional neural networks in genomics is to predict functional effects of  
261 genomic variants, i.e. a variant's potential to disrupt an existing molecular mechanism or  
262 generate a new one. To investigate the variant effect prediction of different neural network  
263 architectures, we compared their ability to correctly predict allele-specific binding for 52,413  
264 variants in 83 distinct transcription factors generated by ChIP-seq experiments<sup>24</sup>. These  
265 experiments measure the effect of specific alleles on binding of transcription factors, providing  
266 an independent evaluation set for our predictions. For comparison, in addition to AMBER-Seq  
267 and AMBER-Base, we included a set of commonly used models and motifs for scoring variant  
268 effects: expert-designed CNNs DeepSEA<sup>4</sup> and DeepBind<sup>6</sup>, deltaSVM<sup>25</sup>, Jaspars<sup>26</sup> and MEME<sup>27</sup>  
269 (**Figure 4A**). For comparison across different models, variant scores were rank transformed to  
270 the range of [-1, 1] and AUROC was computed for each method's ability to distinguish  
271 loss/gain-of-binding alleles versus neutral alleles (**Methods**). In general, machine learning  
272 methods (AMBER, DeepSEA, DeepBind, deltaSVM) predict variant effects significantly better  
273 than the motif-based methods (i.e. Jaspars and MEME). Importantly, AMBER-Seq's performance  
274 matched or exceeded all other methods, including expert-designed architectures and the  
275 AMBER-Base model, demonstrating the power of automated architecture search (asteroid,  
276 **Figure 4A**).

277

278 As a biological case study, we focused on the effect of genomic variant rs11658786 on binding  
279 of the SPI1 transcription factor (**Figure 4B**). SPI1 (also known as PU.1) is a transcription  
280 activator with important functions in hematopoiesis<sup>28</sup>, leukemogenesis<sup>29</sup>, and adipogenesis<sup>30,31</sup>.  
281 AMBER-Seq predicted that the alternative allele at this position reduces SPI1 binding, a

282 prediction supported by independent experimental data -- in an independent ChIP-seq dataset,  
283 SPI1 predominantly binds to the G allele (85.2%) than the A allele (14.8%; **Figure 4B, inset**).  
284 Interestingly, all other models except DeepSEA predicted that the alternative allele enhances  
285 SPI1 binding, contradicting experimental results. Moreover, rs1165876 is an eQTL for its target  
286 gene, STARD3 (**Supplementary Figure 4A**), where the gene expression for the G genotype is  
287 the highest and the A genotype is the lowest. The eQTL effect for gene expression is consistent  
288 with the AMBER-Seq predicted effect of SPI1 binding and its transcription activation function.  
289 Finally, STARD3 is a gene that encodes a member of a subfamily of lipid trafficking proteins  
290 that is involved in cholesterol metabolism. By querying GWAS catalog<sup>32</sup>, we confirmed that  
291 rs1165876 is in strong LD with significant GWAS loci in high cholesterol, its interaction terms,  
292 as well as smoking status (**Supplementary Figure 4B**). Overall, this case study illustrates how  
293 variant effects accurately predicted by the automatically generated AMBER-Seq model can be  
294 useful for prioritizing functional variants of interest.  
295

296



297

298 **Figure 4. Benchmarking variant effect prediction with allele-specific binding.**

299 **A)** Performance of distinguishing loss- and gain-of-binding variants from different models and  
300 methods evaluated by AUROC. AMBER-Seq outperformed AMBER-Base on the compendium  
301 of allele-specific transcription factor binding sites, matching or even exceeding previous expert-  
302 designed machine learning methods. In each boxplot, center line marks the median while top and  
303 bottom lines mark the first and third quartiles. **B)** A biological case study of variant effect  
304 prediction of human genomic variant rs11658786. This variant was predicted to alter a SPI1  
305 binding site in gene STARD3. Among different methods, only AMBER-Seq and DeepSEA  
306 predicted the loss-of-binding effect (G>A) of this variant. The A allele significantly reduces SPI1  
307 binding, as illustrated by an independent ChIP-seq experiment (inset).

