RESEARCH ARTICLE SUMMARY

GENERATIVE GENOMICS

Sequence modeling and design from molecular to genome scale with Evo

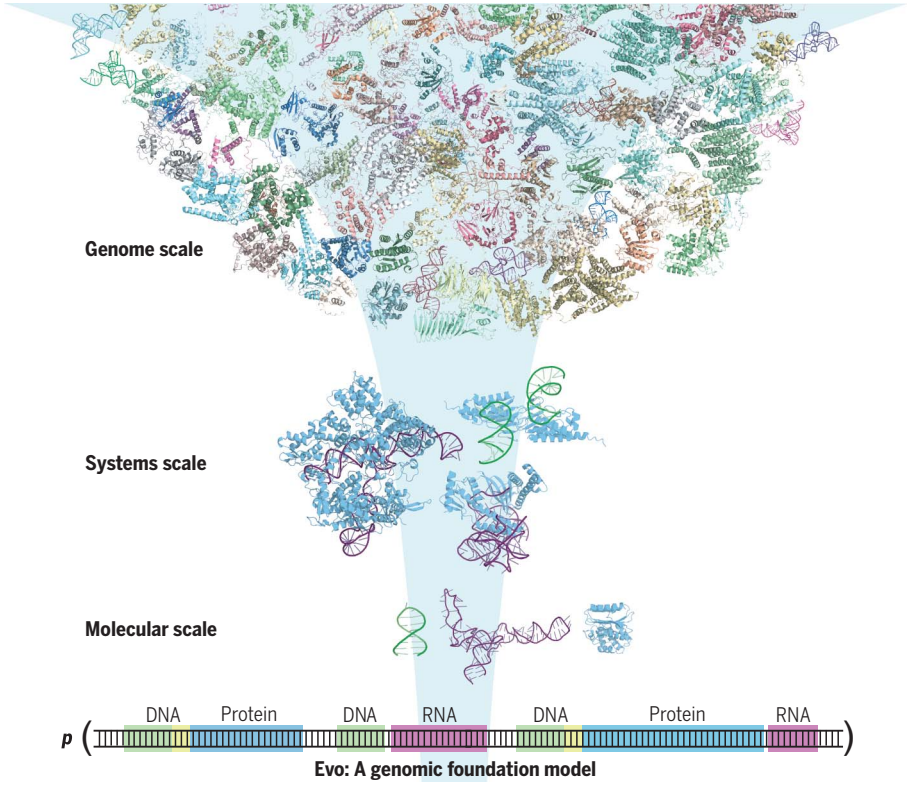
Eric Nguyen†, Michael Poli†, Matthew G. Durrant†, Brian Kang†, Dhruva Katrekar†, David B. Li†, Liam J. Bartie, Armin W. Thomas, Samuel H. King, Garyk Brixi, Jeremy Sullivan, Madelena Y. Ng, Ashley Lewis, Aaron Lou, Stefano Ermon, Stephen A. Baccus, Tina Hernandez-Boussard, Christopher Ré, Patrick D. Hsu*, Brian L. Hie*

INTRODUCTION: The fundamental instructions of life are encoded in the DNA sequences of all living organisms. Understanding these instructions could unlock deeper insights into biological processes and enable new ways to reprogram biology into useful technologies. However, even the simplest microbial genomes are incredibly complex, with millions of DNA base pairs encoding the interplay of DNA, RNA, and proteins—the three modalities of the so-called central dogma of molecular biology and the key actors in cellular function. This complexity exists at multiple scales, from individual molecules to whole genomes, representing a vast landscape of genetic information that has been functionally selected over evolutionary time.

RATIONALE: Rapid progress in artificial intelligence (AI) has led to large language models that demonstrate increasingly advanced multitask reasoning and generation capabilities when trained on massive amounts of data. However, technological limitations in the architecture of these models have restricted efforts to apply them to biology at a similar scale. Current approaches struggle to analyze sequences at the individual character level and are computationally demanding when applied to long sequences. An advanced model maintaining single-nucleotide resolution over large genomic sequences could potentially extract functional information about the complex molecular interactions that are embedded in the patterns of natural evolutionary variation.

RESULTS: In this work, we present Evo, a genomic foundation model that enables prediction and generation tasks from the molecular to the genome scale. Using an architecture based on advances in deep signal processing, we scaled Evo to 7 billion parameters with a context length of 131 kilobases at single-nucleotide resolution. We report scaling laws on DNA, complementing similar observations in natural language and vision. Trained on 2.7 million prokaryotic and phage genomes, Evo demonstrates zero-shot function prediction across DNA, RNA, and protein modalities that is competitive with—or outperforms—domain-specific language models. Evo also excels at multimodal generation tasks, which we demonstrated by generating synthetic CRISPR-Cas molecular complexes and transposable systems. We experimentally validated the functional activity of Evo-generated CRISPR-Cas molecular complexes as well as IS200 and IS605 transposable systems, representing the first examples of protein-RNA and protein-DNA codesign with a language model. Using information learned over whole genomes, Evo learns how small changes in nucleotide sequence affect whole-organism fitness and can generate DNA sequences with plausible genomic architecture more than 1 megabase in length.

CONCLUSION: Evo is a foundation model that is designed to capture two fundamental aspects of biology: the multimodality of the central dogma and the multiscale nature of evolution. The central dogma integrates DNA, RNA, and proteins with a unified code and predictable information flow, whereas evolution unifies the vastly different length scales of biological function represented by molecules, pathways, cells, and organisms. Evo learns both of these representations from the whole-genome sequences of millions of organisms to enable prediction and design tasks from the molecular to genome scale. Further development of large-scale biological sequence models like Evo, combined with advances in DNA synthesis and genome engineering, will accelerate our ability to engineer life. ■



Evo, a 7-billion-parameter genomic foundation model, learns biological complexity from individual nucleotides to whole genomes. Trained on 2.7 million raw prokaryotic and phage genome sequences, Evo is naturally multimodal, enabling the codesign of DNA, RNA, and protein molecules that form higher-order functional systems. Evo is also inherently multiscale, enabling prediction and generation tasks at the level of molecules, systems, and genomes.

The list of author affiliations is available in the full article online.
*Corresponding author. Email: brianhie@stanford.edu (B.L.H.); patrick@arcinstitute.org (P.D.H.)
†These authors contributed equally to this work.
Cite this article as E. Nguyen *et al.*, *Science* **386**, eado9336 (2024). DOI: 10.1126/science.ado9336

READ THE FULL ARTICLE AT
https://doi.org/10.1126/science.ado9336

RESEARCH ARTICLE

GENERATIVE GENOMICS

Sequence modeling and design from molecular to genome scale with Evo

Eric Nguyen[1,2]†, Michael Poli[3,4]†‡, Matthew G. Durrant[1]†, Brian Kang[1,2]†, Dhruva Katrekar[1]†, David B. Li[1,2]†, Liam J. Bartie[1], Armin W. Thomas[5], Samuel H. King[1,2], Garyk Brixi[1,6], Jeremy Sullivan[1], Madelena Y. Ng[7], Ashley Lewis[8], Aaron Lou[3], Stefano Ermon[3,9], Stephen A. Baccus[10], Tina Hernandez-Boussard[8], Christopher Ré[3], Patrick D. Hsu[1,11]*, Brian L. Hie[1,5,12]*

The genome is a sequence that encodes the DNA, RNA, and proteins that orchestrate an organism's function. We present Evo, a long-context genomic foundation model with a frontier architecture trained on millions of prokaryotic and phage genomes, and report scaling laws on DNA to complement observations in language and vision. Evo generalizes across DNA, RNA, and proteins, enabling zero-shot function prediction competitive with domain-specific language models and the generation of functional CRISPR-Cas and transposon systems, representing the first examples of protein-RNA and protein-DNA codesign with a language model. Evo also learns how small mutations affect whole-organism fitness and generates megabase-scale sequences with plausible genomic architecture. These prediction and generation capabilities span molecular to genomic scales of complexity, advancing our understanding and control of biology.

DNA is the fundamental layer of biological information that is responsible for transmitting the results of evolution across generations of life (*1*–*3*). Evolutionary variation in genome sequences reflects adaptation and selection for biological function at the phenotypic level (*4*). Rapid advances in DNA sequencing technologies have enabled the systematic mapping of this evolutionary diversity at the whole-genome scale.

A machine that learns this breadth of information across genomes could model the function of DNA, RNA, and proteins as well as their diverse interactions that orchestrate complex biological functions, mediate disease, or create a complete organism. Modern machine learning algorithms combined with massive datasets of genomic sequences could enable a general biological foundation model that learns the intrinsic logic of whole genomes.

However, current efforts to model molecular biology with machine learning have been focused on creating modality-specific models that are specialized to proteins, coding sequences, RNA, or regulatory DNA (*5*–*9*). In addition, generative applications in biology have been limited to the design of single molecules, simple complexes (*10*–*12*), or short DNA sequences (*13*, *14*). By contrast, complex biological processes, such as gene regulation, CRISPR immunity, or genetic transposition, rely on many interactions involving molecules across multiple modalities.

A DNA model that unifies information across the molecular, systems, and genome scales could learn from large genomic regions to capture systems-wide interactions and enable the design of more-sophisticated biological functions. By operating at single-nucleotide resolution, this model would be able to incorporate the evolutionary effects of sequence variation, such as individual single-nucleotide mutations, that can completely alter organism function.

Inspired by the recent success of large language models, many approaches have applied similar modeling techniques to biological sequences. However, existing attempts to model DNA as a language (*15*–*17*) are limited by the prevailing dense Transformer architecture, which incurs high computational cost as input sequence lengths grow relative to model width (scaling quadratically) and generally underperforms at single-nucleotide or byte-level resolution compared with models trained at coarser resolutions (*18*). Recent algorithmic advances in extending context length of attention-based models (*19*, *20*) have similar resolution limitations. As a result, Transformer-based DNA models are constrained to short context lengths and use schemes that aggregate nucleotides into the basic units of language models, called tokens, thereby sacrificing single-nucleotide resolution (*15*, *16*, *21*–*23*).

We present Evo, a 7-billion-parameter genomic foundation model trained to generate DNA sequences at whole-genome scale. Evo uses a context length of 131,072 tokens and is based on the StripedHyena architecture (*24*), which hybridizes attention and data-controlled convolutional operators to efficiently process and recall patterns in long sequences. Evo is trained on a prokaryotic whole-genome dataset consisting of 300 billion nucleotides and uses a byte-level, single-nucleotide tokenizer. By conducting a scaling laws analysis for DNA pretraining, we observe predictable performance gain with larger scale.

We demonstrate that Evo can be used in both prediction and generation tasks at the molecular, systems, and genome scale. In zero-shot evaluations, Evo is competitive with protein language models at predicting the fitness effects of mutations on bacterial proteins, outperforms RNA language models in predicting fitness effects of mutations on noncoding RNAs (ncRNAs), and predicts how regulatory DNA sequence composition controls gene expression. Evo also learns the coevolutionary linkage of coding and noncoding sequences to design functional biological systems including CRISPR-Cas ribonucleoprotein complexes and transposable elements, requiring codesign of protein-RNA and protein-DNA systems, respectively.

At the whole-genome scale, Evo understands how small mutations in genomes affect organismal fitness, indicating its ability to learn aspects of gene function within a broader genomic context. We also use Evo to generate genome-scale sequences with plausible high-level architecture more than 1 megabase (Mb) in length, a scale that is orders of magnitude greater than previous methods (*10*, *13*, *14*). Taken together, Evo establishes a foundational paradigm for predictive and generative biological modeling (Fig. 1A) that could enable a deeper understanding of biology and accelerate our ability to engineer life.

Modeling long sequences at nucleotide resolution with the StripedHyena architecture

Evo is a genomic foundation model with 7 billion parameters trained with a context length of up to 131,072 tokens, using single-nucleotide, byte-level tokenization. To model long sequences at nucleotide resolution efficiently, we leveraged the StripedHyena architecture (*24*) (Fig. 1B) that builds on emerging techniques in deep signal processing (*25*–*28*). The model is a hybrid of 29 layers of data-controlled convolutional operators (hyena layers) interleaved with three layers (10%) of multihead attention equipped with rotary position embeddings (RoPEs) (*29*) (table S1 and Materials and methods).

Hyena layers process sequences in an input-dependent manner using compositions of short

[1]Arc Institute, Palo Alto, CA, USA. [2]Department of Bioengineering, Stanford University, Stanford, CA, USA. [3]Department of Computer Science, Stanford University, Stanford, CA, USA. [4]TogetherAI, San Francisco, CA, USA. [5]Stanford Data Science, Stanford University, Stanford, CA, USA. [6]Department of Genetics, Stanford University, Stanford, CA, USA. [7]Stanford Center for Biomedical Informatics Research, Stanford, CA, USA. [8]Department of Biomedical Data Science, Stanford University, Stanford, CA, USA. [9]CZ Biohub, San Francisco, CA, USA. [10]Department of Neurobiology, Stanford University, Stanford, CA, USA. [11]Department of Bioengineering and Center for Computational Biology, University of California, Berkeley, Berkeley, CA, USA. [12]Department of Chemical Engineering, Stanford University, Stanford, CA, USA.
***Corresponding author. Email: brianhie@stanford.edu (B.L.H.); patrick@arcinstitute.org (P.D.H.)**
†These authors contributed equally to this work.
‡Present address: Liquid AI, Cambridge, MA, USA.

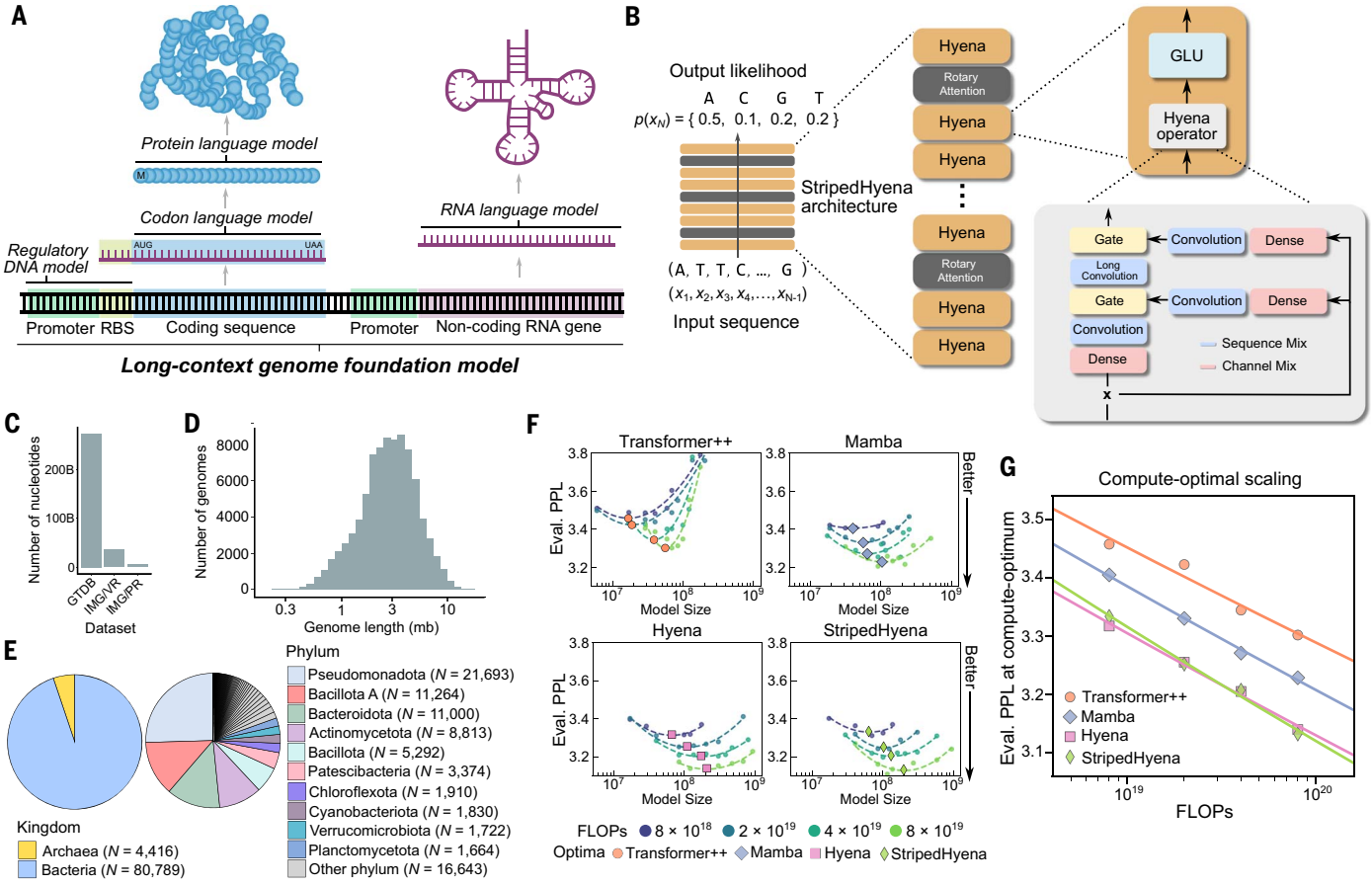


Fig. 1. Pretraining a genomic foundation model across prokaryotic life. (**A**) A model of genome sequences at single-nucleotide resolution could learn all of the information encoded in regulatory DNA and in the sequences of the other modalities within the central dogma (proteins, coding RNA, and ncRNA). Even further, it could learn covariation involving multiple genes and regulatory elements. The status of DNA as the fundamental layer of biological information makes it a productive modality at which to develop a biological foundation model. (**B**) A model that predicts the likelihood of the next token given a sequence of tokens, referred to as autoregressive modeling, can learn complex patterns underlying DNA sequences. StripedHyena is a deep signal processing architecture for long sequences, obtained by hybridizing attention and hyena operators. GLU, gated linear units. (**C**) We pretrained Evo, a 7-billion-parameter model with the StripedHyena architecture, on bacterial genome sequences from GTDB and IMG/PR and viral sequences from IMG/VR, excluding sequences from viruses that infect eukaryotic hosts. (**D**) A histogram depicting the sequence length of the genomes in GTDB. mb, megabases. (**E**) Pie charts depicting the taxonomic makeup of GTDB based on the kingdom (left) and phylum (right). (**F**) Results from a first-of-its-kind scaling laws analysis for large-scale DNA pretraining. Models improve monotonically with scale, with significant differences between architectures. Eval. PPL, evaluation perplexity. (**G**) To determine optimal architecture and scaling for Evo, we compared scaling rates of different models pretrained on the compute-optimal frontier, i.e., with optimal allocation of compute between dataset size and model size.

and long convolution filters (Fig. 1B), making the layer especially effective at filtering noisy patterns that can occur in DNA and at aggregating individual nucleotides into motifs. Model hybridization, first proposed to address shortcomings of state-space models (*30*–*32*), has recently been shown to improve scaling performance on language modeling of both standalone Hyena and Transformer architectures (*24*). Compared with HyenaDNA (*33*), a previous generation of DNA models leveraging a Hyena architecture (*34*), Evo is based on an improved hybrid design and scaled to 1000× larger model size and 100× more data.

Training Evo at scale on OpenGenome

We compiled a large genome dataset called OpenGenome (Materials and methods) with more than 80,000 bacterial and archaeal genomes and millions of predicted phage and plasmid sequences, covering 300 billion nucleotide tokens (Fig. 1, C to E; fig. S1; and table S2) (*35*–*37*). For safety considerations, we excluded viral genomes that infect eukaryotic hosts. Like most language models, Evo is pretrained using a next-token prediction objective on raw genome sequences with no explicit supervision or annotations. Pretraining involved a first stage using a context length of 8192 tokens and a second context-extension stage using a context length of 131,072 tokens.

StripedHyena demonstrates favorable scaling laws on DNA sequence data

Aiding our model design, we performed a scaling laws analysis for DNA sequence modeling to determine the relationship between training, architectural details, and performance metrics through a systematic experimental protocol (*38*, *39*). Once a set of scaling laws is obtained, it can then be used as a guide to optimally scale training to larger models and datasets.

We compare different classes of architectures using a compute-optimal protocol, aimed at evaluating results on the compute-optimal frontier (Materials and methods). We trained more than 300 models across four architectures: Transformer++, Mamba, Hyena, and StripedHyena (table S3). Transformer++ is a state-of-the-art Transformer, and Mamba is a modern architecture using data-controlled state-space models (*40*).

We found Transformer++ to yield substantially worse perplexity (a measure of next token prediction quality) at all compute budgets (Fig. 1, F and G), a symptom of the inefficiency of the architecture at the byte resolution. Both state-space

and deep signal processing architectures had an improved scaling rate over Transformer++, with Hyena and StripedHyena resulting in the best scaling rate. We observed stable training for StripedHyena throughout all the studied model sizes and learning rates during the scaling analysis.