308 Statistical significance of results of AMBER-Seq versus each of the other models (Wilcoxon  
309 test) ns:  $p > 0.05$ , \*:  $p < 0.05$ , \*\*:  $p < 0.01$ , \*\*\*:  $p < 0.001$ , \*\*\*\*:  $p < 0.0001$

310



## 311 **Heritability enrichment analysis of genome-wide association studies**

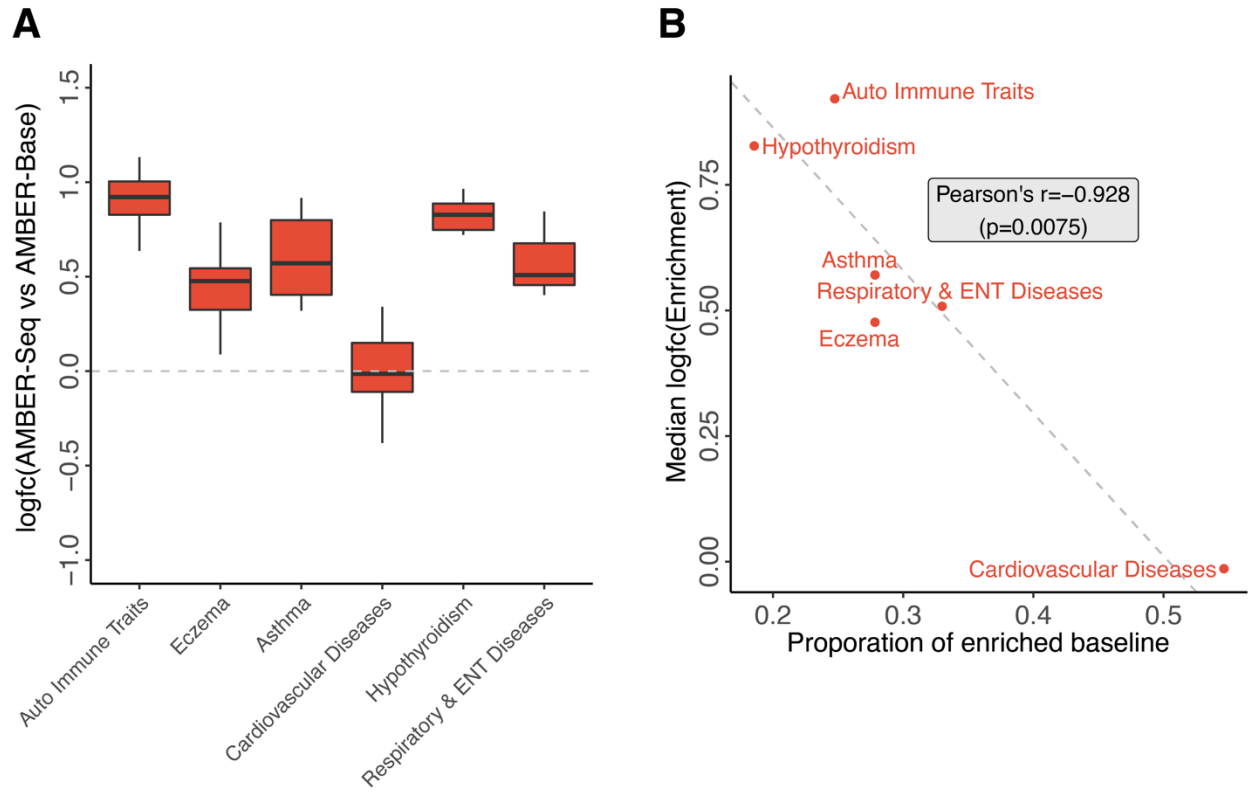
312 Finally, we assessed the utility of automatic architecture search by comparing AMBER-Seq with  
313 the uniformly sampled AMBER-Base model for explaining disease heritability in GWAS from  
314 UK Biobank<sup>33</sup>. Using AMBER-Seq and AMBER-Base models, variant annotations for each of  
315 the 919 transcriptional regulatory features of each model were generated, followed by stratified  
316 LD-score regression<sup>34</sup> to evaluate their heritability enrichment for a given GWAS (**Methods**).  
317 We analyzed the GWAS summary statistics of disease phenotypes previously reported<sup>35</sup>  
318 (**Methods**). The union of the significantly enriched variant annotations (FDR<0.05) from both  
319 models were used for downstream comparisons and were subsequently examined for overlapping  
320 between the AMBER-Seq and AMBER-Base models, or unique to either one of the models  
321 (**Supplementary Figure 5**). Of the six GWAS diseases we studied, five have significantly more  
322 enriched heritability in AMBER-Seq variant annotations (**Figure 5A; Methods**). On average,  
323 AMBER-Seq variant annotations were 1.81x more enriched in heritability compared to their  
324 counterparts in AMBER-Base across all diseases, indicating that AMBER-designed model  
325 produced more informative variant effect predictions for interpreting disease-associated genomic  
326 loci.

327

328 Moreover, the variant annotations from AMBER-Seq were particularly useful where baseline  
329 annotations<sup>34</sup> fail to explain heritability (**Figure 5B**). Baseline annotations are a collection of 97  
330 functional annotations previously curated<sup>34</sup> that cover major known regulatory patterns for  
331 human genome. Specifically, to quantify how well the baseline annotations alone explained  
332 heritability, we regressed baseline annotations for each GWAS phenotype and calculated the  
333 proportion of baseline annotations that were significantly enriched in heritability. We observed a

334 significant negative correlation between the median log fold-change of heritability enrichment of  
335 annotations from AMBER-Seq over AMBER-Base, versus the proportion of baseline  
336 annotations that are significant (**Figure 5B**). This demonstrates that for disease where only a few  
337 baseline annotations were significantly enriched in heritability, AMBER-Seq provides the most  
338 improvement over AMBER-Base in variant annotation. Conversely, when AMBER-Seq and  
339 AMBER-Base heritability enrichment was comparable, the majority of the heritability was  
340 largely explained by baseline annotations. Therefore, the automated model design pipeline of  
341 AMBER is able to deliver more informative variant annotations in the cases where they are  
342 arguably most needed, i.e. for diseases that are poorly annotated by baseline annotations.  
343

344



345

346

347 **Figure 5. Benchmarking heritability enrichment in disease GWAS.**

348 **A)** Comparison of heritability enrichment of AMBER-Seq and AMBER-Base's variant

349 annotations for six disease GWAS. On average, AMBER-Seq annotations were 1.81x more  
350 enriched in disease heritability than the annotations of AMBER-Base. In each boxplot, center

351 line marks the median while top and bottom lines mark the first and third quartiles. **B)** The

352 median magnitude of enrichment fold-change between AMBER-Seq and AMBER-Base was

353 negatively correlated with the proportion of enriched baseline annotations in various diseases,

354 indicating that AMBER can deliver more informative variant annotations in diseases with poor

355 baseline annotations.

356

357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379

## **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 modeling biological sequences, transforming our ability to interpret genomes<sup>4-6</sup>. These methods generally employ convolutional neural networks 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. To date, the vast majority (if not all) 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 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.

380 An important additional application of AMBER is for upgrading existing models with advanced  
381 model architectures or updating models when additional data become available. Compared to the  
382 original implementation of DeepSEA in 2015, it is interesting to observe that all six runs of  
383 AMBER searched models performed better (**Supplementary Table 1**). This is especially  
384 relevant as new and powerful architectures are being developed continuously (e.g. residual  
385 connections<sup>13</sup> that likely contribute to AMBER-Seq's high performance), yet it is non-trivial to  
386 adapt models with the latest deep learning techniques, and such adoption is time- and effort-  
387 consuming. AMBER enables readily integrating such modern approaches into existing expert-  
388 designed models. With AMBER, researchers can easily build and apply modern deep learning  
389 techniques to find the optimal neural architecture, thereby accelerating the scientific discoveries  
390 in biology.