We also compare architecture performance outside the compute-optimal frontier, namely with allocations of the computational budget that may be suboptimal. Performance outside the compute-optimal frontier is important in practice, as most models (including Evo) are trained for more tokens than recommended by compute-optimal scaling laws. We estimate 250 billion to be the compute-optimal number of tokens for Evo 7B given the floating point operation (FLOP) budget, meaning the model was trained at a 17% offset from the compute-optimal model size during the initial 8192 sequence length pretraining phase of 300 billion tokens. Both Transformer++ and Mamba experienced numerical instability during training and suffered from a higher performance degradation of the scaling rate outside the compute-optimal frontier, in contrast to StripedHyena (figs. S3 to S7). These findings motivate the choice of StripedHyena as the architecture for Evo.

Evo learns across DNA, RNA, and protein modalities

Predicting mutational effects on protein function

Beyond evaluating perplexity, we next investigated the model's zero-shot performance on biologically relevant downstream tasks. For example, language models specifically trained on large corpuses of protein sequences or nucleotide coding sequences have demonstrated an ability to predict mutational effects on protein function (*41*–*43*) without any task-specific fine-tuning or supervision. Because Evo's training data contains protein coding sequences, we tested whether the model could also perform zero-shot protein function prediction. Notably, Evo is trained on genomic sequences without any explicit coding sequence annotations.

Following work in evaluation of protein language models, we leveraged deep mutational scanning (DMS) studies, which introduce an exhaustive set of mutations to a protein coding sequence and then experimentally measure the effects of these mutations on various fitness metrics, which quantify functional activity (*42*, *44*, *45*). The language-model likelihood or pseudolikelihood (Materials and methods) of the amino acid sequence is used to predict the experimental fitness score (Fig. 2A). To adapt this task to nucleotide sequences, we use the wild-type coding sequence and nucleotide mutations reported in the original DMS studies (Materials and methods).

On DMS datasets of prokaryotic proteins, Evo's zero-shot performance exceeded all other nucleotide models tested (Fig. 2B and table S4), including GenSLM (*15*)—a model explicitly trained only on coding sequences with a codon vocabulary (Fig. 1A). Evo also reaches competitive performance with leading protein-specific language models (*41*, *46*–*48*) (Fig. 2B). Previous work has shown that improvement beyond this performance range is difficult for protein language models with self-supervised pretraining alone (*49*), indicating that Evo is already competitive with state-of-the-art protein language modeling on bacterial proteins. On DMS datasets of human proteins, Evo is unable to predict mutational effects on fitness (fig. S8A and table S5), most likely because the pretraining dataset is composed of prokaryotic sequences. However, we observed a strong association between language-model perplexity on the wild-type sequence and fitness prediction performance (fig. S8B), which indicates that additional fine-tuning or future pretraining on mammalian coding sequences could improve Evo's performance beyond bacterial proteins.

Predicting mutational effects on ncRNA function

Next, we tested whether the same pretrained model could learn functional information about ncRNAs, such as tRNAs, ribosomal RNAs (rRNAs), and ribozymes. We collected ncRNA DMS datasets (Materials and methods) and evaluated Evo's ability to perform zero-shot ncRNA fitness prediction using the results of experimental ncRNA DMS studies as the ground truth score (Fig. 2C).

We found that Evo again outperforms all other tested nucleotide language models at this task, including RNA-FM (*50*), an RNA language model that is explicitly trained on ncRNA sequences (Fig. 2D and table S6). We observed especially strong predictive performance on a study that measured the effects of mutations to the 5*S* rRNA on the growth rate of *Escherichia coli* (Spearman correlation coefficient $r = 0.60$, two-sided t-distributed $P = 1.9 \times 10^{-3}$) (*51*). Beyond protein sequences, these results demonstrate that Evo can learn mutational effects on ncRNA function.

Predicting activity of regulatory DNA

Given that Evo's training also contains prokaryotic regulatory DNA sequences, we investigated whether Evo has learned information that is useful for regulatory DNA tasks. We focused on predicting gene expression from promoter sequences and protein expression from sequences of ribosome-binding sites (RBSs) (Fig. 2E).

For supervised promoter activity prediction, we followed a previous study (*52*) in which a regression model is developed using train and validation splits from a single study, and the final model is then tested on promoter datasets from other studies to assess out-of-domain generalizability (Materials and methods). We used the three test datasets from LaFleur *et al.* (*52*–*55*) and a dataset in which Kosuri *et al.* constructed ~12,000 combinations of common promoters and RBSs and measured the corresponding mRNA expression of a reporter gene for each promoter-RBS pair in *E. coli* (*56*).

Evo's zero-shot likelihoods had non-negligible correlation with promoter activity across these four studies (mean Spearman $r = 0.43$). These correlations also exceed those of the sequence guanine-cytosine (GC) content (mean Spearman $r = 0.35$) and the zero-shot likelihoods of GenSLM (mean Spearman $r = 0.09$) (Fig. 2F and table S7). We also trained two supervised models, a ridge regression linear model and a convolutional neural network (CNN), on either Evo embeddings or one-hot-encoded sequence. The CNN architecture substantially outperformed ridge regression across both embeddings, and the Evo embeddings substantially outperformed one-hot embeddings across both architectures (Fig. 2F and table S7). Notably, even zero-shot Evo likelihoods had comparable predictive performance (mean Spearman $r = 0.43$) to a CNN trained on one-hot embeddings (mean Spearman $r = 0.44$), which indicates that Evo's pretraining contributes useful information to function prediction. Combining the Evo embeddings with a supervised CNN architecture (mean Spearman $r = 0.56$) also approached the performance of Promoter Calculator (*52*), a state-of-the-art method for promoter activity prediction (mean Spearman $r = 0.62$). These results indicate that Evo has learned sequence-intrinsic information that is a useful correlate of promoter activity and motivates improving zero-shot learning within the foundation model to improve downstream performance in specific, supervised tasks.

For protein expression prediction, we used the dataset collected by Kosuri *et al.* (*56*), which contains RBSs in addition to promoters and which also measured protein expression in addition to mRNA expression. Evo's zero-shot likelihoods of the RBS sequence alone had weak correlation with protein expression (Spearman $r = 0.17$). However, when concatenating the promoter and RBS sequence together, Evo's zero-shot likelihoods improved substantially (Spearman $r = 0.61$); this correlation is also higher than the zero-shot correlation of just the promoter sequence alone (Spearman $r = 0.47$), which indicates that additional regulatory sequence could provide useful functional context. Evo's zero-shot correlation on promoter-RBS sequences is also higher than the GC content of the promoter-RBS sequences (Spearman $r = 0.47$), zero-shot GenSLM likelihoods (Spearman $r = 0.11$), and RBS Calculator (Spearman $r = 0.39$)—a state-of-the-art protein expression predictor (Fig. 2G) (*57*, *58*).

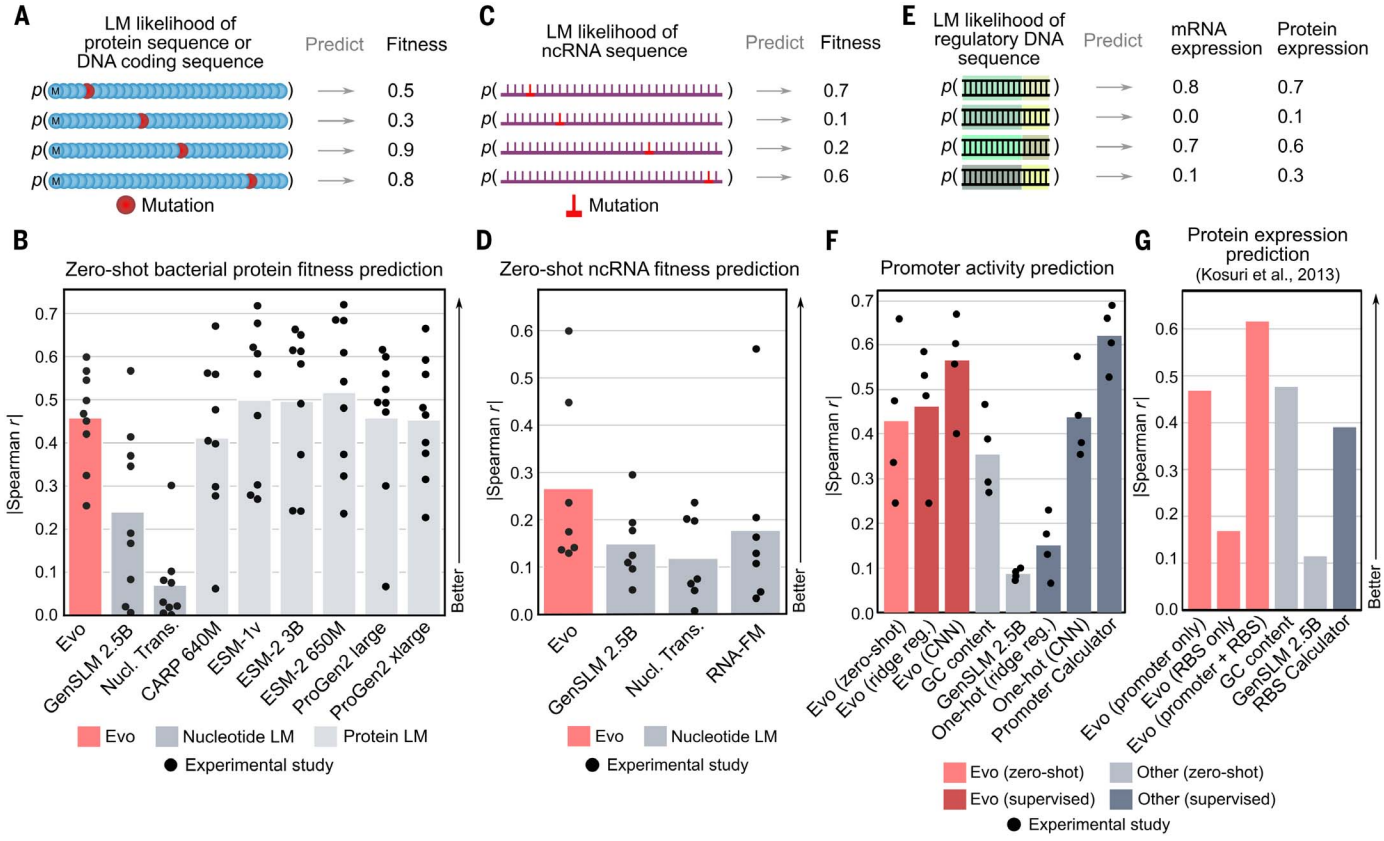


Fig. 2. Evo learns function across proteins, ncRNAs, and regulatory DNA. (**A**) We obtained DMS datasets in which many mutations are made to a protein and a corresponding fitness score is experimentally measured for each protein variant. On the same set of mutated sequences, we compute its likelihood (or pseudolikelihood) under a protein language model or a nucleotide language model (LM). We then correlated these likelihoods with the experimental fitness measurements and used the strength of the correlation to measure the performance of zero-shot function prediction. (**B**) Correlation between zero-shot language model likelihoods or pseudolikelihoods and experimental fitness across nine prokaryotic protein DMS datasets. Bar height indicates the mean; each dot indicates a different DMS study. Nucl. Trans., Nucleotide Transformer. (**C**) We obtained datasets in which many mutations are made to a ncRNA and a corresponding fitness score is experimentally measured. Predictive performance is measured as in the method described in (A). (**D**) Correlation between zero-shot language model likelihoods or pseudolikelihoods and experimental fitness across seven ncRNA DMS datasets. Bar height indicates the mean; each dot indicates a different DMS study. (**E**) We obtained datasets in which many regulatory DNA sequences were measured for their effect on mRNA or protein expression. (**F**) Correlation between promoter activity across four studies and zero-shot language model likelihoods, sequence GC content, or supervised models. The supervised models include ridge regression or a CNN trained on one-hot embeddings or Evo embeddings, as well as a state-of-the-art supervised biophysical model of promoter activity, Promoter Calculator (*52*). Supervised models are evaluated in an out-of-domain prediction setting (Materials and methods). Ridge reg., ridge regression. Bar height indicates the mean; each dot indicates a different promoter activity study. (**G**) We obtained a dataset in which Kosuri *et al*. (*56*) measured protein expression of a gene downstream of ~12,000 promoter-RBS pairs in *E. coli*. When provided with both the promoter and RBS sequences, Evo has higher predictive performance of protein expression compared with zero-shot sequence statistics or a method trained with some supervision to predict protein expression data from mRNA sequence.

Overall, we show how a single model can perform well on tasks that have previously been accomplished by different, domain-specific models. Despite being trained on long genomic sequences without explicit annotations, Evo demonstrates a robust and general understanding of the constitutive protein coding sequences, ncRNA sequences, and regulatory elements.

Generative design of CRISPR-Cas molecular complexes

Next, we reasoned that Evo should be able to generate functional complexes that involve interactions between distinct molecular modalities. In prokaryotes, functionally related genes are generally organized into operons and located next to each other on the genome sequence. Because Evo learns covariation patterns involving any genetic elements within its context window, the model should understand interactions between encoded protein and ncRNA molecules. To demonstrate this capability, we fine-tuned Evo on a dataset of genomic loci containing CRISPR-Cas sequences—molecular machines that consist of protein and ncRNA components that, together, direct adaptive immunity against viral infection (*59*).

The DNA-targeting Cas9 nuclease is typically encoded within 3000 to 4800 base pairs (bp) of coding sequence and found in close genomic proximity to its cognate CRISPR array (*60*). Transcription from the CRISPR array generates noncoding CRISPR RNA (crRNA) molecules that are bound by the Cas protein to generate a functional defense complex that is required for sequence-specific DNA targeting (Fig. 3A). For Cas9 in particular, a second trans-activating CRISPR RNA (tracrRNA) forms a duplex with the crRNA to create a full guide RNA (gRNA). Diverse families of CRISPR-Cas systems are found throughout bacterial and archaeal life, such as Cas12- or Cas13-based systems that target DNA and RNA, respectively (*61*).

We fine-tuned Evo on 72,831 CRISPR-Cas loci extracted from public metagenomic and genomic sequences, adding special prompt tokens for Cas9, Cas12, and Cas13 that were prepended to the beginning of each training

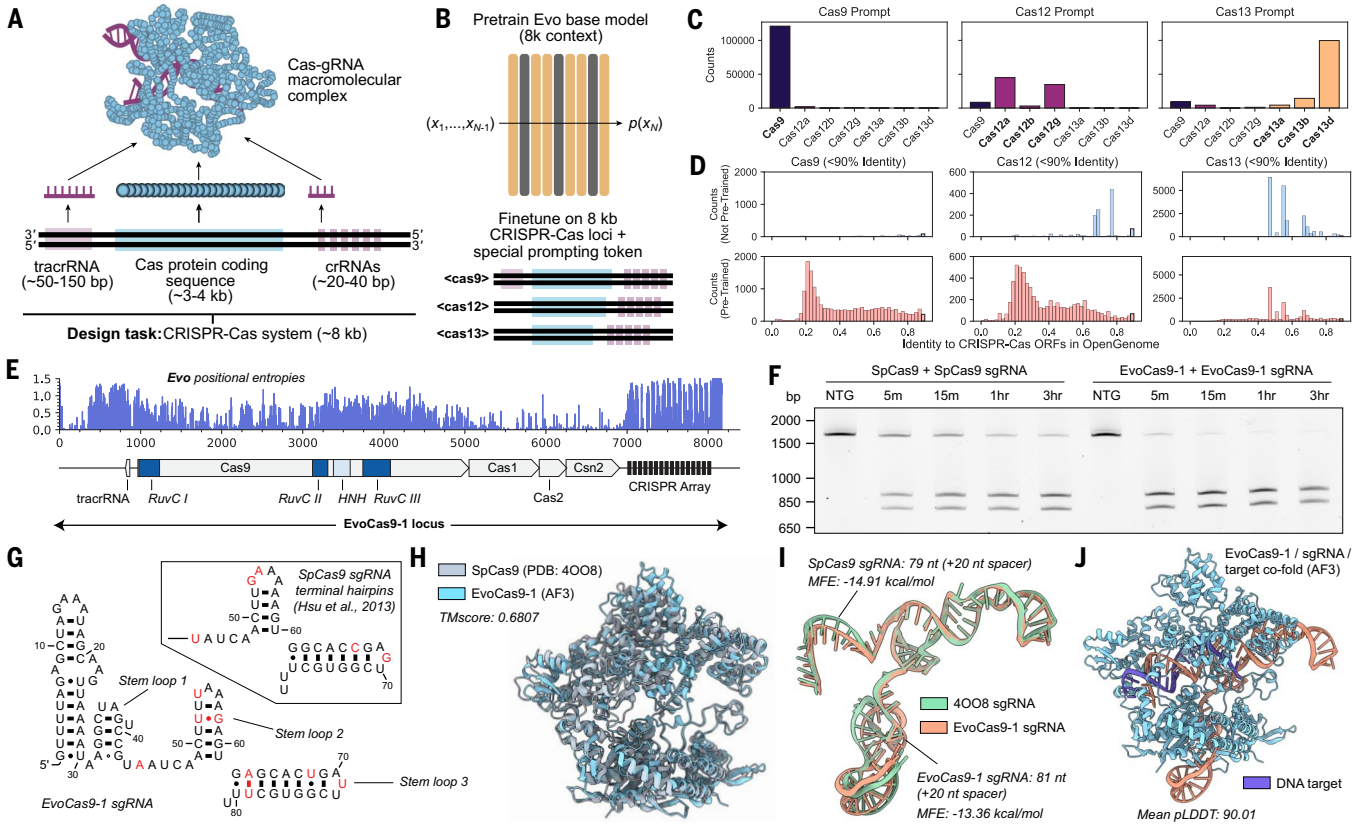


Fig. 3. Fine-tuning on CRISPR-Cas sequences enables generative design of protein-RNA complexes. (**A**) Design task: Generating sequences encoding CRISPR-Cas defense complexes composed of protein and ncRNA components. (**B**) Fine-tuning Evo on 8-kb-length genomic sequences containing CRISPR-Cas systems after its initial 8k pretraining phase. Special conditioning tokens ("cas9," "cas12," or "cas13") prepended to the beginning of each sequence during fine-tuning. (**C**) When prompting with the token for a given type of Cas protein, the most common Cas protein found in the resulting generated sequences corresponds to that token prompt (Materials and methods). (**D**) Histograms representing the distribution of percentage identity of a generated Cas protein sequence to any Cas protein sequence in the training dataset. Samples from a model trained only on CRISPR-Cas sequences (top) and samples from a model fine-tuned on CRISPR-Cas off the base Evo model (bottom). Both models were trained on CRISPR-Cas sequences using the same hyperparameters. (**E**) Annotated core protein-coding genes and ncRNA components found in type II CRISPR systems in the EvoCas9-1 locus as determined by pHMMs and CRISPR ncRNA prediction algorithms. (**F**) Time course results for SpCas9 and EvoCas9-1 cleavage reactions after incubation with cognate sgRNAs and 1 nM DNA target at a 10:10:1 molar ratio of Cas9:sgRNA:target. Nontargeting guide RNA used to verify in vitro cleavage specificity. (**G**) Predicted secondary structure of the sgRNA from the EvoCas9-1 generation. Secondary structure differences between the EvoCas9-1 sgRNA and the SpCas9 sgRNA are highlighted in red. (**H**) AlphaFold3 (AF3) structure prediction of EvoCas9-1 aligned to the crystal structure of SpCas9 (PDB: 4OO8). (**I**) AlphaFold3 (AF3) structure prediction of the EvoCas9-1 sgRNA aligned to the crystal structure (PDB: 4OO8) of the SpCas9 sgRNA (79 nt scaffold + 20 nt spacer). nt, nucleotide. (**J**) AlphaFold3 (AF3) structure prediction of EvoCas9-1 in complex with its codesigned sgRNA (81 nt scaffold + 20 nt spacer).

sequence (Fig. 3B). During sampling, these tokens allow us to guide generation of a specific CRISPR-Cas system type by prompting with the corresponding special token. Sampling 8-kb sequences using each of the three Cas token prompts resulted in coherent generations containing Cas coding sequences and CRISPR arrays corresponding to the expected subtype (Fig. 3C and Materials and methods). Evo generations were classified as Cas9, Cas12, or Cas13 sequences if they contained a CRISPR array detected with the MinCED package and an open reading frame (ORF) that returns a positive hit using a Cas9, Cas12, or Cas13 profile hidden Markov model (pHMM), with a significance threshold of an E value $< 1 \times 10^{-3}$. Sequence alignment with the training dataset revealed that some of the predicted ORFs that returned a positive hit using a Cas9 pHMM also exhibited <40% protein sequence identity to the closest natural Cas9 (Fig. 3D). We also found that the Evo model fine-tuned on CRISPR-Cas loci produces higher quality and more diverse generations across all Cas subtypes compared with a model trained solely on CRISPR-Cas sequences (Fig. 3D and Materials and methods).

Next, we filtered ~2 million Evo-generated sequences for Cas9 loci that contained a Cas9 ORF with RuvC and HNH domains, a CRISPR repeat array, and a detectable tracrRNA sequence (fig. S9), selecting 11 Cas9 systems with robust predicted local distance difference test (pLDDT) scores for functional validation. These samples contain conserved CRISPR-associated genes such as Cas1 and Cas2 involved in CRISPR adaptation, and the positional entropies from the fine-tuned Evo model delimit the boundaries of the protein-coding genes within the locus as well as the noncoding CRISPR repeat motifs (Fig. 3E).

We evaluated the 11 Cas9 generations using an initial in vitro transcription-translation assay followed by the introduction of a DNA target containing an NGG protospacer adjacent motif (PAM) sequence (fig. S14). One of the generations exhibited robust activity, which we named EvoCas9-1. Recombinant expression and purification of EvoCas9-1 paired with chemically synthesized Evo-generated single guide RNA (sgRNA) exhibited comparable in vitro cleavage activity to SpCas9 paired with the canonical SpCas9 sgRNA (Fig. 3F) (*62*, *63*). We further observed that the

Evo-generated sgRNA also improved cleavage efficiency of SpCas9 when compared with a canonical SpCas9 sgRNA (fig. S15 and Materials and methods).

The EvoCas9-1 amino acid sequence shares 79.9% identity with the closest Cas9 in the database of Cas proteins used for model fine-tuning and 73.1% identity with SpCas9. Evo-designed sgRNA is 91.1% identical to the canonical SpCas9 sgRNA and exhibits secondary structure differences in the two terminal stem loops, notably extending the length of stem loops 2 and 3 (Fig. 3G). Although the predicted backbone structure of EvoCas9-1 resembles that of SpCas9, the predicted structure of EvoCas9-1 exhibits a more positive surface charge distribution (Fig. 3H and fig. S16B). The isolated sgRNA structures from the SpCas9 crystal structure and the structure of the EvoCas9-1 sgRNA predicted by the AlphaFold3 model (*64*) show strong agreement in RNA secondary structure (Fig. 3I). The AlphaFold3 cofolded structure prediction for EvoCas9-1 has a high mean pLDDT score of 90 across its protein, RNA, and DNA components (Fig. 3J).

EvoCas9-1 was generated from just 11 codesigns, representing a robust success rate given the complexity of Cas9's multistep mechanism (fig. S14), which requires intricate coordination of protein domains and nucleic acid interactions. Furthermore, the diverse generations were tested on a single NGG PAM, and this sequence preference is known to vary across Cas9 orthologs.

Designing new Cas systems currently relies on mining sequence databases for homologous proteins, where natural evolution provides functional diversity. By leveraging Evo's inherent multimodal capabilities, we can codesign protein-RNA complexes with a single language model, providing a design methodology that can be harnessed across the broad diversity of CRISPR systems and expanding the repertoire of CRISPR technologies beyond what is found in nature.

Generative design of transposon systems

In addition to molecular complexes, Evo learns patterns underlying multigene systems. Mobile genetic elements (MGEs) are biological systems that often contain multiple genes and are found throughout all domains of life. Their opportunistic spread drives sequence variation, new gene function, and even speciation (*65*). The IS200/IS605 family of MGEs spreads through "peel-and-paste" transposition catalyzed by the homodimeric transposase TnpA interacting with terminal hairpins at the left end (LE) and right end (RE) of the element. The insertion sequence (IS) is excised from single-stranded DNA (ssDNA) as a circular product containing an RE-LE junction, which serves as an intermediate for insertion into a new ssDNA target site. IS605 elements additionally contain an RNA-guided TnpB nuclease and a cognate ωRNA that bias the selfish inheritance of the transposable element (Fig. 4A) (*66*–*69*). The ability to generate new MGEs could improve our understanding of their biological function and enable the design of more effective genome engineering tools.

We fine-tuned Evo on 10,720 IS605 elements and 219,866 IS200 elements in their natural sequence context (Fig. 4B and Materials and methods). We next calculated the entropy of the conditional probabilities at each position across natural IS200/IS605 loci (fig. S18) and observed a sharp and sustained increase in entropy corresponding with the 3′ end of the element in particular, indicating that Evo learned a representation of the MGE boundaries. Beyond first-order positional statistics, we also observed that the model learns pairwise relationships between positions in the sequence using a "categorical Jacobian" analysis (*70*), in which we vary the value of each position in the input sequence and measure the resulting changes in the model outputs at all positions. We observed that the model uses information from one end to specify the other end across a distance of ~1 to 2 kb, reflecting the model's understanding of the tight evolutionary linkage of the two terminal elements (fig. S19).

Using special prompt tokens, we used the fine-tuned model to generate IS200 or IS605 elements (fig. S18A). TnpA and TnpB proteins that were detected within these generated sequences varied widely in their distance from the nearest examples in the training set (Fig. 4C), with consistently high ESMFold pLDDT values for predicted structures that were >40 to 50% identity to the training set (fig. S18B) and a sequence length distribution that closely matched proteins in the training set (fig. S18C).

To select sequences for experimental validation, we filtered by similarity to natural systems (ISSpn6, ISStin10, ISHp608, and ISDge10) as well as TnpA protein–level and DNA sequence–level features (fig. S20) and experimentally tested 24 IS200-like and 24 IS605-like designs in vitro. We assay for TnpA-mediated excision and insertion by incubating TnpA protein produced through in vitro transcription-translation with a ssDNA substrate containing the putative left and right ends, followed by a polymerase chain reaction (PCR) with outward-facing primers. If excision occurs, a band is produced from the formation of the RE-LE junction. If the donor contains other target sites and insertion also occurs, bands are produced from the joining of the two ssDNA substrates by the same PCR reaction (Fig. 4D).

We observed that 11 out of 24 Evo-generated IS200-like elements and 3 out of 24 Evo-generated IS605-like elements demonstrated evidence for both excision and insertion in vitro (Fig. 4, E to J, and fig. S21). This activity was also dependent on the presence of a putative catalytic tyrosine and on having a ssDNA substrate instead of double-stranded DNA (dsDNA), consistent with the known mechanism for IS200/IS605 TnpA (Fig. 4, F and I). To identify the precise boundaries of each element, we performed nanopore sequencing of the PCR products (Fig. 4, G and J, and figs. S22 and S23). As a control, we tested the natural IS200 element ISSpn6 and IS605 element ISHp608, and in both cases, we successfully detected the ISFinder-annotated boundaries (*71*), additionally revealing that the ISSpn6 TnpA can also mobilize using additional left and right ends within the locus (fig. S24). Three of our generated elements also appeared to mobilize using more than one left or right end pair (figs. S23, S25, and S26). The functional IS605-like elements, which contain putative TnpB coding sequences, also contain sequences with significant matches (cmsearch E value < 0.001) to a covariance model constructed from known ωRNAs (Fig. 4E and fig. S26). As a whole, the 14 active elements use a diverse set of hairpins (Fig. 4, E and H, and figs. S25 and S26) and encode functional TnpA proteins with sequence identity as low as 67% to the fine-tuning database.

These generative results are notable given that successful transposition requires TnpA proteins that functionally dimerize, TnpA dimer interactions with DNA hairpins in the LE and RE, base pairing between the LE and RE hairpins and the target site, and strand cleavage and exchange. Despite the complexity of this mechanism, we observed a high design success rate, nearing 50% for the IS200-like systems. Generative design and diversification of this functional class of MGEs could explore regimes of high activity unconstrained by natural evolutionary pressure on transposon fitness, expanding our understanding of transposase protein requirements and enabling biotechnological applications.

Learning gene essentiality with long genomic context

Beyond the molecular or systems level, we designed Evo to be capable of analyzing whole genomes. We conducted a second stage of pretraining in which Evo processed sequences with 131,072-token context (Fig. 5A) that also contained species-specific tokens. This stage used data from the genome taxonomy database (GTDB) and a subset of IMG/VR that excludes eukaryotic viruses (Fig. 1C, fig. S1, and Materials and methods). Evo maintains single-nucleotide resolution at its 131,072 context length, which is important because even a single-nucleotide mutation in an essential gene can be incompatible with life if it disrupts that gene's expression or function (*72*).

To this end, we evaluated whether Evo would be sensitive to mutations in essential genes solely based on small changes in a long genomic sequence. We conducted an experiment in which we inserted premature stop codons

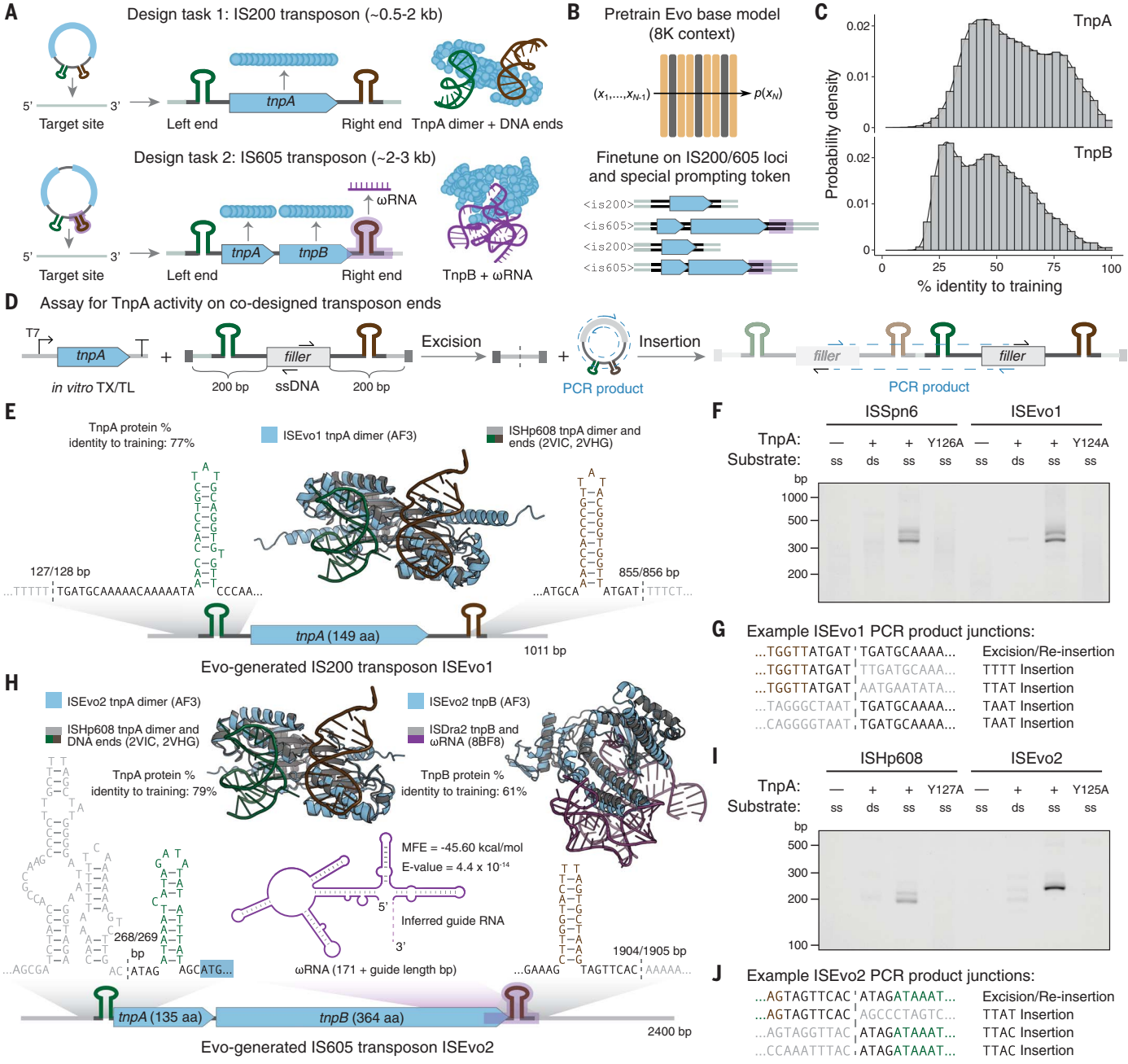


Fig. 4. Fine-tuning on IS200/IS605 sequences enables generative design of transposable biological systems. (**A**) IS200 and IS605 MGEs contain a TnpA transposase and are flanked by left and right end terminal hairpins that interact with the TnpA to accomplish transposition. IS605 MGEs additionally encode a TnpB-ωRNA complex that performs DNA cleavage. Our design task is to produce sequences that contain these DNA, ncRNA, and protein components. (**B**) We fine-tuned Evo, after its initial 8k pretraining phase, on natural sequences containing IS200/IS605 systems. (**C**) Histograms representing the distribution of the percentage identity of Evo-generated TnpA and TnpB proteins to their best match in the fine-tuning set of natural TnpA and TnpB proteins. (**D**) Schematic of the in vitro assay for evaluating designed TnpA activity on codesigned DNA ends. Excision will produce a band corresponding to the formation of the RE-LE junction in the resulting circular product, and (re-)insertion will produce a band from the joining of two ssDNA substrates, both detectable by a single PCR. (**E**) Schematic of the Evo-generated IS200-like system, ISEvo1, containing element annotations and its relevant DNA and protein features. (**F**) A 2% agarose gel with SYBR Gold showing that ISEvo1 TnpA functions in vitro on ssDNA substrates, requiring the catalytically active tyrosine (Y124) and with substantially reduced activity on dsDNA substrates. (**G**) Example reads from nanopore sequencing of PCR products from the ISEvo1 TnpA in vitro assay. (**H**) Schematic of the Evo-generated IS605-like system, ISEvo2, containing element annotations and its relevant DNA, RNA, and protein features. (**I**) A 2% agarose gel with SYBR Gold showing that ISEvo2 TnpA functions in vitro on ssDNA substrates, requiring the catalytically active tyrosine (Y125) and with substantially reduced activity on dsDNA substrates. (**J**) Example reads from nanopore sequencing of PCR products from the ISEvo2 TnpA in vitro assay.

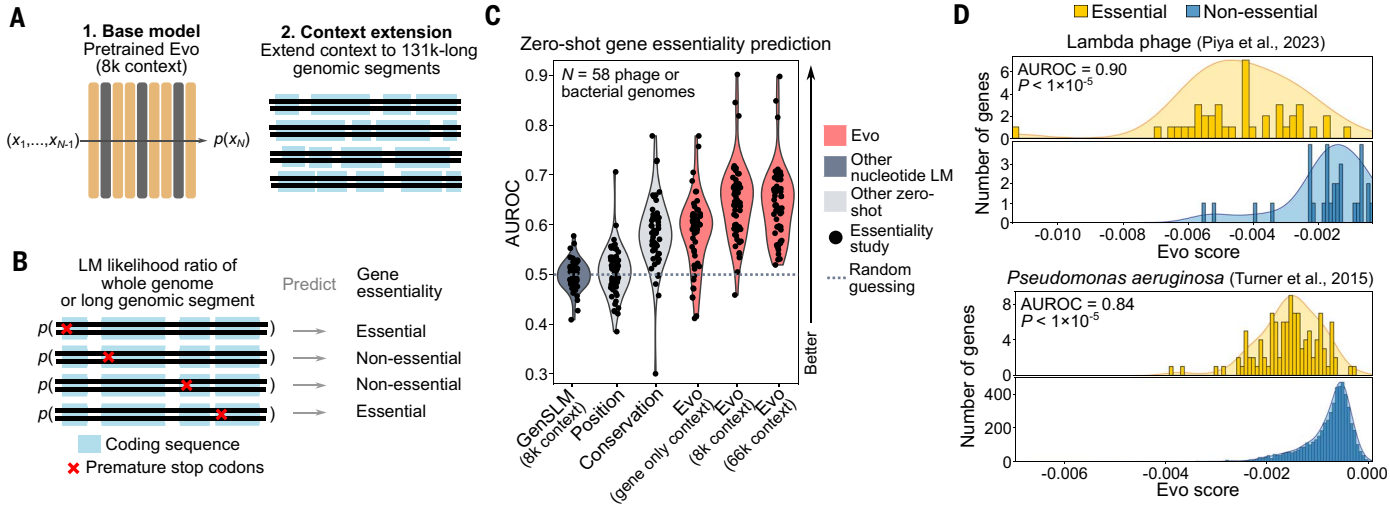


Fig. 5. Evo learns mutational effects on organismal fitness across diverse bacterial and phage genomes. (**A**) For genome-scale prediction and generation tasks, we first pretrained Evo on sequences with 8192 tokens and then extended its context window size in a second pretraining phase to sequences of 131,072 tokens. (**B**) We performed an in silico, genome-wide mutagenesis screen in which we introduced premature stop codons at each coding sequence in a genome. We computed the language model (LM) likelihood of the mutated gene sequence plus some amount of additional genomic context (up to 66 kb). We then took the ratio of this likelihood to the likelihood of the unmutated sequence. We tested whether these likelihood ratios would be predictive of gene essentiality. (**C**) Violin and strip plots of the distribution of the strength of gene essentiality prediction across 58 studies (each dot corresponds to a different study), in which each study conducted a genome-wide essentiality screen in a bacterial (N = 56) or phage (N = 2) species. We measured predictive performance as the AUROC in which the LM likelihood ratio is used to predict a binary label of "essential" or "nonessential." "Gene-only context" indicates that the model is provided with only the gene sequence and no additional flanking genomic context. "8k context" and "66k context" indicate that the LM is provided with the gene sequence and flanking genomic context up to a total of 8192 or 65,536 tokens, respectively. Evo has some predictive performance with gene-only context, has vastly improved performance from gene-only to 8k context, and some outlier improvements from 8k to 66k context. (**D**) Histograms representing the distributions of the log of the likelihood ratios ("Evo score") for the essential genes (blue) and the nonessential genes (yellow) in two genomes: lambda phage (top) and *P. aeruginosa* (bottom). These results are based on providing Evo with 66k context.

at the beginning of each coding sequence in a given organism's genome and measured the effects of these changes on Evo's likelihood with respect to the likelihood of the wild-type sequence (Fig. 5B). When computing the changes to the mutant versus wild-type sequences, we evaluated Evo on the gene sequence alone ("gene-only context") or the gene sequence with lanking context up to a total of 8192 tokens ("8k context") or 66,000 tokens ("66k context") (Materials and methods). We hypothesized that mutations to essential genes would result in larger, more negative changes in log-likelihood compared with mutations to nonessential genes.

On a dataset of 56 whole-genome essentiality studies in bacteria from the DEG database (*73*) and two whole-genome essentiality studies in phage from Piya *et al.* (*74*), we observed that the changes in Evo log-likelihood with 66k context are significantly associated (Bonferroni-corrected permutation-based $P < 0.05$) with gene essentiality in 49 of 58 genomes. We also observed that providing the model with additional genomic context beyond the gene sequence results in a substantial improvement in performance, especially from gene-only context to 8k context. From 8k to 66k context, the average predictive performance is comparable, although performance on the lower range of examples does improve with longer context (Fig. 5C and fig. S27, A and B). For a few genomes, the zero-shot performance with 66k context is notably strong, with an AUROC of 0.90 on lambda phage essentiality data (*74*) and an AUROC of 0.84 on *Pseudomonas aeruginosa* essentiality data (*75*) (Fig. 5D).

Evo likelihood changes are also indicative of gene essentiality when using different in silico mutagenesis strategies, such as varying the number of stop codons inserted or deleting the gene sequence entirely (fig. S27C and Materials and methods), though we did not attempt an exhaustive search of the best prompting strategy for this task. GenSLM, a codon language model that had mild predictive performance of mutational effects on single-gene protein function (Fig. 2B), did not demonstrate sensitivity to gene essentiality (Fig. 5C).

As control analyses, we examined genome position and sequence conservation. A gene's position in the genome showed no link to essentiality (Fig. 5C). We observed that more conserved sequences tended to be essential, with an association strength similar to that of Evo with gene-only context but weaker than that of Evo with genomic sequence context (Fig. 5C).

These results highlight the added value of Evo's ability to consider genomic context when predicting gene essentiality. Together, these results demonstrate that Evo can learn how small mutations affect fitness at a whole-organism level across many bacterial and phage species, without any explicit genome annotations, task-specific training data, or functional labels. In contrast to protein or codon language models, Evo can learn how individual genes interact with a broader genomic context.