391

392 Finally, an important future direction for architecture search in biology is to jointly optimize the  
393 prediction accuracy as well as model interpretability. For example, elucidating the decision logic  
394 behind variant prediction can help identify molecular pathways that likely led to the predicted  
395 effects, shedding new light on molecular mechanisms of transcriptional regulation<sup>36</sup>. In general,  
396 an interpretable model is particularly desirable when practitioners need explicit evidence for  
397 decision making and/or for knowledge discovery, such as in hypothesis testing and variant  
398 prioritization in genetics studies. Moving forward, we hope frameworks like AMBER can be  
399 further developed to identify neural network architectures that are balanced in predictive power  
400 and interpretability.

401

402 **Methods**

403

#### 404 **Designing model search space**

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

409

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

424

425 Formally, let the model space of  $T=12$  layers be  $\Omega = \{\Omega_1, \Omega_2, \dots, \Omega_T\}$ , where  $\Omega_t$  is the  $t$ -th layer.

426 Under the current setup,  $\Omega_t = \{\text{conv8}, \text{conv4}, \text{dconv8}, \text{dconv4}, \text{maxpool}, \text{avgpool}, \text{identity}\}, \forall t$ .

427 Let the selection of computational operations at  $t$ -th layer be a sparse categorical encoder, i.e.

428  $a_t^o \in \{1, 2, \dots, |\Omega_t|\}$ . For example,  $a_2^o = 1$  describes the operation for the second hidden layer of

429 the child model is conv8. Therefore, child model computational operations are fully specified by

430 a sequence of integers  $\{a_1^o, a_2^o, \dots, a_{12}^o\}$ ; in total, different combinations of computational

431 operations constitutes  $8^{12} \approx 6.9 \times 10^{10}$  viable child models in the model space. The task of

432 finding the child model computational operations can be subsequently considered as a multi-

433 class classification problem with auto-regressive characteristics.

434

435 In addition to searching operations, we also incorporated the residual connections in the model

436 search space. For the  $t$ -th layer, the residual connections from layers  $1, 2, \dots, t-1$  are binary

437 encoded by  $a_{t,k}^r, \forall k \in \{1, 2, \dots, t-1\}$ . If  $a_{t,k}^r = 1$ , the residual connection is added from the

438 output of the  $k$ -th layer to the  $t$ -th layer<sup>13</sup>. Having residual connections are essential for training

439 deeper neural networks, but also significantly increases the complexity in architecture searching.

440 For our 12-layer model space, residual connection search increased the search space by around

441  $2^{12 \times 11/2} \approx 7.4 \times 10^{19}$ . Now with the residual connections, a full child model can be specified

442 by a sequence of integers  $\{a_1^o, \dots, a_t^o, a_{t,1}^r, \dots, a_{t,t-1}^r, \dots\}$ ; for brevity, we use  $a_t$  to denote both the

443 operations and residual connections in the same layer and use  $\{a_1, \dots, a_t\}$  to represent the child

444 model architecture.

445

446

447 **Efficient neural architecture search**

448 We adopted Efficient Neural Architecture Search (ENAS) as the optimization method for  
449 searching the child network architectures in the model space<sup>15</sup>. ENAS employs a Recurrent  
450 Neural Network (RNN) as the controller model to sequentially predict the child model  
451 architecture from the model space. Briefly, the controller RNN, parameterized by  $\theta$ , generates  
452 the child model architectures  $a$  with log-likelihood  $\pi(a; \theta)$  and is trained by REINFORCE<sup>37</sup>. The  
453 policy gradient to maximize the reward  $R_k$  over a batch of  $m$  sampled architectures is obtained  
454 by:

$$\frac{1}{m} \sum_{k=1}^m (R_k - b) \cdot \sum_{t=1}^T \nabla_{\theta} \log P(a_{(t-1):1}; \theta) =$$
$$\frac{1}{m} \sum_{k=1}^m \nabla_{\theta} \pi(a; \theta) \cdot (R_k - b)$$