Generating DNA sequences at genome scale

Given Evo's generative capabilities, we were interested in testing its generation quality at long sequence lengths without additional fine-tuning. We used Evo to sample 16 sequences each containing ~1 Mb, representing more than seven times the model's context length of 131 kb. For comparison, the smallest "minimal" bacterial genomes are ~580 kb in length (*76*). We prompted the model to generate bacterial genomes using the species-level tokens in the training dataset (Fig. 6A). To evaluate how closely our generated sequences resemble natural genomes, we used CheckM (*77*), a tool originally designed to assess the quality of bacterial DNA sequenced from nature. CheckM computes various metrics, including coding sequence density and the presence of highly conserved prokaryotic marker genes. We used these statistics to compare the key characteristics

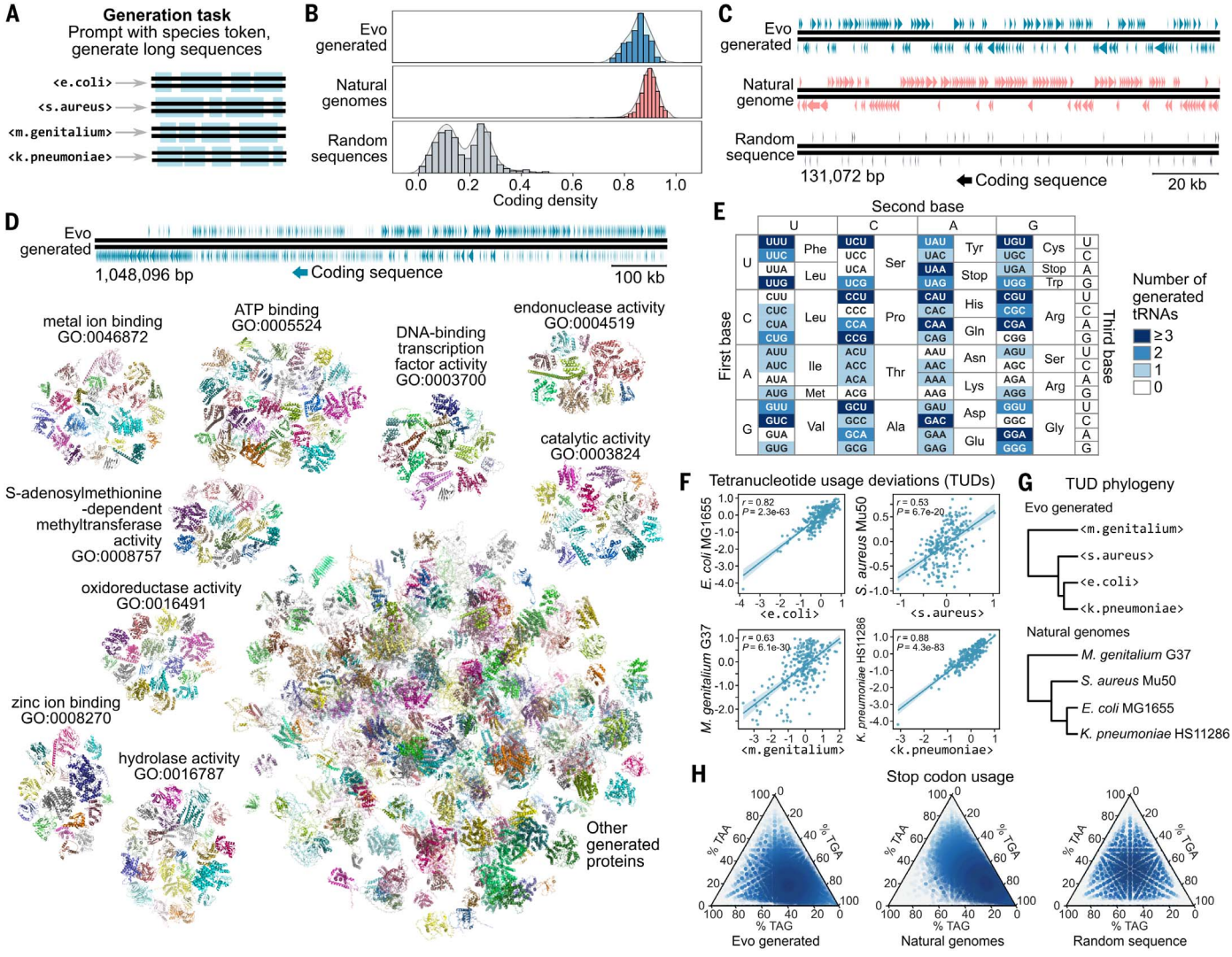


Fig. 6. Evo generates megabase-scale sequences with plausible genomic architecture. (**A**) We prompted Evo with species-level tokens used during the second pretraining stage. We use bacterial species prompts and generate sequences of ~650 kb in length. (**B**) Histograms depicting the distribution of coding density scores among 131-kb crops of sequences generated by Evo ("Evo generated"), sequences from natural bacteria ("natural genomes"), or sequences in which the four base pairs were sampled uniformly at random ("random sequences"). (**C**) Arrow plots depicting the organization of coding sequences on an example 131-kb sequence generated by Evo, derived from a natural genome, or sampled randomly. Coding sequences are depicted as arrows in which the horizontal length of the arrow corresponds to the genomic interval and the direction of the arrow indicates the strand. The top and bottom rows of arrows indicate the 5′-to-3′ and 3′-to-5′ strands, respectively, and the Evo-generated sequence was designated as the 5′-to-3′ strand. Both Evo-generated and natural genomes exhibit operon-like structure in which clusters of colocated genes are on the same strand. (**D** and **E**) An ~1-Mb generated sequence is represented as an arrow plot, as in (C). Below this arrow plot are ESMFold structure predictions of all protein coding sequences from 100 through 1024 amino acids in length, as identified by Prodigal. Structure predictions are aligned to natural proteins, which are then mapped to associated GO molecular function terms (Materials and methods). The largest GO categories are displayed as clusters alongside a large cluster containing all other proteins. ATP, adenosine triphosphate. (**F**) Log_2 of TUDs of Evo-generated versus natural genomes for each species prompt. Statistics are the Pearson correlation coefficient test. Shaded regions indicate a 95% confidence interval. (**G**) Hierarchical clustering of Evo-generated and natural genomes based on Euclidean distances of the TUDs. (**H**) Percent usage of each stop codon in all three reading frames of Evo-generated, natural, and random ORFs.

of our generated sequences with those of natural genomes.

Notably, Evo generated sequences have nearly the same coding densities as natural genomes, and substantially higher than that of random sequences (Fig. 6B). When visualized, both natural and generated sequences display similar patterns of coding organization (Fig. 6C), with sequences in close proximity typically found with the same strand orientation; in bacteria, these closely linked groups of coding sequences typically correspond to functionally tied gene clusters or operons. When using ESMFold to obtain protein structure predictions corresponding to these coding sequences, almost all showed predicted secondary structure and globular folds (Fig. 6, D and E, and fig. S28). Many proteins also showed structural similarity to natural proteins involved in fundamental molecular functions as annotated by gene ontology (GO) terms (Fig. 6, D and E). Across all our generated sequences representing ~16 Mb, Evo was also able to generate 128 tRNA sequences containing anticodons that correspond to all canonical amino acids (Fig. 6E).

We further observed that various genome-wide sequence patterns including the GC content,

dinucleotide frequencies, and certain codon usage patterns more closely resembled those of natural genomes compared with random sequences (fig. S28, A to C). To assess the accuracy of species-specific prompting, we calculated tetranucleotide usage deviations (TUDs), a strong indicator metric of phylogenetic relatedness (*78*). We found strong correlations between species-specific generations and their corresponding natural reference sequences, with TUDs sufficiently accurate to reconstruct natural phylogenetic relationships among the generated sequences (Fig. 6, F and G). We also examined stop codon frequencies across reading frames, a conserved genomic feature in prokaryotes (*79*). TGA and TAA stop codons appeared most frequently, whereas TAG was least common, consistent with previously observed patterns in prokaryotic genomes (Fig. 6H) (*80*). By contrast, random sequences showed an unbiased proportion of stop codons. These analyses collectively demonstrate that Evo's generated sequences capture multiple layers of genomic signatures characteristic of natural prokaryotic genomes.

However, there are characteristics of these genomes that are unnatural. The generated sequences do not contain many highly conserved marker genes that typically indicate complete genomes and, across the ~16 Mb of sample sequence, Evo generated only three rRNAs (*81*). Many of the protein structure predictions are of low confidence, are biased toward evolutionarily simpler α-helical secondary structures (*82*), and have limited structural matches to any entry in a representative database of naturally occurring proteins (fig. S28E).

These results suggest that Evo can generate genome sequences containing plausible high-level genomic organization at an unprecedented scale without extensive prompt engineering or fine-tuning. These samples represent a "blurry image" of a genome that contains key characteristics but lacks the finer-grained details typical of natural genomes. This is consistent with findings involving generative models in other domains, such as natural language or image generation. For example, directly sampling from a large natural language model typically produces sequences that are grammatically correct yet locally biased toward simpler sentence constructions and that are globally incoherent, especially at long lengths. Promisingly, in these domains, algorithmic techniques have emerged to improve the quality of generations compared with sampling from the pretrained model alone (*83*–*85*). The baseline generation quality observed without any fine-tuning suggests that Evo is also amenable to these techniques.

Discussion

Evo is a genomic foundation model trained on hundreds of billions of DNA tokens across the evolutionary diversity of prokaryotic life, capable of prediction and generation tasks at the scale of individual molecules, molecular complexes, systems, and even whole genomes. Based on a state-of-the art hybrid model architecture, Evo enables single-nucleotide-resolution language modeling at a context length of 131,072. We conducted the first scaling laws analysis of DNA pretraining across several architectures, where we observed StripedHyena outperforming several baseline architectures, including Transformers. Evo accurately performed zero-shot prediction across diverse fitness or expression prediction tasks on proteins, ncRNAs, or regulatory DNA that matches or outperforms specialized models while also understanding how mutations to individual genes can affect broader organismal fitness. As a multimodal generative model, we use Evo to generate CRISPR-Cas proteins and their non-coding guide RNAs, multicomponent transposable systems, and megabase-long sequences that recapitulate the architecture of real genomes. We experimentally validated the functional activity of EvoCas9-1 and Evo-generated IS200 and IS605 systems. We make open-access code and models for Evo publicly available at https://github.com/evo-design/evo.

A model capable of genome-scale design has the potential to advance therapeutic discovery, sustainability, and our understanding of fundamental biology but simultaneously raises biosafety and ethical considerations. The Global Alliance for Genomics and Health (GA4GH) (*86*) has developed principles for the oversight of genetic engineering technologies and could provide a robust foundation for transparency, accountability, and shared responsibility. Such a framework is essential to foster international cooperation that benefits all humanity. A proactive discussion involving the scientific community, security experts, and policy-makers is imperative to prevent misuse and to promote effective strategies for mitigating existing and emerging threats. We open-source the model to promote transparency and begin a dialogue with the broader scientific community, and we apply the precaution of excluding eukaryotic viruses from our pretraining dataset. We further include an extended supplementary discussion on safety and ethical considerations (see supplementary materials). Clear, comprehensive guidelines that delineate ethical practices for the field are required for the responsible development and use of genome-scale language models.

Despite the notable capabilities of this first-generation DNA foundation model, a number of technical limitations and challenges remain. We pretrained Evo on a dataset of 300 billion prokaryotic tokens, which represents a miniscule portion of petabytes of publicly available genomic data. Because our model is trained only on prokaryotic data, our ability to predict functional effects of mutations on human protein fitness is limited. Natural language models often struggle to maintain coherent and diverse generation over long sequences, and Evo can demonstrate similar properties. For example, we observed that many CRISPR-Cas generations had clearly problematic sequences, such as missing or truncated *cas* genes. At the genome-scale, Evo generates megabase-long sequences that demonstrate a high-level understanding of genome organization, but it struggles to include key marker genes, such as full sets of rRNAs. Improvement on long-range prediction or generation tasks will require both methodological improvements and biologically motivated problem selection and evaluation. These limitations mirror the constraints of natural language models, which have been improved over time with increased scale, labeled data, prompt engineering, and alignment with human preferences (*39*, *83*–*85*, *87*). We expect a similar trajectory for models of DNA.

We expect that Evo will benefit from additional scale, longer context length, and more diverse pretraining data. Given the success of language model–guided directed evolution of proteins (*88*, *89*), genomic language models may also help guide the directed evolution of multigene systems. The coevolutionary information contained in these models could improve molecular structure prediction in a multigene context (*5*, *47*). With better conditioning or prompt engineering, Evo could form the basis of a next-generation sequence search algorithm by enabling metagenomic mining at a relational or a semantic level rather than extracting literal sequences from existing organisms. The incorporation of eukaryotic genomes into Evo will need to consider the far higher complexity of these genomes and require substantial resource investment in engineering, compute, and safety-related model alignment. Combined with advances in large-scale genome modification (*90*), Evo expands the scope of biological engineering and design to the scale of whole genomes.

Materials and methods

StripedHyena architecture

Evo is based on StripedHyena (*34*), a state-of-the-art hybrid model architecture for sequence modeling. Evo comprises 32 blocks at a model width of 4096 dimensions. Each block contains a sequence mixing layer, tasked with processing information along the sequence dimension, and a channel mixing layer, focused on processing information along the model width dimension. In the sequence mixing layers, Evo uses 29 hyena layers, interleaved with 3 rotary (*29*) self-attention layers at equal intervals. We parametrize convolutions in hyena operators using the modal canonical form described in reference (*28*). For the channel mixing layers, Evo uses gated linear units (*91*, *92*). Evo further normalizes the inputs to each

layer using root mean square layer normalization (*93*).

Hyena layers

Hyena (*34*) is a sequence mixer implementing an input-dependent (data-controlled) operator via a composition of short convolutions, long convolutions and data-controlled gating (Fig. 1B). Hyena belongs to the class of deep signal processing primitives (*28*, *34*, *94*), designed for efficient, input-dependent computation in large-scale sequence models. Input dependence allows an architecture built with deep signal processing layers to adapt such computation based on the input, unlocking in-context learning (*95*, *96*). Hyena relies on structured operators compatible with fast multiplication algorithms, which can be evaluated in subquadratic time, e.g., via Fast Fourier Transforms or parallel scans. The operators are parametrized implicitly, i.e., by learning a map from positional embeddings, or the input, to the parameters of the operator itself. Typical choices of implicit parametrizations are linear projections, hypernetworks (*34*, *97*) or linear state-space models in modal or companion form (*27*, *28*, *98*–*100*).

Self-attention layers

Self-attention is the core sequence mixing operator of Transformer models. Self-attention constructs the output sequence as a weighted combination of the input elements, where the weights themselves are input-dependent. Given an input sequence, the forward pass of a self-attention layer is

$$(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \mapsto A(\mathbf{Q}, \mathbf{K})\mathbf{V}$$
$$A(\mathbf{Q}, \mathbf{K}) = \mathrm{softmax}\left(\mathbf{Q}\mathbf{K}^{\mathrm{T}}\right)$$

where queries $\mathbf{Q} \in \mathbb{R}^{L\times D}$, keys $\mathbf{K} \in \mathbb{R}^{L\times D}$, and values $\mathbf{V} \in \mathbb{R}^{L\times D}$ are obtained through a linear transformation of an input matrix $\mathbf{U} \in \mathbb{R}^{L\times D}$, e.g., $\mathbf{V} = \mathbf{U}\mathbf{W}_v$, and L denotes the sequence length and D denotes the hidden dimension. The softmax is applied to rows of A. The query, key, value terminology is borrowed from databases, where keys are used to index stored values. Conceptually, the values of the attention matrix $A(\mathbf{Q}, \mathbf{K})$ measure the similarity between queries and keys akin to matching queries to keys in a database.

Positional embeddings

By itself, the self-attention operator does not have any notion of the different positions of the input embeddings in an input sequence. For this reason, it is generally supplemented with a positional encoding mechanism. The attention layers of StripedHyena use a rotary position embedding mechanism (RoPE) to model relative positional information (*29*). Position information is encoded by rotating the query and key token vectors of the attention operator. Specifically, RoPE implements a rotation to queries and keys, with the rotation magnitude defined as a function of their relative position in the sequence.

To extend the context window length from 8k to 131k during our second pretraining stage, we apply linear position interpolation to extend the rotary position embedding applied in the first pretraining stage at 8k sequence length [for details, see (*19*)]. Interpolating enables the model to continue leveraging its learned representations when applied to longer sequences than it was originally trained on. We also tested other position interpolation methods but found that they performed slightly worse than linear interpolation on our data.

Tokenization

In language modeling, tokens describe the smallest unit of semantic information that is used by a model to process language. For example, tokens can indicate individual words of a vocabulary or even lower-level semantic information such as individual characters. Tokenization describes the process of mapping these semantic language units, such as words or characters, to specific integer values, each indicating an entry in a lookup table. These integer values are mapped by embedding layers to vectors, which are then processed by the model in an end-to-end fashion. Evo tokenizes DNA sequences at single-nucleotide resolution, using the UTF-8 encoding implemented in Python. During pretraining, Evo uses an effective vocabulary of four tokens, one per base, from a total vocabulary of 512 characters, which allows for vocabulary expansion during subsequent downstream tasks. We use the additional characters to enable prompting with special tokens during generation with fine-tuned models.

OpenGenome datasets

The OpenGenome pretraining dataset (table S2) was compiled from three different sources: (i) bacterial and archaeal genomes from the Genome Taxonomy Database (GTDB) v214.1 (*77*), (ii) curated prokaryotic viruses from the IMG/VR v4 database (*36*), and (iii) plasmid sequences from the IMG/PR database (*37*). For GTDB, representative genomes for each species were retained to reduce data redundancy.

For IMG/PR, only one representative per plasmid taxonomic unit (PTU) was kept. For IMG/VR, sequences were retained only if they were labeled as "high-confidence" according to the database metadata, and only one representative per viral operational taxonomic unit (vOTU) was kept. These sequences were further curated to remove potential eukaryotic viruses by keeping only sequences whose assigned taxonomic classification was found within a prokaryotic host at least twice. Next, the remaining taxonomic classifications were inspected and further filtered to exclude all viruses assigned to any of 19 families (Adenoviridae, Caliciviridae, Coronaviridae, Filoviridae, Flaviviridae, Hantaviridae, Hepadnaviridae, Herpesviridae, Orthomyxoviridae, Papillomaviridae, Paramyxoviridae, Picornaviridae, Poxviridae, Reoviridae, Retroviridae, Rhabdoviridae, Circoviridae, Geminiviridae, Picobirnaviridae) or 12 orders (Amarillovirales, Durnavirales, Geplafuvirales, Herpesvirales, Lefavirales, Ortervirales, Orthopolintovirales, Piccovirales, Picornavirales, Priklausovirales, Cirlivirales, and Mulpavirales). Next, viruses with poor taxonomic specificity were excluded, including those with no assigned realm at all, and those only assigned up to the level of r:Riboviria, r:Monodnaviria, k:Heunggongvirae, k:Bamfordvirae, p:Preplasmiviricota, p:Cressdnaviricota, p:Pisuviricota, or c:Tectiliviricetes.

The CRISPR-Cas and IS200/IS605 fine-tuning datasets were compiled from a previously described custom database gathered from multiple sources (*101*). Briefly, this custom database includes genomic and metagenomic sequence data from NCBI RefSeq (*102*), UHGG (*103*), JGI IMG (*104*), the Gut Phage Database (*105*), the Human Gastrointestinal Bacteria Genome Collection (*106*), MGnify (*107*), Youngblut *et al.* animal gut metagenomes (*108*), MGRAST (*109*), and Tara Oceans samples (*110*).

To compile the CRISPR-Cas genomic loci, this custom database was searched using profile HMM models and the HMMER software package to identify Cas9, Cas12, and Cas13 sequences (*111*). Several pHMMs were collected from the CRISPRCasTyper annotation tool (*112*), and a recent computational survey of TnpB and Cas12 (*113*). Custom Cas13 pHMMs that were previously generated by our group were also used (*101*). These models were searched against our large custom database using hmmsearch and the parameter "-Z 1000000." All hits that met $E < 1 \times 10^{-6}$ with at least one pHMM were kept. Only hits that were at least 300 amino acids long and covered over 80% of the pHMM were kept. For all hits to a given pHMM, only proteins that were within the middle 99% of the size distribution were kept. Corresponding genetic loci were extracted from the database, including 8192 nucleotides of flanking sequence on both the 5′ and 3′ ends of the Cas effector CDS. The tool minced was used to identify CRISPR arrays in the flanking sequences using the parameters "-minRL 18 -maxRL 50 -minSL 18 -maxRL 50." Only loci with both a predicted Cas effector and a CRISPR array were retained. The final CRISPR-Cas loci were extracted by first identifying the subsequence that covered both the Cas effector and the CRISPR array, and then including additional flanking nucleotides on both sides up until 8192 were retained for fine-tuning purposes. Only 1 locus per 90% identity Cas cluster was retained, clustered using the MMseqs2 command "easy-cluster --cluster-reassign --cluster-mode 0 --cov-mode 0 -c 0.7 --min-seq-id 0.9" (*114*).

To compile the IS200/IS605 loci, this custom database was searched using a Pfam Y1 HUH Transposase pHMM model (Pfam ID: PF01797). This pHMM identifies IS200/IS605 TnpA proteins. All matches meeting E value < 1×10^{-6} that covered at least 80% of the pHMM and were less than 400 amino acids were kept. 8196 nucleotides of CDS-flanking sequence was then extracted for each hit. Loci that also contained TnpB coding sequences were identified using previously compiled pHMMs (*113*), and a custom pHMM compiled using jackhmmer and the ISDra2 TnpB as an initial query against the MGnify protein database, followed by a MAFFT alignment of hits and pHMM construction with HMMER (*107*, *111*, *115*). Hits that were between 250 and 650 amino acids in length were retained, and only loci where the distance between the beginning and end of the TnpA and TnpB sequences was less than 2500 nucleotides were retained. For TnpA-only loci, up to 300 nucleotides of flanking sequence were added to either side of the CDS. For TnpA+TnpB loci, up to 300 nucleotides were added to the TnpA side of the IS200/IS605 element, while 600 nucleotides were added to the TnpB side (to account for the presence of an ωRNA). Only 1 locus per 90% identity TnpA cluster was retained.