457  
458 We set the reward  $R_k$  to be the validation AUROC of the  $k$ -th child model architecture;  $b$  is an  
459 exponential moving average of previous rewards to reduce the high variance of the policy  
460 gradient.

461  
462 Another important feature that enables efficiently sampling of child architectures is the  
463 parameter sharing scheme among child models<sup>15</sup>. The computational graph for a child model is a  
464 Directed Acyclic Graph (DAG). Under the parameter sharing scheme, we build a large  
465 computational graph, named child DAG with parameters  $\omega$ , which hosts all possible  
466 combinations of child model architectures. The key observation of ENAS is that each child  
467 model architecture is a subgraph of the child DAG, therefore the training of child model



468 parameters is shared and significantly faster. The gradient for the child model parameters  $\omega$  is  
469 obtained through Monte Carlo estimate:

470 
$$\nabla_{\omega} E_{a \sim \pi(a; \theta)} [L(\omega; a)] = \frac{1}{M} \sum_{i=1}^M \nabla_{\omega} L(\omega; a)$$

471  
472 In this study, we made the following specifications and modifications in training the controller  
473 RNN parameters  $\theta$  and the child DAG parameters  $\omega$ . The controller RNN was parameterized as  
474 a 1-layer LSTM of 64 hidden units. Following the original ENAS implementation<sup>15</sup>, we set  $M=1$   
475 for updating  $\omega$ ; and regularized the proportion of residual connections if it deviated from 0.4.  
476 The child DAG was set according to the model space described in the previous section. The child  
477 DAG was first trained for a whole pass of the training data with a batch size of 1000 as a warm-  
478 up process. Next, the controller RNN sampled 100 child architectures from the child DAG and  
479 evaluated their rewards. The child architectures and the rewards were used to train the controller  
480 RNN parameters  $\theta$ . Then we trained the child DAG with architectures sampled from updated  
481  $\pi(a; \theta)$ . Both controller and child models were trained by Adam optimizer with a learning rate  
482 of 0.001. These two training processes were alternated for 300 iterations, and the child  
483 architecture with the best reward in the last controller step was extracted.

484  
485 Sampled architectures were generated by sampling the computational operations uniformly and  
486 sampling the residual connections at the proportion of 0.4 as used in searched models. Finally,  
487 the child models of searched and sampled were trained from scratch to convergence using  
488 identical setup to facilitate downstream comparisons. Convergence was defined as validation  
489 AUROC not increasing for at least 10 epochs. To more robustly measure the accuracy of

490 AMBER, we ran the search and sample processes for six times, respectively. Throughout the  
491 manuscript, all processing and analysis of searched and sampled models were strictly identical,  
492 except for how we derived their corresponding architectures. We referred to the searched model  
493 with best testing performance as AMBER-Seq and referred to the sampled model with best  
494 testing performance as AMBER-Base.

495

496

### 497 **Dataset for transcriptional regulatory activity prediction**

498 The generic tasks of interest in this study were to predict transcriptional regulatory activity for a  
499 given DNA sequence. We aimed to design an end-to-end convolutional neural network model  
500 that takes raw one-hot encoded DNA as input. Following the previous work<sup>4</sup>, we used the pre-  
501 compiled training, validation and testing dataset downloaded from  
502 <http://deepsea.princeton.edu/help/>. The inputs were one-hot encoded matrices of DNA  
503 sequences built on hg19 reference human genome assembly. The training labels were compiled  
504 from a large compendium of publicly available ChIP-seq datasets, which measure the genome-  
505 wide molecular profiles such as protein binding or chemical modifications using high-throughput  
506 sequencing. In total, there are 919 distinct labels for ChIP-seq profiles of transcription factor  
507 binding, histone modification, and DNase accessibility assays in diverse human cell lines and  
508 tissues; and there are 4400000 training samples, 8000 validation samples and 455024 testing  
509 samples, each of 1000 bp (1000 x 4 when one-hot encoded) in length.