Training procedure

We pretrain Evo in two stages, first with a context size of 8k tokens, followed by a second stage where we increase the context size to 131k tokens. Multistage sequence length pretraining has been shown to reduce the overall number of compute hours required to train long context models (*116*). The pretraining was distributed across GPUs using pipeline parallel with 2 stages (pipeline parallel value of 2), where each stage processes a part of the training pipeline (depthwise). This reduces the memory footprint while allowing us to maximize throughput during training. In total, we trained Evo in stage 1 on 64 NVIDIA H100 GPUs for 2 weeks and on 128 NVIDIA A100 GPUs in stage 2 for an additional 2 weeks. In total, Evo was trained on ~340B tokens, using ~2×10^{22} FLOPs. Because OpenGenome contains 300B tokens, this equates to 1.13 epochs, where dataloading beyond 300B tokens would consist of repeated tokens that are uniformly randomly sampled in a different order than in the first epoch. For specific generation tasks, we further fine-tuned Evo, as described in the following sections. We also report long context perplexity scaling of Evo 131k in fig. S2. Additional details on training settings are provided in table S1.

Dataloading

We use sequence packing to generate training samples. A sequence of the specified context length is sampled at random from the entire training dataset, where the sampling is done without replacement over an entire training epoch. Because some DNA sequences are shorter than the context length, multiple DNA sequences can be appended until the context length (8k or 131k) is reached; likewise, because some DNA sequences are longer than the context length, a training sample could consist of a genomic subsequence. Individual DNA sequences at the level of assembled contigs are separated by end-of-sequence (EOS) tokens. Depending on the dataset or task, we additionally prepend special token(s) to condition the model, for example, to steer its generations through prompting.

Hyperparameter tuning and direct model comparisons

Before training Evo, we carried out hyperparameter tuning on partially trained 7B Transformer++ models and compared to similarly sized Hyena and StripedHyena models. We swept batch size, learning rate and other architectural details. Even when controlling for training iterations instead of compute (FLOPs), Transformer++ performance is substantially worse than StripedHyena (fig. S4). Out of all the baselines, we find that StripedHyena achieves the overall lowest perplexity at the 7B scale, consistent with the scaling rates presented in Fig. 1G.

Scaling laws

We compare different classes of architectures via a compute-optimal protocol, aimed at evaluating results on the compute-optimal frontier. Compute-optimal analysis studies the best performance of a pretraining run given a compute budget, typically indicated in floating point operations (FLOPs), and achieved by optimally allocating portions of the compute budget to model size and dataset size. Architecture types differ in compute efficiency, as well as how they allocate this compute budget.

We started by tuning hyperparameters such as learning rate and batch size for Transformer++ with a grid search, then used the same values for all architectures except in settings where numerical instability was observed. To address instability, we lowered the learning rate gradually and repeated the experiment until convergence. In all experiments, we trained models with 8192 tokens in context length. For each compute budget defined by a total FLOP count, we varied the model sizes (6 million to 1 billion parameters) and the number of tokens trained. To measure model performance, we use the perplexity metric, which indicates how well an autoregressive model performs at predicting the next token of a sequence and is highly correlated with performance on downstream tasks. A lower perplexity value indicates better performance.

Scaling laws procedure

We provide a summary of the steps involved in our scaling laws analysis. Quantifying scaling rates allows us to predict performance as model size, dataset size, and compute grow.

1) Define a set of compute budgets to study. We use 8×10^{18}, 2×10^{19}, 4×10^{19}, and 8×10^{19} FLOPs.

2) Calculate the FLOPs (floating point operations) required to process a fixed input size for the model architecture of interest (i.e., the "cost" of using the model).

3) Identify the model's compute-optimal allocation for each compute budget: (a) Select a wide range of possible model sizes and calculate for each model size the corresponding number of tokens that need to be processed to reach the compute budget. Other hyperparameters are chosen according to table S3. We generally observe minor changes to model topology (depth, width) to only minimally affect perplexity, aligning our results with the findings presented by (*39*) for Transformers. (b) Train a model of each size and record its performance (e.g., in terms of perplexity). (c) Identify the optimal compute allocation: Following prior analysis, we fit a second-order polynomial as a function from (log) model size to perplexity, and extract obtained the compute-optimal point as its minimum. The compute-optimal point identifies the optimal allocation of model size and training tokens at the given compute budget.

After deriving the compute-optimal scaling rates (Fig. 1G), we compare architectures and compute optimal allocation of tokens and model size (fig. S5). In fig. S3, we also show rates for compute-suboptimal model sizes by architecture. We quantify the effect on perplexity scaling caused by a suboptimal allocation of compute budget to model or dataset size (e.g., training a smaller model for more tokens). We estimate the compute-optimal model size for each compute budget, then reduce it by a percentage (the offset). The corresponding perplexity is obtained via the IsoFLOP curves (Fig. 1F). Transformer++ perplexity scaling rapidly degrades outside the compute-optimal frontier, in contrast to Hyena and StripedHyena. Architecture details of models trained for our scaling law analysis provided in table S3.

Transformer++

We use a modern decoder-only Transformer architecture with rotary position embeddings (*29*), pre-norm with root mean square layer normalization, and SwiGLU as channel mixer. The inner width of the SwiGLU is 4/3 the model width. We experimented with grouped-query attention (GQA) (*117*) and found minimal differences in final loss, suggesting the technique may be suited to DNA sequence modeling, to further reduce memory footprint during inference. All scaling results with Transformer++ do not use GQA.

Hyena

The Hyena baseline is designed with the same architecture improvements applied to the Transformer++ model. We replace all multi-headed self-attention layers with hyena layers

and use a modal canonical parametrization for the long convolution, with state dimension 8.

Mamba

We use the implementation of Mamba as provided by the public repository (https://github.com/state-spaces/mamba).

Generating DNA sequences with Evo

We sample sequences from Evo using standard top-k and temperature-based methods for autoregressive models. Evo benefits from the fast recurrent mode of hyena layers, enabling lower latency and memory cost (*24*, *28*). In particular, we use the recurrent form of the modal canonical form as shown in (*28*), first processing the prompt with a Fast Fourier Transform modified to return output and state. We use a cache for the states of short convolutions. Evo can generate sequences of up to 650k nucleotides on a single 80GB GPU, in contrast to other long context methods for dense Transformers requiring a larger number of nodes. We use standard kv-caching for rotary attention layers in StripedHyena.

Controllable generation

We follow standard language model prompting techniques that condition generation on a given prefix. For class-conditional generation we prompt with a single token, representing the desired class, or genomic sequence type (e.g., CRISPR-Cas system, IS200/605). The model can also be steered by prompting on desired DNA subsequences.

Protein function prediction

We used DMS datasets to benchmark protein and nucleotide language models in their ability to predict mutational effects on protein function. In all cases, we used the nucleotide sequences reported by the original study authors. We limited our analysis to prokaryotic and human proteins, where notably the Evo training dataset only contains prokaryotic protein sequences.

To compile the nucleotide information from prokaryotic DMS studies, we used all the datasets listed as "prokaryote" in the ProteinGym benchmark for which we could also find nucleotide-level information reported by the original study authors. This resulted in nine studies: a β-lactamase DMS by Firnberg *et al.* (*118*), a β-lactamase DMS by Jacquier *et al.* (*119*), a CcdB DMS (*120*), a multiprotein thermostability dataset (*121*), an IF-1 DMS (*122*), an Rnc DMS (*123*), an HaeIII DMS (*124*), a VIM-2 DMS (*125*), and an APH(3′)II DMS (*126*).

To compile the nucleotide information from human DMS studies, we narrowed the scope of the set of datasets used in our human benchmark to the human datasets used in reference (*45*) to benchmark mutational effect predictors. We also limited our analysis to studies where we could also find nucleotide-level information reported by the original study authors. This resulted in six studies: a CBS DMS (*127*), a GDI1 DMS (*128*), a PDE3A DMS (*129*), a P53 DMS by Kotler *et al.* (*130*), a P53 DMS by Giacomelli *et al.* (*131*), and a BRCA1 DMS (*132*).

We compared Evo (pretrained with 8k context) to two genomic DNA language models: GenSLM 2.5B, which was trained with a codon vocabulary on sets of genes from prokaryotic organisms (*15*) and Nucleotide Transformer 2B5_multi_species, which was trained with a 6-mer nucleotide vocabulary on genome sequences from prokaryotic and eukaryotic species (*16*). We also compared Evo to several protein language models trained on nonredundant, generic corpora of protein sequences: CARP 640M (*46*), ESM-1v (*41*), ESM-2 650M, ESM-2 3B (*47*), ProGen2 large, and ProGen 2 xlarge (*48*). For studies that provide models with multiple parameter sizes, we selected the largest size on which we could perform inference with an 80 GB NVIDIA H100 GPU on sequences from all our benchmarked studies without exceeding GPU memory. We also included ESM-2 650M and ProGen2 large given that these models have sometimes shown better performance at function prediction than larger versions of these models (*44*).

To compare nucleotide and protein language models, we used all unique nucleotide sequences and their corresponding fitness values as reported by the original studies. Occasionally, we observed that the fitness values reported for nucleotide sequences differed from fitness values reported for protein sequences; in such cases, we used the fitness values reported for nucleotide sequences and evaluated the protein language models using the translated sequence. In cases where there are multiple nucleotide sequences for a single protein sequence due to different codon usage, the nucleotide language models were evaluated on each unique nucleotide sequence and the protein language models were evaluated on the coding sequence corresponding to each unique nucleotide sequence; this means that a protein language model could have been evaluated on the same protein sequence multiple times for a given study. Some studies report fitness values for mutations that involve stop codons; in such cases, we evaluated the nucleotide language model on the sequence containing the stop codon and excluded these examples from the protein language model benchmark.

We computed the Spearman correlation between the experimental fitness values and the sequence likelihood (for autoregressive language models) or the sequence pseudolikelihood (for masked language models). When using Evo sequence likelihoods to score sequences, we also prepend the EOS token (used in the pretraining data to delimit different sequences) to the full sequence, which we find empirically to boost zero-shot performance. We assessed statistical significance of the Spearman correlation coefficient under a null hypothesis that the correlation coefficient is drawn from a t-distribution with $N - 2$ degrees of freedom, where N is the number of samples over which we compute the correlation. We used this null distribution to compute a P value based on the observed correlation. We used the scipy Python library (https://scipy.org/) to compute these values.

ncRNA function prediction

We used DMS datasets to benchmark protein and nucleotide language models based on their ability to predict mutational effects on ncRNA function. Given that no well-established benchmark datasets exist for ncRNA function prediction, we curated the literature for examples of ncRNA mutational scanning experiments. We obtained the following datasets: a ribozyme DMS by Kobori *et al.* (*133*), a ribozyme DMS by Andreasson *et al.* (*134*), a tRNA DMS by Domingo *et al.* (*135*), a tRNA DMS by Guy *et al.* (*136*), a ribozyme DMS by Hayden *et al.* (*137*), a ribozyme DMS by Pitt *et al.* (*138*), and a rRNA mutagenesis study by Zhang *et al.* (*51*).

We compared Evo (pretrained with 8k context) to the nucleotide language models described above as well as RNA-FM, which was trained on a single-nucleotide vocabulary on short ncRNA sequences (*50*). Like the methods applied to protein coding sequences above, we compiled experimental fitness values for each ncRNA variant. We computed the Spearman correlation between the experimental fitness values and the sequence likelihood (for autoregressive language models) or the sequence pseudolikelihood (for masked language models). When scoring sequences with Evo sequence likelihood, we also prepend the EOS token to each sequence. Correlation coefficients and associated P values were computed as described above.

Gene expression prediction from regulatory DNA

From LaFleur *et al.* (*52*), we obtained a dataset of 5193 promoter sequences that we randomly split into 4673 promoters in the training dataset and 520 in the validation dataset following the train-validation split sizes used in the original study. We also obtained another 5391 promoter sequences from the same study, which we used as a second validation dataset. We also obtained 4350 promoter sequences from Hossain *et al.* (*54*), 10,898 promoter sequences from Urtecho *et al.* (*53*), and 1493 promoter sequences from Yu *et al.* (*55*), which we used as held-out test sets. The datasets were further processed to remove the background DNA sequence by identifying the subsequence with the maximum predicted transcription initiation rate using the method of LaFleur *et al.* (*52*). We also obtained a dataset of 12,243

promoter-RBS sequences from Kosuri *et al.* (*56*), which we used as an additional test set. All promoter sequences had associated activity labels related to gene expression and the data from Kosuri *et al.* (*56*) quantifies both mRNA and protein expression. The supervised tasks described below were all trained only on data generated by LaFleur *et al.* (*52*) and then evaluated based on their ability to make predictions on data from other studies.

For the promoter activity prediction tasks, we computed the predictive performance of promoter GC content and the zero-shot sequence likelihoods from Evo and from GenSLM on the four test datasets. When scoring sequences with Evo sequence likelihood, we prepended the EOS token to each sequence. We evaluated the performance of Promoter Calculator (*52*) on the four test datasets, using the minimum predicted dG_total across the forward sequence as the prediction score.

We additionally trained supervised models on the training set of 4673 promoters and associated activity values, using the two validation datasets described above to guide model development. These supervised models used either one-hot-encoded sequence embeddings or neural embeddings from Evo. The neural embeddings leveraged the output of the last hidden hyena layer, which takes the form of a matrix with a dimension of the sequence length × the hidden dimension (4096). On these embeddings, we trained either a ridge regression model or a convolutional neural network (CNN). To implement ridge regression, we used the RidgeCV module from scikit-learn with default values, which identifies the α hyperparameter used to weight the ℓ_2-regularization term. As input features for ridge regression, we additionally averaged the Evo embedding over the sequence dimension to produce an embedding vector of length 4096 for each sequence

The CNN consists of two convolutional layers, each followed by a ReLU activation function. The first convolutional layer starts with an input embedding (where the sequence dimension was suffix-padded with zeros up to length 256) with 4096 channels, using a kernel size of 8 and a stride of 1, with "same" padding to preserve the input sequence length. The second convolutional layer takes the output from the first layer and applies similar operations. Following the convolutional layers, a max pooling layer with a kernel size of 7 and a stride of 1 is applied, with padding adjusted to maintain the sequence length. The pooled output is then flattened into a two-dimensional tensor, which is passed through a fully connected layer that reduces the data to 128 channels. A final fully connected layer further reduces the data to a single output. The forward pass through the network involves applying the ReLU activation after each convolutional and fully connected layer (except for the final output layer). The model was trained for 10 epochs with the Adam optimizer, a learning rate of 0.0001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

For the protein expression prediction task, we used the data linking RBS sequences to protein expression from Kosuri *et al.* (*56*). We evaluated the zero-shot predictive performance of the sequence likelihoods from Evo when only providing the model with the sequence of the promoter, the sequence of the RBS, or the sequence of the promoter-RBS pair. When scoring sequences with Evo sequence likelihood, we also prepend the EOS token to each sequence. We also evaluated the predictive performance of the GC content of the promoter-RBS concatenated sequence and the zero-shot likelihoods from GenSLM. We also evaluated the performance of RBS Calculator (*57*, *58*) by providing the online webtool (https://salislab.net/software/predict_rbs_calculator) with a simulated mRNA sequence created by concatenating the RBS sequence and the sequence of sfGFP used by Kosuri *et al.* (*56*). To ensure that the Spearman correlation is comparable across these settings, we computed the correlation over all 12,243 examples (which involves duplicating sequences in the promoter-alone or RBS-alone settings).

CRISPR-Cas fine-tuning and generation

To generate CRISPR-Cas systems, we fine-tuned Evo by continuing to train the 8k-context pretrained model on a dataset of CRISPR-Cas sequences, which was curated as described above. We retained most of the hyperparameters used during pretraining but set the batch size to 524,288 tokens and an initial learning rate of 0.00009698, which was the learning rate at the final step of pretraining. During fine-tuning, we prepended a single class token corresponding to the type of Cas protein (Cas9, Cas12, or Cas13), which was identified as described in the OpenGenome datasets section; this class token was then followed by the nucleotide sequence. We also modified the dataloader such that each sample provided to the model during training would begin with the first token of the CRISPR-Cas sequence and, if a sequence was shorter than the context length, we padded the sequence to the remaining context (where padding did not contribute to the loss computation). This ensured that each training sample would correspond to a single CRISPR-Cas sequence. We fine-tuned the model for ~10 epochs.

We prompted the model with a given class token and one additional character for each sequence generation. For example, to prompt for Cas9 sequences, we used either "``" or "`A" as the Cas9 prompt, since we found that, in some instances, adding an additional random nucleotide character would improve the quality of generations. We performed standard temperature-based and top-k autoregressive sampling (*139*). In our generations, we performed an exhaustive sweep consisting of temperatures of 0.1, 0.3, 0.5 and top-k values of 2 and 4. All sampled sequences were then combined and used for downstream extraction and analysis of candidate CRISPR systems.

CRISPR-Cas sampling evaluation

The in silico Cas evaluation pipeline consisted of an initial open reading frame (ORF) search using Prodigal (*140*) and subsequent profiling of the extracted ORFs using hidden markov model (HMM) profiles for each Cas subtype. Sampled sequences with a positive pHMM hit with an E value under 1×10^{-3} and a sequence length above a given threshold were further analyzed using the MinCED package to identify possible CRISPR arrays (*141*). Generations with Cas ORFs and CRISPR arrays were aligned against Cas ORF sequences in the training data with MMSeqs2 to identify the closest sequences in the training data in sequence identity (*114*). We then performed MAFFT alignments with nearest hits to recompute alignments. MAFFT alignments were trimmed to 80% of the full alignment length centered at the middle of the alignment and end-gaps were removed before determining an estimate for percent identity to the closest item in the training data (*115*). To assess generation quality, we computed a "degeneracy score" as the percent coverage of a sequence by any repetitive substring longer than a cutoff value. For example, the degeneracy score of "ATAGAAAA-AATAGGGGGAGA" with a cutoff of 4 would be 0.55.

To select candidates for experimental validation, Cas9 generations with an ORF sequence identity higher than 90% to a training sequence were first filtered out. Remaining generations were then scored based on the distribution of mismatches in the pairwise alignments between the candidate sequence and its closest hit in the training dataset. Sequences with alignments containing an even distribution of mismatches across the ORF sequence were scored highly and those with an uneven distribution (e.g., concentration of mismatches or gaps at the N and C termini) were down-weighted. The Cas9 ORFs from the top-ranking 2000 generations were folded with AlphaFold2 (*5*). From the predicted structures, generations were filtered based on pLDDT, radius of gyration, the presence of a detected tracrRNA sequence, and the presence of RuvC and HNH domains in the Cas9 ORF. The Biotite package was used to calculate radius of gyration (*142*). CRISPRtracrRNA was used to extract potential tracrRNA sequences from candidate generations and co-folded with the extracted crRNA sequence using RNAmultifold (*143*, *144*). The final 11 Evo-generated Cas9 candidates were selected from this subset through manual inspection of predicted Cas9 structure and predicted sgRNA secondary structure.

CRISPR-Cas in vitro cleavage

For an initial screen of 11 selected Cas9 candidates, we expressed the protein and sgRNA in vitro using the PURExpress (IVTT) kit (NEB E6800S) and the HiScribe T7 High Yield RNA Synthesis (IVT) kit (NEB E2050S), respectively, following the manufacturer's recommendations. The sgRNA IVT product was column-purified using the 500 μg Monarch RNA Cleanup kit (NEB T2050L) before use; the in vitro expressed protein was not purified before use. The IVT and IVTT products were performed in 20 μL reactions with 2 μL of expressed protein, 2 μL of gRNA, 2 μL of DNA target at a final concentration of 1 nM, and 2 μL of NEBuffer r3.1 (NEB B6003S) at a final concentration of 1X. Cleavage reactions were incubated at 37°C for 20 hours and quenched with a final concentration of 50 mM EDTA (Invitrogen no. 15575020) followed by 2 μL of RNase A treatment (NEB T3018L) for 30 min at 37°C and 2 μL of Proteinase K treatment (NEB P8107S) for 15 min at 65°C. Cleavage products were then column-purified using a QIAquick PCR Purification kit (Qiagen no. 28104) and stored at 4°C before performing gel electrophoresis on Novex 4 to 12% TBE gels (Invitrogen EC62352BOX) at a constant voltage of 200 V. Gels were stained with SYBR Gold Nucleic Acid Gel Stain for 5 min at a 1X concentration (Invitrogen S11494).

SpCas9 and EvoCas9-1 was recombinantly expressed in the *E. coli* strain OverExpress C43(DE3) (Sigma Aldrich CMC0019) and purified via His-tag and size-exclusion chromatography using the procedure described in the section "CRISPR-Cas recombinant expression and purification." 2 μL of commercially available SpCas9 (NEB M0386T), purified SpCas9, or purified EvoCas9-1 were incubated with 2 μL of either a targeting or nontargeting gRNA and 2 μL of a DNA target at a 10:10:1 molar ratio of Cas9:sgRNA:target. A final concentration of 1 nM was used for the target and final concentrations of 10 nM for both the Cas9 protein and sgRNA. Cleavage reactions were performed in 20 μL volumes with 2 μL of NEBuffer r3.1 (NEB B6003S) used at a final concentration of 1X. Reactions were incubated at 37°C for up to 12 hours with timepoints collected at 5 min, 15 min, 1 hour, 3 hours, and 12 hours. Separate and independent reactions were used for each timepoint and condition and quenched with a final concentration of 50 mM EDTA (Invitrogen no. 15575020) before treating with 2 μL of RNase A (NEB T3018L) at 37°C for 10 min and 2 μL of Proteinase K (NEB P8107S) at 65°C for 15 min. Cleavage products were column-purified using a QIAquick PCR Purification kit (Qiagen no. 28104) before performing gel electrophoresis on a Novex 4 to 12% TBE gel (Invitrogen EC62352BOX) at a constant voltage of 200 V. Gels were stained with SYBR Gold Nucleic Acid Gel Stain for 5 min at a 1X concentration (Invitrogen S11494).