510

511

### 512 **Allele-specific binding analysis**

513 A compendium of allele-specific transcription factor binding sites reported previously<sup>24</sup> were  
514 compiled for benchmarking the variant effect predictions of the AMBER searched models.  
515 Briefly, ChIP-seq data were collected that measured genome-wide binding profiles for 83 unique  
516 transcription factors. For each binding site, binomial test was performed to test allelic imbalance  
517 and Benjamini-Hochberg False Discovery Rate (FDR) was used to correct for multiple testing.  
518 The baseline machine learning methods and the motif scorings were computed previously<sup>24</sup>. We  
519 further divided the variants into loss-of-binding alleles (reference reads ratio>0.6 and  
520 FDR<0.01), gain-of-binding alleles (reference reads ratio<0.4 and FDR<0.01), and neutral  
521 alleles (FDR>0.9).

522

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

531

532 For the biological case study of variant effect prediction on SNP rs11658786, we reported its  
533 variant effect predictions from AMBER-Seq and AMBER-Base along with available baseline  
534 variant scoring methods<sup>24</sup>. Variants in high LD with the allele-specific variant of interest were  
535 queried from LDlink webserver<sup>38</sup> (<https://ldlink.nci.nih.gov/>) using the EUR/CEU population and

536  $R^2 > 0.9$ . Then the set of variants were processed by plink<sup>39</sup> (v1.90) and plotted by R package  
537 *gaston*<sup>40</sup>. The eQTLs for allele-specific variants were queried using the GTEx web portal<sup>41</sup>  
538 (<https://www.gtexportal.org/home/>).

539

540

#### 541 **GWAS analysis**

542 To evaluate the informativeness of the variant annotations from different model architectures, we  
543 used stratified LD-score regression<sup>34</sup> to assess the heritability enrichment for variant annotations.

544 First, we downloaded the summary statistics files from UK Biobank for disease phenotypes  
545 reported previously<sup>35</sup>. *Selene*<sup>17</sup> (v0.4.2) was employed to process the genome-wide variant effect

546 predictions for SNPs from the 1000 Genome Project (European cohort) for each transcriptional  
547 regulatory feature in both AMBER-designed AMBER-Seq model and uniformly-sampled

548 AMBER-Base model. Then the variant effect predictions were subsequently converted to LD  
549 scores and regressed on the  $\chi^2$  statistics using *ldsc* v1.0.1 Python implementation

550 (<https://github.com/bulik/ldsc>), conditioned on a set of 97 baseline LD annotations from

551 *baselineLD* v2.2 (<https://data.broadinstitute.org/alkesgroup/LDSCORE/>). We restricted our

552 analyses for phenotypes with the ratio statistics less than 10% to avoid potential model

553 misspecifications<sup>34</sup>. The enrichment P-values were computed by *ldsc* and corrected for multiple

554 testing by Benjamini-Hochberg FDR. Regulatory features whose variant annotations were

555 significant (FDR < 0.05) in either the searched AMBER-Seq or the sampled AMBER-Base

556 models were analyzed for their overlapping statistics and enrichment fold-changes across

557 models.

558

559

## 560 **Data Availability**

561 All data used in this study are publicly available and the URLs are provided in the corresponding  
562 sections in Methods.