CRISPR-Cas recombinant expression and purification

The sequence encoding the protein of interest was subcloned into a protein expression vector containing an N-terminal 8xHis tag followed by a TEV protease cleavage site using Gibson assembly. The protein was expressed in *E. coli* strain OverExpressC43(DE3) (MilliporeSigma) grown in Terrific Broth at 18°C for 16 hours after induction with 0.4 mM IPTG. The protein was purified by sequential affinity and size exclusion chromatography steps. Cells were centrifuged at 4000 ×g, 4°C for 15 min and resuspended in lysis buffer (50mM Tris–HCl pH 7.5, 0.5 M NaCl, 2 mM MgCl2, 10 mM imidazole, 10% glycerol) supplemented with EDTA-free protease inhibitor tablets (Roche) and 1 mg/mL lysozyme (ThermoFisher). Cell suspensions were then disrupted using a sonicator (Fisher Scientific). Crude lysate was subsequently ultracentrifuged at 40,000 ×g, 4°C for 45 min using a 70Ti rotor in a XE-90 ultracentrifuge (Beckman Coulter). Clarified lysate was then filtered through a 0.22 μm filter and loaded onto a 5 mL HisTrapFF column (Cytiva) using a peristaltic pump.

After the entire volume of the clarified lysate was flowed through the HisTrapFF affinity column, the column was washed extensively with Wash Buffer (50 mM Tris–HCl pH 7.5, 0.5 M NaCl, 30 mM imidazole, and 10% glycerol). The HisTrapFF column was then connected to an AktaPure system (Cytiva) and eluted using a linear gradient of Elution Buffer (50mM Tris–HCl pH 7.5, 0.5 M NaCl, 0.5 M imidazole, and 10% glycerol) in 1.5 mL fractions. Fractions corresponding to the peak identified to contain the protein of interest were pooled and concentrated using an Amicon 30 kDa MWCO filter (MilliporeSigma) before overnight cleavage of the 8xHis tag using TEV protease. Following TEV protease cleavage, the solution was applied to a second HisTrapFF column to remove the cleaved tag from the preparation. The column was washed with 15 mL Wash Buffer and the flow through was collected for concentration using an Amicon 30 kDa MWCO filter (MilliporeSigma). The concentrated protein was then applied to a Superdex200 10/300 column for purification by size exclusion chromatography, with an isocratic elution program using SEC Buffer (20 mM Tris–HCl pH 7.5, 0.5 M NaCl, and 1 mM DTT, 10% glycerol). Eluted protein was concentrated again using an Amicon 30 kDa MWCO filter (MilliporeSigma), flash frozen in liquid nitrogen and stored at −80°C.

IS200/IS605 fine-tuning and generation

To generate IS200 and IS605 systems, we fine-tuned Evo by continuing to train the 8k-context pretrained model on a dataset of IS200/IS605 sequences, which was curated as described above. We retained most of the hyperparameters used during pretraining but set the batch size to 524,288 tokens and an initial learning rate of 0.00009698, which was the learning rate at the final step of pretraining. During pretraining, we prepended a start token to each sequence labeling whether the system corresponded to an IS200 or an IS605 system. We used the token corresponding to the character "~" as the IS200 prompt and the token corresponding to the character "#" as the IS605 prompt. We also modified the data loader such that each sample provided to the model during training would begin with the first token of the IS200/IS605 sequence and, if a sequence was shorter than the context length, we padded the sequence to the remaining context (where padding did not contribute to the loss computation), similar to the strategy described for CRISPR-Cas9 systems above. We fine-tuned the model for ~10 epochs.

We prompted the model with a special prompting token for each sequence generation. We performed standard temperature-based and top-k autoregressive sampling (*139*). In our generations, we performed an exhaustive sweep consisting of temperatures of 0.1, 0.3, 0.5, 0.7, 0.9, 1.0, and 1.3, and top-k values of 2 and 4. Sampled sequences were further processed by splitting on the first whitespace character, keeping the first non-whitespace sequence, and only keeping generated sequences that were composed entirely of valid nucleotides.

We analyzed generated sequences using Prodigal to identify coding sequences and proteins (*140*), followed by hmmsearch (-Z 1000000) using pHMMs to identify TnpA and TnpB sequences (*111*), and cmsearch (-Z 4) using covariance models developed in a previous publication (*66*) to identify candidate ωRNAs (*145*). Candidate TnpA sequences were kept if they had an E value $< 1 \times 10^{-3}$ to the pHMM and if they covered at least 50% of the pHMM. Candidate TnpB sequences were kept if they had an E value $< 1 \times 10^{-3}$ to at least one pHMM, if they covered at least 50% of the pHMM, and if they were between 300 and 600 amino acids in length.

Predicted TnpA and TnpB protein sequences were aligned back to proteins in the training set using MMseqs2 (*114*). The top three hits for each protein were extracted and separately aligned using the MAFFT default algorithm to estimate the amino acid identity across the full lengths of the two sequences (*115*). To account for different start codons and to generate a more conservative percentage identity estimate, these alignments were trimmed to the middle 80% of each sequence, end gaps were trimmed, and the amino acid percent identity was recalculated, which we called a "trimmed percent identity."

TnpA and TnpB protein sequences were binned by distance from the training set in 9 equal width bins from 10% to 100% trimmed percent identity. 200 proteins were randomly selected from each bin for TnpA proteins that appeared in the absence of a TnpB protein

(IS200-like), TnpA proteins that appeared with a TnpB protein (IS605-like), and TnpB proteins that appeared with a TnpA protein (IS605-like). ESMFold was used to fold all 5400 proteins, with TnpA protein sequences folded as dimers with a glycine pseudo-linker of length 100. The mean backbone atom pLDDT was calculated and reported as a measurement of ESMFold prediction confidence. Example TnpA and TnpB proteins were aligned to the 2VIC and 8BF8 Protein Data Bank (PDB) structures, respectively, using the US-align tool (*146*), right-end and left-end DNA sequences from PDB structures 2VIC and 2VHG were overlayed on the aligned structure, and structures were visualized in PyMOL (*147*). RNAfold from the ViennaRNA package was used to fold the predicted ωRNA with default parameters (*148*, *149*). Visualizations of ωRNAs were drawn using R2R (*150*). Visualizations of ISEvo1 TnpA and TnpB were also computed using AlphaFold3 by uploading sequences to the AlphaFold Server (*64*).

Evo was also used to calculate the entropy of the conditional probabilities at each position in each sequence with the pertinent special token prepended. For example, the entropy at position i was calculated using the likelihoods $p(x_i|x_1, \ldots, x_{i-1})$ over the entire vocabulary. We then visualized these entropies alongside the annotated sequence positions for several canonical IS200/IS605 systems and summarized the average entropy values within 250 bp of TnpA and TnpB coding sequences.

IS200/IS605 categorical Jacobian analysis

We computed the "categorical Jacobian" matrix on a sequence of nucleotides based on a procedure introduced by Zhang *et al.* (*70*) and clarified in the accompanying code at the GitHub repository (https://github.com/zzhangzzhang/pLMs-interpretability). To summarize this procedure, let $\mathbf{x} = (x_1, x_2, \ldots, x_L)$, $x_i \in \mathcal{X}$ denote a sequence of length L where in our study we define $\mathcal{X} = \{\text{"A"}, \text{"C"}, \text{"G"}, \text{"T"}\}$ to be a nucleotide vocabulary. Let $f: \mathcal{X}^L \to \mathbb{R}^{L \times |\mathcal{X}|}$ denote the function for computing the language-model logits (where a softmax function computed over the logits for a given position corresponds to the language-model likelihoods for that position) given an input sequence $\mathbf{x}$.

Now we define a sequence $\mathbf{x}[\hat{x}_i] = (x_1, \ldots, \hat{x}_i, \ldots, x_L)$ as the sequence $\mathbf{x}$ mutated to $\hat{x}_i \in \mathcal{X}$ at position $i \in [L]$, where $[L]$ is defined as the set $\{1, 2, \ldots, L\}$. We also define $g(\mathbf{x}, \hat{x}_i, i) = f(\mathbf{x}) - f(\mathbf{x}[\hat{x}_i])$ where $g: \mathcal{X}^L \times \mathcal{X} \times [L] \to \mathbb{R}^{L \times |\mathcal{X}|}$ is a function that computes the difference in logits between the original sequence $\mathbf{x}$ and the mutated sequence $\mathbf{x}[\hat{x}_i]$.

The "categorical Jacobian" tensor $\mathbf{J}$ is then defined as

$$\begin{bmatrix} g(\mathbf{x}, \text{"A"}, 1) & \cdots & g(\mathbf{x}, \text{"T"}, 1) \\ \vdots & \ddots & \vdots \\ g(\mathbf{x}, \text{"A"}, L) & \cdots & g(\mathbf{x}, \text{"T"}, L) \end{bmatrix}$$

which requires mutating $\mathbf{x}$ to all nucleotides at all positions. Note that $\mathbf{J} \in \mathbb{R}^{L \times |\mathcal{X}| \times L \times |\mathcal{X}|}$. This tensor $\mathbf{J}$ is then modified to produce a mean-centered tensor $\hat{\mathbf{J}}$ by computing each entry in this tensor as

$$\hat{\mathbf{J}}_{i,j,k,l} = \mathbf{J}_{i,j,k,l} - \frac{1}{L}\sum_{i'=1}^{L} \mathbf{J}_{i',j,k,l} - \frac{1}{|\mathcal{X}|}\sum_{j'=1}^{|\mathcal{X}|} \mathbf{J}_{i,j',k,l} - \frac{1}{L}\sum_{k'=1}^{L} \mathbf{J}_{i,j,k',l} - \frac{1}{|\mathcal{X}|}\sum_{l'=1}^{|\mathcal{X}|} \mathbf{J}_{i,j,k,l'}$$

and is then symmetrized by computing, for each entry

$$\tilde{\mathbf{J}}_{i,j,k,l} = \frac{1}{2}\left(\hat{\mathbf{J}}_{i,j,k,l} + \hat{\mathbf{J}}_{k,l,i,j}\right)$$

to produce a final symmetrized tensor $\tilde{\mathbf{J}}$.

We can turn $\tilde{\mathbf{J}}$ into a positional "couplings map" matrix $\mathbf{C}' \in \mathbb{R}^{L \times L}$ in which each entry can be intuitively thought of as representing a "Euclidean" magnitude of the change in the logits across all values of the vocabulary $|\mathcal{X}|$, where a larger magnitude change indicates a greater information "coupling" between the two corresponding positions; more concretely, to calculate each entry in $\mathbf{C}'$, we compute

$$\mathbf{C}'_{i,j} = \left(\sum_{n=1}^{|\mathcal{X}|}\sum_{m=1}^{|\mathcal{X}|} \tilde{\mathbf{J}}^2_{i,n,j,m}\right)^{\frac{1}{2}}$$

We now define the "average product correction" (APC) function $a: [L] \times [L] \to \mathbb{R}$ as computing, for each entry in a matrix $\mathbf{X} \in \mathbb{R}^{L \times L}$

$$a(i,j;\mathbf{X}) = \frac{\mathbf{X}_{ij} - \left(\sum_{i'=1}^{L} \mathbf{X}_{i',j}\right)\left(\sum_{j'=1}^{L} \mathbf{X}_{i,j'}\right)}{\sum_{i'=1}^{L}\sum_{j'=1}^{L} \mathbf{X}_{i',j'}} - 1\{i = j\}$$

where $1\{\cdot\} \in \{0, 1\}$ is the indicator function. We are now ready to define the final matrix, $\mathbf{C} \in \mathbb{R}^{L \times L}$, which is obtained by computing, for each entry in $\mathbf{C}$

$$\mathbf{C}_{i,j} = a(i,j;\mathbf{C}')$$

Throughout the text, when we refer to the "categorical Jacobian matrix" or simply the "categorical Jacobian," we are referring to the matrix $\mathbf{C}$.

We computed the categorical Jacobian matrix using Evo fine-tuned on IS200/IS605 sequences for natural IS605 elements ISHp608, ISDge10, and ISDra2 using the full IS sequence flanked with 500 bp of natural context on either side, where each pair of flanking sequences is extracted from the best BLAST (*151*) hit against the nr/nt databases for the IS sequence from ISFinder (*71*).

IS200/IS605 filtering of generations and construct design

To nominate generated IS200/IS605 sequences for synthesis and experimental validation, the sequences were further curated as follows. TnpA proteins from generated sequences were first searched with blastp (*151*) against four natural TnpA proteins that were used as positive controls, originating from IS200/IS605 elements ISSpn6, ISHp608, ISDge10, and ISStin10. Alignments were filtered to keep only those that were between 100 and 200 amino acids in length, and to keep only those that had a trimmed percent identity with the nearest training example that was <90%, and those that were at least 50% identical to the nearest positive control as estimated by the blastp alignment. Structures of TnpA proteins from the remaining 723 ISSpn6-like, 697 ISHp608-like, 123 ISDge10-like, and 1686 ISStin10-like generated sequences were predicted using ESMFold (*47*) as monomers and only proteins with mean pLDDTs ≥0.7 were retained. Generations were further reduced by selecting for generations where the TnpA protein contained at least one HUH and one YXXXQ amino acid motif, had a TnpA start codon within ≤500 bp from the start of the generation, and where the TnpA protein length was ≤180 amino acids.

For remaining IS200-like generations, we further required that at least 250 bp be on either side of the predicted TnpA CDS. The 200 bp sequences flanking the TnpA CDS were searched for perfect hairpins (no mismatches or gaps allowed in the stem, and loop length ≤5 bp), and sequences with max length perfect hairpin stems ≤6 bp in the 200 bp left of the TnpA CDS or ≤8 bp in the 200 bp right of the TnpA CDS were filtered out (fig. S20C).

For the 247 ISStin10-like and 102 ISSpn6-like generations passing these filters, we computed upstream base pair propensity vectors using ViennaRNA (*144*) for the 200 bp on either side of the TnpA CDS (fig. S20D) by taking the row sum of the base pair propensity matrix where all pairwise base pair propensities were calculated using ViennaRNA.get_pr(i,j) for $i \leq j$. The resulting upstream base pair propensity vectors for each generation were hierarchically clustered with the upstream base pair propensity vectors for ISSpn6 and ISStin10 on Euclidean distance with the UPGMA algorithm. A dendrogram threshold was chosen manually by visual examination, and selected clusters were extracted using scipy.cluster.hierarchy.fcluster (fig. S20E). This process was repeated with remaining IS200-like candidates with best matches to ISStin10 against the IStin10 upstream base pair propensity vectors (fig. S20F), as well as with best matches to ISSpn6 against the ISSpn6 upstream base pair propensity vectors (fig. S20G). For any remaining sequences, the TnpA dimer structure was predicted using AlphaFold-Multimer-v2.3.0 via ColabFold (*152*) using two models with three recycles each, and sequences with TnpA dimer structures that did not appear to dimerize via pAE scores were discarded.

Remaining candidates were formatted for IDT synthesis as 520 bp sequences containing 30 bp of filler sequence containing a primer binding site for amplification followed by the 200 bp to the left of the TnpA CDS followed by 60 bp of filler sequence containing primer binding sites for two primers facing out followed by the 200 bp to the right of the TnpA CDS followed by 30 bp of filler sequence containing a primer binding site for amplification (data S1). Resulting sequences were uploaded to the IDT web portal and 12 ISStin10-like and 12 ISSpn6-like candidates were selected from the sequences that had green and yellow IDT synthesizability scores. The TnpA corresponding to these sequences were codon optimized using the IDT codon optimization tool set to *E. coli* and flanked with the standard T7 promoter, RBS, and T7 terminator sequences for PURExpress (NEB) as listed in the manufacturer's manual. An additional TnpA mutant construct in which any YXXXQ motif in the sequence was mutated to AXXXQ was also designed for each candidate. The codon-optimized TnpA and TnpA mutant protein coding sequences for PURExpress and end-containing sequences were ordered as IDT eBlocks.

For remaining IS*605*-like generations, we further required that at least 250 bp be upstream of the predicted TnpA CDS and that at least 200 bp be downstream of the predicted TnpB CDS. We then filtered for sequences with TnpB protein start codon distances of at most 100 bp downstream of the TnpA protein stop codon.

For the 407 ISHp608-like and 67 ISDge10-like generations passing these filters, we formatted the ends for IDT synthesis as 520 bp sequences containing 30 bp of filler sequence containing a primer binding site for amplification followed by the 200 bp to the left of the TnpA CDS followed by 60 bp of filler sequence containing primer binding sites for two primers facing out followed by the −50:150 bp to the right of the TnpB CDS followed by 30 bp of filler sequence containing a primer binding site for amplification (data S1).Resulting sequences were uploaded to the IDT web portal and only the 37 ISDge10-like and the 20 ISHp608-like sequences that were green by IDT synthesizability scores were retained. For these sequences, the TnpA dimer structure was predicted using AlphaFold-Multimer-v2.3.0 via ColabFold (*152*) using two models with three recycles each, and sequences with TnpA dimer structures that did not appear to dimerize via pAE scores were discarded. From the remaining sequences, 12 ISStin10-like and 12 ISSpn6-like candidates were selected ensuring that the best sequence identity matches to the fine-tuning set were ≥50%. For final synthesis and experimental validation, a different 60 bp filler sequence was used for the ISHp608-like candidates compared to the ISStin10-like, ISSpn6-like, or ISDge10-like sequences to eliminate a primer-binding site containing a TTAC, which is the canonical ISHp608 target site. The TnpAs corresponding to these 24 candidate sequences were codon optimized using the IDT codon optimization tool set to *E. coli* and flanked with the standard T7 promoter, RBS, and T7 terminator sequences for PURExpress (NEB) as listed in the manufacturer's manual. An additional TnpA mutant construct in which any YXXXQ in the sequence was mutated to AXXXQ was also designed for each candidate. The codon-optimized TnpA and TnpA mutant PURExpress and end-containing sequences were ordered as IDT eBlocks.

Similar eBlocks encoding TnpA using the natural sequence, encoding a TnpA mutant with the catalytic tyrosine mutated to alanine, and a 520 bp sequence containing the ends were ordered for the natural IS*200* transposon ISSpn*6* and the natural IS*605* transposon ISHp608.

IS200/IS605 TnpA protein preparation

TnpA and TnpA-mutant eBlocks were PCR amplified using NEBNext 2xPCR mastermix (New England Biolabs) for 35 cycles using an annealing temperature of 65°C and an elongation time of 15 s in 50 μL reactions with primers PURExpress_T7_F and PURExpress_T7_F (sequences provided in data S1), column purified using a QIAQuick PCR purification kit (Qiagen), and diluted to 30 ng/μL. In vitro transcription-translation reactions were performed using PURExpress (New England Biolabs) in 27 μL reactions containing 10 μL solution A, 7.5 μL solution B, 1 μL of Murine RNAse Inhibitor (NEB), and 8.5 ul (255 ng) of template DNA. DHFR expression plasmid provided with the PURExpress kit was used as template DNA for reactions lacking TnpA protein. Reactions were incubated for 3 hours at 37°C and directly transferred to in vitro reactions.

IS200/IS605 substrate DNA preparation

Substrate eBlocks were PCR amplified using NEBNext 2xPCR master mix (NEB) for 35 cycles using an annealing temperature of 65°C and an elongation time of 15 s in 100 μL reactions with a forward primer containing 3 PTOs and a reverse primer containing a 5′ phosphate (ssDNA_substrate_PTO_F and ssDNA_substrate_5phos_R; sequences provided in data S1), column purified using QIAprep Spin Miniprep Columns (Qiagen), and eluted in 45 μL water. The Guide-it Long ssDNA Production System v2 (Takara Bio) was used to generate substrate ssDNA in 50 μL reactions with 30 μL purified PCR product following the manufacturer's conditions with an incubation time of 10 min at 37°C and 5 min at 80°C with Strandase A, and 5 min at 37°C and 5 min at 80°C with Strandase B. The resulting ssDNA substrates were then column purified using a NucleoSpin Gel and PCR Clean-Up kit (Takara Bio) by diluting the reaction to 100 μL total volume, adding 200 μL buffer NTC (Takara Bio), mixing thoroughly before adding to the column, and washing with 600 μL buffer NT3 before eluting in 30 μL elution buffer. Resulting ssDNA products were diluted to 20 ng/μL as quantified using a NanoDrop One in ssDNA mode (ThermoScientific).

Substrate PCR products for use in the in vitro assay as dsDNA were further treated with exonuclease I (*E. coli*, New England Biolabs) to remove residual PCR primers or other ssDNA in 20 μL reactions containing 600 ng PCR product, 2 μL 10x exonuclease I buffer, and 5 μL of exonuclease I. After column purification using a QIAQuick PCR purification kit (Qiagen), the resulting dsDNA substrate was diluted to 20 ng/μL.

IS200/IS605 in vitro TnpA excision/insertion assays

In vitro transposition reactions were performed by incubating 10 μL PURExpress product with 10 μL (200 ng) of ssDNA or dsDNA substrate for 2 hours at 37°C. Reactions were treated with 1 μL RNase A (20 mg/mL, New England Biolabs) for 5 min at 37°C and 10 μL Proteinase K (8 units, New England Biolabs) for 15 min at 37°C. Resulting ssDNA products were then column purified using a NucleoSpin Gel and PCR Clean-Up kit (Takara Bio) by diluting the reaction to 100 μL total volume, adding 200 μL buffer NTC (Takara Bio), mixing thoroughly before adding to the column, and washing with 600 μL buffer NT3 before eluting in 30 μL elution buffer. PCRs were then performed in 50 μL reactions for 35 cycles using an annealing temperature of 65°C and an elongation time of 20 s using 4 μL eluate, NEBNext 2x PCR master mix (New England Biolabs) and primers FillerOut_F and FillerOut_R for ISStin10-like, ISSpn6-like, and ISDge10-like candidates and using primers ISHp608-like_FillerOut_F and FillerOut_R for ISHp608-like candidates (sequences are provided in data S1). PCR products were column-purified using a QIAquick PCR Purification kit (Qiagen) and run on either a 2% E-Gel EX agarose gel pre-stained with SYBR Gold or on a 48-well 2% E-Gel agarose gel pre-stained with SYBR Safe (ThermoScientific).

IS200/IS605 nanopore sequencing analysis of PCR products

PCR products from TnpA reactions were submitted for nanopore sequencing via the Premium PCR sequencing service from Plasmidsaurus (2 samples per condition), which uses the ligation sequencing kit v14 (Oxford Nanopore Technologies) and R10.4.1 flow cells (Oxford Nanopore Technologies). Reads were then processed by filtering for the expected read structure (FillerOut_F/ISHp608-like_FillerOut_F followed by sequence followed by FillerOut_R

reverse complemented or FillerOut_R followed by sequence followed by FillerOut_F/ISHp608-like_FillerOut_F reverse complemented), by looking for expected primer sequences in the 30 bp on either end, allowing for up to four errors (sequences are provided in data S1). Reads passing this filtering were then mapped to the relevant substrate sequence by sliding a window across the sequence, splitting each window into a left and right half, and matching each half to the substrate sequence, requiring a perfect match for both sides. The window was twice the minimum length i required for all substrings of length i from the substrate sequence to be unique. Each match was then added to a jump map matrix for each condition at the position corresponding to the rightmost base of the left side match and the leftmost base of the right-side match (fig. S22A). Transposon boundaries and hairpins were annotated based on these jump maps and additional manual processing and inspection of reads and alignments via Geneious Prime 2024 (https://www.geneious.com).

Gene essentiality prediction

We obtained binary genome-wide essentiality results for 56 bacterial genomes from the DEG database (*73*) in which coding genes are labeled with "essential" or "nonessential" binary labels. We also obtained genome-wide essentiality results for two phage genomes, lambda and P1, from Piya *et al.* (*74*) and used the binary labels assigned by the study authors based on the results of their CRISPRi screen.

To perform the in silico gene essentiality screen, we obtained the whole bacterial genome using the RefSeq IDs provided by DEG. We used RefSeq: NC_001416 as the reference genome for lambda phage and RefSeq: NC_005856 as the reference genome for P1 phage. We iterated over all genes annotated as protein coding and computed a score with a nucleotide language model for each gene. To compute the score, we provided the language model with different levels of context: (i) the sequence of the gene only, (ii) the sequence of the gene plus equally distributed context on both sides of the gene up to a total 8192 bp, or (iii) the sequence of the gene plus equally distributed context on both sides of the gene up to a total 65,536 bp. If a gene extended beyond 8192 bp, we used the first 8192 bp of the gene sequences. We computed the score as the difference in log-likelihoods between a mutated sequence and the unmutated wild-type sequence. To mutate the sequence, we inserted multiple stop codons "TAATAATAATAGTGA" at an offset of 12 nucleotides into the sequence; for the 8192 and 65,536 bp context settings, we add context to both sides of the gene after the insertion. Additionally, for the 8192 bp setting, we tested three other strategies: (i) inserting a single stop codon "TAA" 12 nucleotides into the sequence, (ii) deleting the entire gene sequence (after which we provided 8192 context centered on the deleted gene) (fig. S27), or (iii) inserting stop codons tiled across the coding sequence at an interval of every 20 codons (or 60 bp) beginning with the first codon. As an additional control, we also used the gene's linear position in the reference genome as the value with which to predict essentiality. If a model were simply using positional information to make essentiality predictions, the performance would be similar to this control.

We also used the conservation of a gene as another control. To estimate conservation, we extracted all protein sequences from the OpenGenome dataset. For each genome corresponding to each essentiality study, we performed an all-by-all sequence search between all of the protein sequences in the genome-of-interest and all of the proteins in OpenGenome. To do this reasonably efficiently, we used mmseqs easy-search with default parameters, where the protein sequences in the genome-of-interest constituted the query sequences and the OpenGenome protein sequences constituted the target sequences. To compute the conservation of each gene, we counted the number of significant hits identified by mmseqs under a nominal E value threshold of 1×10^{-2}. We assumed that a greater number of hits corresponds to higher conservation, which in turn corresponds to greater essentiality.

We used the change in log-likelihoods (or the control "scores") to predict the binary gene essentiality labels and compute the strength of the prediction with the AUROC score and the average precision score as implemented in scikit-learn. We assessed statistical significance of the AUROC with a permutation-based method in which a null distribution is constructed by permuting the binary labels and recomputing the subsequent AUROC. We performed 100,000 permutations to construct this null distribution.

Genome-scale generation and evaluation

We used Evo pretrained at 131k context to sample sixteen sequences of lengths ~1 Mb. We sampled with a temperature of 1.0 and a top-k value of 4 following a standard autoregressive sampling procedure (*139*). We prompted the model with four species-specific prompts:

1) |d__Bacteria;p__Tenericutes;c__Mollicutes;o__Mycoplasmatales; f__Mycoplasmataceae;g__Mycoplasma;s__Mycoplasma genitalium||

2) |d__Bacteria;p__Bacillota;c__Bacilli;o__Staphylococcales;f__Staphylococcaceae; g__Staphylococcus;s__Staphylococcus aureus||

3) |d__Bacteria;p__Pseudomonadota;c__Gammaproteobacteria;o__Enterobacterales;f__Enterobacteriaceae;g__Klebsiella;s__Klebsiella pneumoniae||

4) |d__Bacteria;p__Pseudomonadota;c__Gammaproteobacteria;o__Enterobacterales;f__Enterobacteriaceae;g__Escherichia;s__Escherichia||

These prompts correspond to the species *Mycoplasma genitalium*, *Staphylococcus aureus*, *Klebsiella pneumoniae*, and *E. coli*, respectively, and follow Greengenes-style lineage strings, which concatenate all taxa starting with the most ancestral and ending with the most current, separated by semicolons. A single character prefix is also added to each taxon indicating its rank. These lineages strings were prepended to each contig during the 131k-context-extension phase of pretraining. We sampled four sequences for each prompt, leading to a total of sixteen sequences.

We evaluated these generations with CheckM (*77*), a tool that computes basic genome quality metrics based on whether a given long DNA sequence has similar properties as known bacterial genomes. CheckM uses Prodigal (*140*) to identify coding sequences and computes the coding density as one metric of genome quality. CheckM will also search for the presence of genes that are highly conserved across much of prokaryotic diversity. We divided all of our generations into discrete segments of up to 131,072 bp and computed the distribution of CheckM coding densities across these crops. As a positive control, we randomly selected 100 bacterial genomes from GTDB and used CheckM to compute the coding densities for 131,072 bp crops from these genomes. As a negative control, we generated 1000 sequences of length 131,072 in which the four DNA base pairs were sampled uniformly at random. We then used CheckM to compute the coding densities on this random sequence. We also used tRNAscan-SE to search for tRNA sequences in our generated sequences and we used barrnap to search for rRNA sequences.

We used ESMFold to obtain atomic-level structure predictions for all of the Prodigal-defined coding sequences in each of our generations. We limited ESMFold structure predictions to coding sequences between 100 and 1024 amino acids, inclusive. We computed the mean backbone pLDDT for all predicted structures. We used the biotite Python package to compute the percentages of secondary structure elements for all predicted structures. We used FoldSeek easy-search to perform efficient TM-based alignment (--alignment-type 1), and all other parameters set to their default values, to perform an all-by-all structural search between ESMFold structures corresponding to Evo-generated sequences and the structure predictions for UniRef50 provided in the AlphaFold Protein Structure Database (https://alphafold.ebi.ac.uk/). Structure alignments were scored as the average of the query TMscore and the target TMscore, where a score greater than 0.4 was considered a structural match. We used these structural matches, along with GO terms assigned to UniRef50 clusters, to

infer GO terms for the Evo-generated proteins as well. We used PyMOL to visualize protein structures corresponding to the five GO "molecular function" terms with the most representation among the Evo generated proteins.

We evaluated genomic sequence patterns including tetranucleotide and stop codon frequencies. Tetranucleotide usage deviations (TUDs) were calculated as previously described (*78*). TUD phylogenies were generated by hierarchical clustering using a distance matrix constructed from the Euclidean distances of log2transformed TUDs for each genome. Stop codon frequencies in the three reading frames of Prodigal-identified ORFs were stored as vectors consisting of nine scalar counts. Percentages of stop codons were calculated as the total sum of each stop codon (TAA, TAG, or TGA) relative to the total sum of all stop codons in a given vector. Stop codon ratios were calculated as the relative proportions of all nine scalars in a given vector.

REFERENCES AND NOTES

1. T. H. Morgan, Sex limited inheritance in *Drosophila*. *Science* **32**, 120–122 (1910). doi: 10.1126/science.32.812.120; pmid: 17759620
2. J. D. Watson, F. H. C. Crick, Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature* **171**, 737–738 (1953). doi: 10.1038/171737a0; pmid: 13054692
3. M. W. Nirenberg, J. H. Matthaei, The dependence of cell-free protein synthesis in *E. coli* upon naturally occurring or synthetic polyribonucleotides. *Proc. Natl. Acad. Sci. U.S.A.* **47**, 1588–1602 (1961). doi: 10.1073/pnas.47.10.1588; pmid: 14479932
4. T. Dobzhansky, *Genetics and the Origin of Species* (Columbia Univ. Press, 1951).
5. J. Jumper *et al*., Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021). doi: 10.1038/s41586-021-03819-2; pmid: 34265844
6. A. Rives *et al*., Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2016239118 (2021). doi: 10.1073/pnas.2016239118; pmid: 33876751
7. C. Outeiral, C. M. Deane, Codon language embeddings provide strong signals for use in protein engineering. *Nat. Mach. Intell.* **6**, 170–179 (2024). doi: 10.1038/s42256-024-00791-0
8. S. Li *et al*., CodonBERT large language model for mRNA vaccines. *Genome Res.* **34**, 1027–1035 (2024). doi: 10.1101/gr.278870.123; pmid: 38951026
9. Ž. Avsec *et al*., Effective gene expression prediction from sequence by integrating long-range interactions. *Nat. Methods* **18**, 1196–1203 (2021). doi: 10.1038/s41592-021-01252-x; pmid: 34608324
10. J. L. Watson *et al*., De novo design of protein structure and function with RFdiffusion. *Nature* **620**, 1089–1100 (2023). doi: 10.1038/s41586-023-06415-8; pmid: 37433327
11. A. Madani *et al*., Large language models generate functional protein sequences across diverse families. *Nat. Biotechnol.* **41**, 1099–1106 (2023). doi: 10.1038/s41587-022-01618-2; pmid: 36702895
12. J. B. Ingraham *et al*., Illuminating protein space with a programmable generative model. *Nature* **623**, 1070–1078 (2023). doi: 10.1038/s41586-023-06728-8; pmid: 37968394
13. L. F. DaSilva *et al*., DNA-Diffusion: Leveraging Generative Models for Controlling Chromatin Accessibility and Gene Expression via Synthetic Regulatory Elements. bioRxiv 2024.02.01.578352 [Preprint] (2024); https://doi.org/10.1101/2024.02.01.578352.
14. A. Lal, D. Garfield, T. Biancalani, G. Eraslan, regLM: Designing realistic regulatory DNA with autoregressive language models. bioRxiv 2024.02.14.580373 [Preprint] (2024); https://doi.org/10.1101/2024.02.14.580373.
15. M. Zvyagin *et al*., GenSLMs: Genome-scale language models reveal SARS-CoV-2 evolutionary dynamics. *Int. J. High Perform. Comput. Appl.* **37**, 683–705 (2023). doi: 10.1177/10943420231201154
16. H. Dalla-Torre *et al*., The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics. bioRxiv 2023.01.11.523679 [Preprint] (2023); https://doi.org/10.1101/2023.01.11.523679.
17. Z. Zhou, Y. Ji, W. Li, P. Dutta, R. Davuluri, H. Liu, DNABERT-2: Efficient foundation model and benchmark for multi-species genome. arXiv:2306.15006 [q-bio.GN] (2023).
18. Y. Tay *et al*., Charformer: Fast Character Transformers via Gradient-based Subword Tokenization. arXiv:2106.12672 [cs.CL] (2022).
19. S. Chen, S. Wong, L. Chen, Y. Tian, Extending context window of large language models via positional interpolation. arXiv:2306.15595 [cs.CL] (2023).
20. H. Liu, M. Zaharia, P. Abbeel, Ring attention with blockwise transformers for near-infinite context. arXiv:2310.01889 [cs.CL] (2023).
21. V. Fishman *et al*., GENA-LM: A family of open-source foundational DNA language models for long sequences. bioRxiv 2023.06.12.544594 [Preprint] (2024); https://doi.org/10.1101/2023.06.12.544594.
22. Y. Ji, Z. Zhou, H. Liu, R. V. Davuluri, DNABERT: Pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics* **37**, 2112–2120 (2021). doi: 10.1093/bioinformatics/btab083; pmid: 33538820
23. Y. Hwang, A. L. Cornman, E. H. Kellogg, S. Ovchinnikov, P. R. Girguis, Genomic language model predicts protein co-regulation and function. *Nat. Commun.* **15**, 2880 (2024). doi: 10.1038/s41467-024-46947-9; pmid: 38570504
24. M. Poli *et al*., StripedHyena: Moving Beyond Transformers with Hybrid Signal Processing Models, GitHub (2023); https://github.com/togethercomputer/stripedhyena.
25. Z. Li *et al*., Fourier neural operator for parametric partial differential equations. arXiv:2010.08895 [cs.LG] (2021).
26. A. Gu, K. Goel, C. Ré, Efficiently modeling long sequences with structured state spaces. arXiv:2111.00396 [cs.LG] (2022).
27. A. Orvieto *et al*., Resurrecting Recurrent Neural Networks for Long Sequences. arXiv:2303.06349 [cs.LG] (2023).
28. S. Massaroli *et al*., "Laughing Hyena Distillery: Extracting Compact Recurrences From Convolutions" in *Advances in Neural Information Processing Systems*, vol. 36, A. Oh *et al*., Eds. (Curran Associates, Inc., 2023), pp. 17072–17116.
29. J. Su *et al*., RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing* **568**, 127063 (2024). doi: 10.1016/j.neucom.2023.127063
30. X. Ma *et al*., Mega: Moving average equipped gated attention. arXiv:2209.10655 [cs.LG] (2023).
31. D. Y. Fu *et al*., Hungry hungry hippos: Towards language modeling with state space models. arXiv:2212.14052 [cs.LG] (2023).
32. J. Pilault *et al*., "Block-state transformers" in *Advances in Neural Information Processing Systems*, vol. 36, A. Oh *et al*., Eds. (Curran Associates, Inc., 2023), pp. 7311–7329.
33. E. Nguyen *et al*., "HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution" in *Advances in Neural Information Processing Systems*, vol. 36, A. Oh *et al*., Eds. (Curran Associates, Inc., 2023), pp. 43177–43201.
34. M. Poli *et al*., Hyena Hierarchy: Towards Larger Convolutional Language Models. arXiv:2302.10866 [cs.LG] (2023).
35. D. H. Parks *et al*., GTDB: An ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. *Nucleic Acids Res.* **50**, D785–D794 (2022). doi: 10.1093/nar/gkab776
36. A. P. Camargo *et al*., IMG/VR v4: An expanded database of uncultivated virus genomes within a framework of extensive functional, taxonomic, and ecological metadata. *Nucleic Acids Res.* **51**, D733–D743 (2023). doi: 10.1093/nar/gkac1037
37. A. P. Camargo *et al*., IMG/PR: A database of plasmids from genomes and metagenomes with rich annotations and metadata. *Nucleic Acids Res.* **52**, D164–D173 (2024). doi: 10.1093/nar/gkad964; pmid: 37930866
38. J. Hoffmann *et al*., Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs.CL] (2022).
39. J. Kaplan *et al*., Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG] (2020).
40. A. Gu, T. Dao, Mamba: Linear-time sequence modeling with selective state spaces. arXiv:2312.00752 [cs.LG] (2024).
41. J. Meier *et al*., "Language models enable zero-shot prediction of the effects of mutations on protein function" in *Advances in Neural Information Processing Systems*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, J. Wortman Vaughan, Eds. (Curran Associates, Inc., 2021), pp. 29287–29303.
42. P. Notin *et al*., "Tranception: Protein Fitness Prediction with Autoregressive Transformers and Inference-time Retrieval" in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, K. Chaudhuri *et al*., Eds. (PMLR, 2022), pp. 16990–17017.
43. G. Benegas, C. Albors, A. J. Aw, C. Ye, Y. S. Song, GPN-MSA: an alignment-based DNA language model for genome-wide variant effect prediction. bioRxiv 2023.10.10.561776 [Preprint] (2024); https://doi.org/10.1101/2023.10.10.561776.
44. P. Notin *et al*., ProteinGym: Large-Scale Benchmarks for Protein Design and Fitness Prediction. bioRxiv 2023.12.07.570727 [Preprint] (2023); https://doi.org/10.1101/2023.12.07.570727.
45. B. J. Livesey, J. A. Marsh, Updated benchmarking of variant effect predictors using deep mutational scanning. *Mol. Syst. Biol.* **19**, e11474 (2023). doi: 10.15252/msb.202211474; pmid: 37310135
46. K. K. Yang, N. Fusi, A. X. Lu, Convolutions are competitive with transformers for protein sequence pretraining. *Cell Syst.* **15**, 286–294.e2 (2024). doi: 10.1016/j.cels.2024.01.008; pmid: 38428432
47. Z. Lin *et al*., Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023). doi: 10.1126/science.ade2574; pmid: 36927031
48. E. Nijkamp, J. A. Ruffolo, E. N. Weinstein, N. Naik, A. Madani, ProGen2: Exploring the boundaries of protein language models. *Cell Syst.* **14**, 968–978.e3 (2023). doi: 10.1016/j.cels.2023.10.002; pmid: 37909046
49. F.-Z. Li, A. P. Amini, Y. Yue, K. K. Yang, A. X. Lu, Feature Reuse and Scaling: Understanding Transfer Learning with Protein Language Models. bioRxiv 2024.02.05.578959 [Preprint] (2024); https://doi.org/10.1101/2024.02.05.578959.
50. J. Chen *et al*., Interpretable RNA foundation model from unannotated data for highly accurate RNA structure and function predictions. arXiv:2204.00300 [q-bio.QM] (2022).
51. Z. D. Zhang, M. Nayar, D. Ammons, J. Rampersad, G. E. Fox, Rapid *in vivo* exploration of a 5S rRNA neutral network. *J. Microbiol. Methods* **76**, 181–187 (2009). doi: 10.1016/j.mimet.2008.10.010
52. T. L. LaFleur, A. Hossain, H. M. Salis, Automated model-predictive design of synthetic promoters to control transcriptional profiles in bacteria. *Nat. Commun.* **13**, 5159 (2022). doi: 10.1038/s41467-022-32829-5; pmid: 36056029
53. G. Urtecho, A. D. Tripp, K. D. Insigne, H. Kim, S. Kosuri, Systematic dissection of sequence elements controlling σ70 promoters using a genomically encoded multiplexed reporter assay in *Escherichia coli*. *Biochemistry* **58**, 1539–1551 (2018). doi: 10.1021/acs.biochem.7b01069; pmid: 29388765
54. A. Hossain *et al*., Automated design of thousands of nonrepetitive parts for engineering stable genetic systems. *Nat. Biotechnol.* **38**, 1466–1475 (2020). doi: 10.1038/s41587-020-0584-2; pmid: 32661437
55. T. C. Yu *et al*., Multiplexed characterization of rationally designed promoter architectures deconstructs combinatorial logic for IPTG-inducible systems. *Nat. Commun.* **12**, 325 (2021). doi: 10.1038/s41467-020-20094-3; pmid: 33436562
56. S. Kosuri *et al*., Composability of regulatory sequences controlling transcription and translation in *Escherichia coli*. *Proc. Natl. Acad. Sci. U.S.A.* **110**, 14024–14029 (2013). doi: 10.1073/pnas.1301301110; pmid: 23924614
57. H. M. Salis, E. A. Mirsky, C. A. Voigt, Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol.* **27**, 946–950 (2009). doi: 10.1038/nbt.1568
58. A. C. Reis, H. M. Salis, An automated model test system for systematic development and improvement of gene expression models. *ACS Synth. Biol.* **9**, 3145–3156 (2020). doi: 10.1021/acssynbio.0c00394; pmid: 33054181
59. J. Y. Wang, P. Pausch, J. A. Doudna, Structural biology of CRISPR–Cas immunity and genome editing enzymes. *Nat. Rev. Microbiol.* **20**, 641–656 (2022). doi: 10.1038/s41579-022-00739-4