563

## 564 **Code Availability**

565 The AMBER package is available at GitHub: <https://github.com/zj-zhang/AMBER> ; the analysis  
566 presented in this study is available at <https://github.com/zj-zhang/AMBER-Seq>

567

568

569

## 570 **References**

- 571 1. Eraslan, G., Avsec, Ž., Gagneur, J. & Theis, F. J. Deep learning: new computational  
572 modelling techniques for genomics. *Nat. Rev. Genet.* 1 (2019). doi:10.1038/s41576-019-  
573 0122-6
- 574 2. Ching, T. *et al.* Opportunities and obstacles for deep learning in biology and medicine. *J.*  
575 *R. Soc. Interface* **15**, 20170387 (2018).
- 576 3. Bengio, Y. *Convolutional Networks for Images, Speech, and Time-Series.* (1997).
- 577 4. Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep  
578 learning-based sequence model. *Nat. Methods* **12**, 931–934 (2015).
- 579 5. Kelley, D. R., Snoek, J. & Rinn, J. L. Basset: learning the regulatory code of the  
580 accessible genome with deep convolutional neural networks. *Genome Res.* **26**, 990–9  
581 (2016).

- 582 6. Alipanahi, B., DeLong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence  
583 specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.* **33**,  
584 831–838 (2015).
- 585 7. Jaganathan, K. *et al.* Predicting Splicing from Primary Sequence with Deep Learning. *Cell*  
586 **176**, 535-548.e24 (2019).
- 587 8. Zhou, J. *et al.* Deep learning sequence-based ab initio prediction of variant effects on  
588 expression and disease risk. *Nat. Genet.* **50**, 1171–1179 (2018).
- 589 9. Zhou, J. *et al.* Whole-genome deep-learning analysis identifies contribution of noncoding  
590 mutations to autism risk. *Nat. Genet.* **51**, 973–980 (2019).
- 591 10. Kelley, D. R. *et al.* Sequential regulatory activity prediction across chromosomes with  
592 convolutional neural networks. *Genome Res.* **28**, 739–750 (2018).
- 593 11. Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale  
594 Image Recognition. *Int. Conf. Learn. Represent.* 1–14 (2014).
- 595 12. Chollet, F. Xception: Deep learning with depthwise separable convolutions. in  
596 *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*  
597 *2017 2017-January*, 1800–1807 (Institute of Electrical and Electronics Engineers Inc.,  
598 2017).
- 599 13. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. in  
600 *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern*  
601 *Recognition 2016-December*, 770–778 (IEEE Computer Society, 2016).
- 602 14. Zoph, B. & Le, Q. V. Neural Architecture Search with Reinforcement Learning. (2016).
- 603 15. Pham, H., Guan, M. Y., Zoph, B., Le, Q. V. & Dean, J. Efficient Neural Architecture  
604 Search via Parameter Sharing. (2018).

- 605 16. Avsec, Ž. *et al.* The Kipoi repository accelerates community exchange and reuse of  
606 predictive models for genomics. *Nature Biotechnology* **37**, 592–600 (2019).
- 607 17. Chen, K. M., Cofer, E. M., Zhou, J. & Troyanskaya, O. G. Selene: a PyTorch-based deep  
608 learning library for sequence data. *Nat. Methods* **16**, 315–318 (2019).
- 609 18. Real, E., Aggarwal, A., Huang, Y. & Le, Q. V. Regularized Evolution for Image Classifier  
610 Architecture Search. *Proc. AAAI Conf. Artif. Intell.* **33**, 4780–4789 (2019).
- 611 19. Liu, H., Simonyan, K. & Yang, Y. DARTS: Differentiable Architecture Search. (2018).
- 612 20. He, X., Zhao, K. & Chu, X. AutoML: A Survey of the State-of-the-Art. (2019).
- 613 21. Lee, H., Grosse, R., Ranganath, R. & Ng, A. Y. Convolutional deep belief networks for  
614 scalable unsupervised learning of hierarchical representations. in *Proceedings of the 26th*  
615 *International Conference On Machine Learning, ICML 2009* 609–616 (ACM Press,  
616 2009). doi:10.1145/1553374.1553453
- 617 22. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning Transferable Architectures for  
618 Scalable Image Recognition. (2017).
- 619 23. Yu, F. & Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. *4th Int.*  
620 *Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.* (2015).
- 621 24. Wagih, O., Merico, D., DeLong, A. & Frey, B. J. Allele-specific transcription factor  
622 binding as a benchmark for assessing variant impact predictors. *bioRxiv* 253427 (2018).  
623 doi:10.1101/253427
- 624 25. Lee, D. *et al.* A method to predict the impact of regulatory variants from DNA sequence.  
625 *Nat. Genet.* **47**, 955–961 (2015).
- 626 26. Bryne, J. C. *et al.* JASPAR, the open access database of transcription factor-binding  
627 profiles: new content and tools in the 2008 update. *Nucleic Acids Res.* **36**, D102-6 (2008).

- 628 27. Machanick, P. & Bailey, T. MEME-ChIP: motif analysis of large DNA datasets.  
629 *Bioinformatics* (2011).
- 630 28. Zhang, P. *et al.* Negative cross-talk between hematopoietic regulators: GATA proteins  
631 repress PU.1. *Proc. Natl. Acad. Sci. U. S. A.* **96**, 8705–8710 (1999).
- 632 29. Metcalf, D. *et al.* Inactivation of PU.1 in adult mice leads to the development of myeloid  
633 leukemia. *Proc. Natl. Acad. Sci. U. S. A.* **103**, 1486–1491 (2006).
- 634 30. Wang, F. & Tong, Q. Transcription factor PU.1 is expressed in white adipose and inhibits  
635 adipocyte differentiation. *Am. J. Physiol. Physiol.* **295**, C213–C220 (2008).
- 636 31. Lin, L. *et al.* Adipocyte expression of PU.1 transcription factor causes insulin resistance  
637 through upregulation of inflammatory cytokine gene expression and ROS production. *Am.*  
638 *J. Physiol. - Endocrinol. Metab.* **302**, E1550 (2012).
- 639 32. Buniello, A., MacArthur, J. & ... M. C. The NHGRI-EBI GWAS Catalog of published  
640 genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic*  
641 *Acids Res* (2019).
- 642 33. Bycroft, C. *et al.* The UK Biobank resource with deep phenotyping and genomic data.  
643 *Nature* **562**, 203–209 (2018).
- 644 34. Finucane, H. K. *et al.* Partitioning heritability by functional annotation using genome-  
645 wide association summary statistics. *Nat. Genet.* **47**, 1228–1235 (2015).
- 646 35. Loh, P. R., Kichaev, G., Gazal, S., Schoech, A. P. & Price, A. L. Mixed-model association  
647 for biobank-scale datasets. *Nature Genetics* **50**, 906–908 (2018).
- 648 36. Zhang, Z., Zhou, L., Gou, L. & Wu, Y. N. Neural Architecture Search for Joint  
649 Optimization of Predictive Power and Biological Knowledge. (2019).
- 650 37. Williams, R. J. *Simple Statistical Gradient-Following Algorithms for Connectionist*



- 651            *Reinforcement Learning*. Springer **8**, (1992).
- 652    38.    Machiela, M. & Chanock, S. LDlink: a web-based application for exploring population-  
653            specific haplotype structure and linking correlated alleles of possible functional variants.  
654            *Bioinformatics* (2015).
- 655    39.    Purcell, S. *et al.* PLINK: A tool set for whole-genome association and population-based  
656            linkage analyses. *Am. J. Hum. Genet.* **81**, 559–575 (2007).
- 657    40.    Package ‘gaston’ Type Package Title Genetic Data Handling (QC, GRM, LD, PCA) &  
658            Linear Mixed Models. (2020). doi:10.1159/000488519
- 659    41.    Lonsdale, J. *et al.* The Genotype-Tissue Expression (GTEx) project. *Nature Genetics* **45**,  
660            580–585 (2013).
- 661