60. P. D. Hsu, E. S. Lander, F. Zhang, Development and applications of CRISPR-Cas9 for genome engineering. *Cell* **157**, 1262–1278 (2014). doi: 10.1016/j.cell.2014.05.010; pmid: 24906146
61. E. V. Koonin, K. S. Makarova, Origins and evolution of CRISPR-Cas systems. *Phil. Trans. R. Soc. B* **374**, 20180087 (2019). doi: 10.1098/rstb.2018.0087
62. M. Jinek *et al*., A programmable dual-RNA-guided DNA endonuclease in adaptive bacterial immunity. *Science* **337**, 816–821 (2012). doi: 10.1126/science.1225829; pmid: 22745249
63. P. D. Hsu *et al*., DNA targeting specificity of RNA-guided Cas9 nucleases. *Nat. Biotechnol.* **31**, 827–832 (2013). doi: 10.1038/nbt.2647; pmid: 23873081
64. J. Abramson *et al*., Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630**, 493–500 (2024). doi: 10.1038/s41586-024-07487-w; pmid: 38718835
65. N. L. Craig *et al*., Eds., *Mobile DNA III* (Wiley, ed. 3, 2020).
66. C. Meers *et al*., Transposon-encoded nucleases use guide RNAs to promote their selfish spread. *Nature* **622**, 863–871 (2023). doi: 10.1038/s41586-023-06597-1; pmid: 37758954
67. T. Karvelis *et al*., Transposon-associated TnpB is a programmable RNA-guided DNA endonuclease. *Nature* **599**, 692–696 (2021). doi: 10.1038/s41586-021-04058-1; pmid: 34619744
68. H. Altae-Tran *et al*., The widespread IS200/IS605 transposon family encodes diverse programmable RNA-guided endonucleases. *Science* **374**, 57–65 (2021). doi: 10.1126/science.abj6856
69. O. Barabas *et al*., Mechanism of IS*200*/IS*605* family DNA transposases: Activation and transposon-directed target site selection. *Cell* **132**, 208–220 (2008). doi: 10.1016/j.cell.2007.12.029; pmid: 18243097
70. Z. Zhang *et al*., Protein language models learn evolutionary statistics of interacting sequence motifs. *Proc. Natl. Acad. Sci. U.S.A.* **121**, e2406285121 (2024). doi: 10.1073/pnas.2406285121
71. P. Siguier, J. Pérochon, L. Lestrade, J. Mahillon, M. Chandler, ISfinder: The reference centre for bacterial insertion sequences. *Nucleic Acids Res.* **34**, D32–D36 (2006). doi: 10.1093/nar/gkj014; pmid: 16381877
72. E. P. C. Rocha, A. Danchin, Gene essentiality determines chromosome organisation in bacteria. *Nucleic Acids Res.* **31**, 6570–6577 (2003). doi: 10.1093/nar/gkg859; pmid: 14602916
73. R. Zhang, H.-Y. Ou, C.-T. Zhang, DEG: A database of essential genes. *Nucleic Acids Res.* **32**, D271–D272 (2004). doi: 10.1093/nar/gkh024
74. D. Piya *et al*., Systematic and scalable genome-wide essentiality mapping to identify nonessential genes in phages. *PLOS Biol.* **21**, e3002416 (2023). doi: 10.1371/journal.pbio.3002416; pmid: 38048319
75. K. H. Turner, A. K. Wessel, G. C. Palmer, J. L. Murray, M. Whiteley, Essential genome of *Pseudomonas aeruginosa* in cystic fibrosis sputum. *Proc. Natl. Acad. Sci. U.S.A.* **112**, 4110–4115 (2015). doi: 10.1073/pnas.1419677112; pmid: 25775563
76. A. Blanchard, C. Bébéar, The evolution of *Mycoplasma genitalium*. *Ann. N. Y. Acad. Sci.* **1230**, E61–E64 (2011). doi: 10.1111/j.1749-6632.2011.06418.x; pmid: 22417108
77. D. H. Parks, M. Imelfort, C. T. Skennerton, P. Hugenholtz, G. W. Tyson, CheckM: Assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome Res.* **25**, 1043–1055 (2015). doi: 10.1101/gr.186072.114
78. D. T. Pride, R. J. Meinersmann, T. M. Wassenaar, M. J. Blaser, Evolutionary implications of microbial genome tetranucleotide frequency biases. *Genome Res.* **13**, 145–158 (2003). doi: 10.1101/gr.335003
79. L. Xu, J. Kuo, J.-K. Liu, T.-Y. Wong, Bacterial phylogenetic tree construction based on genomic translation stop signals. *Microb. Inform. Exp.* **2**, 6 (2012). doi: 10.1186/2042-5783-2-6
80. G. Korkmaz, M. Holm, T. Wiens, S. Sanyal, Comprehensive analysis of stop codon usage in bacteria and its correlation with release factor abundance. *J. Biol. Chem.* **289**, 30334–30342 (2014). doi: 10.1074/jbc.M114.606632 pmid: 25217634
81. T. Seemann, barrnap, GitHub (2018); https://github.com/tseemann/barrnap.
82. N. Goldman, J. L. Thorne, D. T. Jones, Assessing the impact of secondary structure and solvent accessibility on protein evolution. *Genetics* **149**, 445–458 (1998). doi: 10.1093/genetics/149.1.445; pmid: 9584116
83. J. Wei *et al*., Finetuned language models are zero-shot learners. arXiv:2109.01652 [cs.CL] (2022).
84. L. Ouyang *et al*., Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL] (2022).
85. R. Rafailov *et al*., "Direct Preference Optimization: Your Language Model is Secretly a Reward Model" in *Advances in Neural Information Processing Systems*, vol. 36, A. Oh *et al*., Eds. (Curran Associates, Inc., 2023), pp. 53728–53741.
86. H. L. Rehm *et al*., GA4GH: International policies and standards for data sharing across genomic research and healthcare. *Cell Genomics* **1**, 100029 (2021). doi: 10.1016/j.xgen.2021.100029
87. T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners" in *Advances in Neural Information Processing Systems*, vol. 35, S. Koyejo *et al*., Eds. (Curran Associates, Inc., 2022), pp. 22199–22213.
88. B. L. Hie *et al*., Efficient evolution of human antibodies from general protein language models. *Nat. Biotechnol.* **42**, 275–283 (2024). doi: 10.1038/s41587-023-01763-2
89. V. R. Shanker, T. U. J. Bruun, B. L. Hie, P. S. Kim, Unsupervised evolution of protein and antibody complexes with a structure-informed language model. *Science* **385**, 46–53 (2024). doi: 10.1126/science.adk8946; pmid: 38963838
90. M. G. Durrant *et al*., Bridge RNAs direct programmable recombination of target and donor DNA. *Nature* **630**, 984–993 (2024). doi: 10.1038/s41586-024-07552-4; pmid: 38926615
91. Y. N. Dauphin, A. Fan, M. Auli, D. Grangier, "Language Modeling with Gated Convolutional Networks" in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, D. Precup, Y. W. Teh, Eds. (PMLR, 2017), pp. 933–941.
92. N. Shazeer, GLU variants improve Transformer. arXiv:2002.05202 [cs.LG] (2020).
93. B. Zhang, R. Sennrich, "Root Mean Square Layer Normalization" in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach *et al*., Eds. (Curran Associates, Inc., 2019).
94. D. Fu *et al*., "Monarch Mixer: A simple sub-quadratic GEMM-based architecture" in *Advances in Neural Information Processing Systems*, vol. 36, A. Oh *et al*., Eds. (Curran Associates, Inc., 2023), pp. 77546–77603.
95. S. Arora *et al*., Zoology: Measuring and improving recall in efficient language models. arXiv:2312.04927 [cs.CL] (2023).
96. S. Bhattamishra, A. Patel, P. Blunsom, V. Kanade, Understanding in-context learning in transformers and LLMs by learning to learn discrete functions. arXiv:2310.03016 [cs.LG] (2023).
97. D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, M. Hoogendoorn, CKConv: Continuous kernel convolution for sequential data. arXiv:2102.02611 [cs.LG] (2022).
98. A. Gupta, A. Gu, J. Berant, "Diagonal State Spaces are as Effective as Structured State Spaces" in *Advances in Neural Information Processing Systems*, vol. 35, S. Koyejo *et al*., Eds. (Curran Associates, Inc., 2022), pp. 22982–22994.
99. A. Gu, K. Goel, A. Gupta, C. Ré, "On the Parameterization and Initialization of Diagonal State Space Models" in *Advances in Neural Information Processing Systems*, vol. 35, S. Koyejo *et al*., Eds. (Curran Associates, Inc., 2022), pp. 35971–35983.
100. M. Zhang *et al*., Effectively modeling time series with simple discrete state spaces. arXiv:2303.09489 [cs.LG] (2023).
101. J. Wei *et al*., Deep learning and CRISPR-Cas13d ortholog discovery for optimized RNA targeting. *Cell Syst.* **14**, 1087–1102.e13 (2023). doi: 10.1016/j.cels.2023.11.006; pmid: 38091991
102. N. A. O'Leary *et al*., Reference sequence (RefSeq) database at NCBI: Current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.* **44**, D733–D745 (2016). doi: 10.1093/nar/gkv1189
103. A. Almeida *et al*., A unified catalog of 204,938 reference genomes from the human gut microbiome. *Nat. Biotechnol.* **39**, 105–114 (2021). doi: 10.1038/s41587-020-0603-3; pmid: 32690973
104. I.-M. A. Chen *et al*., The IMG/M data management and analysis system v.6.0: New tools and advanced capabilities. *Nucleic Acids Res.* **49**, D751–D763 (2021). doi: 10.1093/nar/gkaa939
105. L. F. Camarillo-Guerrero, A. Almeida, G. Rangel-Pineros, R. D. Finn, T. D. Lawley, Massive expansion of human gut bacteriophage diversity. *Cell* **184**, 1098–1109.e9 (2021). doi: 10.1016/j.cell.2021.01.029
106. S. C. Forster *et al*., A human gut bacterial genome and culture collection for improved metagenomic analyses. *Nat. Biotechnol.* **37**, 186–192 (2019). doi: 10.1038/s41587-018-0009-7
107. A. L. Mitchell *et al*., MGnify: The microbiome analysis resource in 2020. *Nucleic Acids Res.* **48**, D570–D578 (2020). doi: 10.1093/nar/gkz1035
108. N. D. Youngblut *et al*., Large-scale metagenome assembly reveals novel animal-associated microbial genomes, biosynthetic gene clusters, and other genetic diversity. *mSystems* **5**, e01045-20 (2020). doi: 10.1128/msystems.01045-20
109. F. Meyer *et al*., The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics* **9**, 386 (2008). doi: 10.1186/1471-2105-9-386; pmid: 18803844
110. S. Sunagawa *et al*., Structure and function of the global ocean microbiome. *Science* **348**, 1261359 (2015). doi: 10.1126/science.1261359; pmid: 25999513
111. R. D. Finn, J. Clements, S. R. Eddy, HMMER web server: Interactive sequence similarity searching. *Nucleic Acids Res.* **39**, W29–W37 (2011). doi: 10.1093/nar/gkr367
112. J. Russel, R. Pinilla-Redondo, D. Mayo-Muñoz, S. A. Shah, S. J. Sørensen, CRISPRCasTyper: Automated Identification, Annotation, and Classification of CRISPR-Cas Loci. *CRISPR J.* **3**, 462–469 (2020). doi: 10.1089/crispr.2020.0059; pmid: 33275853
113. H. Altae-Tran *et al*., Diversity, evolution, and classification of the RNA-guided nucleases TnpB and Cas12. *Proc. Natl. Acad. Sci. U.S.A.* **120**, e2308224120 (2023). doi: 10.1073/pnas.2308224120; pmid: 37983496
114. M. Steinegger, J. Söding, MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028 (2017). doi: 10.1038/nbt.3988; pmid: 29035372
115. K. Katoh, K. Misawa, K. Kuma, T. Miyata, MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.* **30**, 3059–3066 (2002). doi: 10.1093/nar/gkf436; pmid: 12136088
116. W. Xiong *et al*., Effective long-context scaling of foundation models. arXiv:2309.16039 [cs.CL] (2023).
117. J. Ainslie *et al*., GQA: Training generalized multi-query transformer models from multi-head checkpoints. arXiv:2305.13245 [cs.CL] (2023).
118. E. Firnberg, J. W. Labonte, J. J. Gray, M. Ostermeier, A comprehensive, high-resolution map of a gene's fitness landscape. *Mol. Biol. Evol.* **31**, 1581–1592 (2014). doi: 10.1093/molbev/msu081; pmid: 24567513
119. H. Jacquier *et al*., Capturing the mutational landscape of the beta-lactamase TEM-1. *Proc. Natl. Acad. Sci. U.S.A.* **110**, 13067–13072 (2013). doi: 10.1073/pnas.1215206110
120. B. V. Adkar *et al*., Protein model discrimination using mutational sensitivity derived from deep sequencing. *Structure* **20**, 371–381 (2012). doi: 10.1016/j.str.2011.11.021
121. K. Tsuboyama *et al*., Mega-scale experimental analysis of protein folding stability in biology and design. *Nature* **620**, 434–444 (2023). doi: 10.1038/s41586-023-06328-6; pmid: 37468638
122. E. D. Kelsic *et al*., RNA structural determinants of optimal codons revealed by MAGE-Seq. *Cell Syst.* **3**, 563–571.e6 (2016). doi: 10.1016/j.cels.2016.11.004
123. R. Weeks, M. Ostermeier, Fitness and functional landscapes of the *E. coli* RNase III gene *rnc*. *Mol. Biol. Evol.* **40**, msad047 (2023). doi: 10.1093/molbev/msad047; pmid: 36848192
124. L. Rockah-Shmuel, Á. Tóth-Petróczy, D. S. Tawfik, Systematic mapping of protein mutational space by prolonged drift reveals the deleterious effects of seemingly neutral mutations. *PLOS Comput. Biol.* **11**, e1004421 (2015). doi: 10.1371/journal.pcbi.1004421; pmid: 26274323
125. J. Z. Chen, D. M. Fowler, N. Tokuriki, Comprehensive exploration of the translocation, stability and substrate recognition requirements in VIM-2 lactamase. *eLife* **9**, e56707 (2020). doi: 10.7554/eLife.56707; pmid: 32510322
126. A. Melnikov, P. Rogov, L. Wang, A. Gnirke, T. S. Mikkelsen, Comprehensive mutational scanning of a kinase *in vivo* reveals substrate-dependent fitness landscapes. *Nucleic Acids Res.* **42**, e112 (2014). doi: 10.1093/nar/gku511; pmid: 24914046
127. S. Sun *et al*., A proactive genotype-to-patient-phenotype map for cystathionine beta-synthase. *Genome Med.* **12**, 13 (2020). doi: 10.1186/s13073-020-0711-1

128. R. A. Silverstein *et al*., A systematic genotype-phenotype map for missense variants in the human intellectual disability-associated gene GDI1. bioRxiv 2021.10.06.463360 [Preprint] (2022); https://doi.org/10.1101/2021.10.06.463360.
129. C. W. Garvie *et al*., Structure of PDE3A-SLFN12 complex reveals requirements for activation of SLFN12 RNase. *Nat. Commun.* **12**, 4375 (2021). doi: 10.1038/s41467-021-24495-w
130. E. Kotler *et al*., A systematic p53 mutation library links differential functional impact to cancer mutation pattern and evolutionary conservation. *Mol. Cell* **71**, 178–190.e8 (2018). doi: 10.1016/j.molcel.2018.06.012; pmid: 29979965
131. A. O. Giacomelli *et al*., Mutational processes shape the landscape of *TP53* mutations in human cancer. *Nat. Genet.* **50**, 1381–1387 (2018). doi: 10.1038/s41588-018-0204-y
132. G. M. Findlay *et al*., Accurate classification of *BRCA1* variants with saturation genome editing. *Nature* **562**, 217–222 (2018). doi: 10.1038/s41586-018-0461-z
133. S. Kobori, Y. Nomura, A. Miu, Y. Yokobayashi, High-throughput assay and engineering of self-cleaving ribozymes by sequencing. *Nucleic Acids Res.* **43**, e85 (2015). doi: 10.1093/nar/gkv265; pmid: 25829176
134. J. O. L. Andreasson, A. Savinov, S. M. Block, W. J. Greenleaf, Comprehensive sequence-to-function mapping of cofactor-dependent RNA catalysis in the *glmS* ribozyme. *Nat. Commun.* **11**, 1663 (2020). doi: 10.1038/s41467-020-15540-1
135. J. Domingo, G. Diss, B. Lehner, Pairwise and higher-order genetic interactions during the evolution of a tRNA. *Nature* **558**, 117–121 (2018). doi: 10.1038/s41586-018-0170-7; pmid: 29849145
136. M. P. Guy *et al*., Identification of the determinants of tRNA function and susceptibility to rapid tRNA decay by high-throughput in vivo analysis. *Genes Dev.* **28**, 1721–1732 (2014). doi: 10.1101/gad.245936.114
137. E. J. Hayden, E. Ferrada, A. Wagner, Cryptic genetic variation promotes rapid evolutionary adaptation in an RNA enzyme. *Nature* **474**, 92–95 (2011). doi: 10.1038/nature10083; pmid: 21637259
138. J. N. Pitt, A. R. Ferré-D'Amaré, Rapid construction of empirical RNA fitness landscapes. *Science* **330**, 376–379 (2010). doi: 10.1126/science.1192001; pmid: 20947767
139. T. A. Chang, B. K. Bergen, Language model behavior: A comprehensive survey. arXiv:2303.11504 [cs.CL] (2023).
140. D. Hyatt *et al*., Prodigal: Prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics* **11**, 119 (2010). doi: 10.1186/1471-2105-11-119; pmid: 20211023
141. C. Bland *et al*., CRISPR recognition tool (CRT): A tool for automatic detection of clustered regularly interspaced palindromic repeats. *BMC Bioinformatics* **8**, 209 (2007). doi: 10.1186/1471-2105-8-209; pmid: 17577412
142. P. Kunzmann *et al*., Biotite: New tools for a versatile Python bioinformatics library. *BMC Bioinformatics* **24**, 236 (2023). doi: 10.1186/s12859-023-05345-6
143. A. Mitrofanov, M. Ziemann, O. S. Alkhnbashi, W. R. Hess, R. Backofen, CRISPRtracrRNA: Robust approach for CRISPR tracrRNA detection. *Bioinformatics* **38**, ii42–ii48 (2022). doi: 10.1093/bioinformatics/btac466; pmid: 36124799
144. R. Lorenz *et al*., ViennaRNA Package 2.0. *Algorithms Mol. Biol.* **6**, 26 (2011). doi: 10.1186/1748-7188-6-26; pmid: 22115189
145. E. P. Nawrocki, S. R. Eddy, Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics* **29**, 2933–2935 (2013). doi: 10.1093/bioinformatics/btt509; pmid: 24008419
146. C. Zhang, M. Shine, A. M. Pyle, Y. Zhang, US-align: Universal structure alignments of proteins, nucleic acids, and macromolecular complexes. *Nat. Methods* **19**, 1109–1115 (2022). doi: 10.1038/s41592-022-01585-1; pmid: 36038728
147. Schrödinger LLC, The PyMOL Molecular Graphics System, version 1.8 (2015).
148. A. R. Gruber, R. Lorenz, S. H. Bernhart, R. Neuböck, I. L. Hofacker, The Vienna RNA Websuite. *Nucleic Acids Res.* **36**, W70–W74 (2008). doi: 10.1093/nar/gkn188
149. W. B. Langdon, J. Petke, R. Lorenz, in *Genetic Programming*, M. Castelli, L. Sekanina, M. Zhang, S. Cagnoni, P. García-Sánchez, Eds. (Springer, 2018), pp. 220–236.
150. Z. Weinberg, R. R. Breaker, R2R - Software to speed the depiction of aesthetic consensus RNA secondary structures. *BMC Bioinformatics* **12**, 3 (2011). doi: 10.1186/1471-2105-12-3; pmid: 21205310
151. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990). doi: 10.1016/S0022-2836(05)80360-2
152. M. Mirdita *et al*., ColabFold: Making protein folding accessible to all. *Nat. Methods* **19**, 679–682 (2022). doi: 10.1038/s41592-022-01488-1; pmid: 35637307
153. B. Hie, Code for paper "Sequence modeling and design from molecular to genome scale with Evo," Zenodo (2024); https://doi.org/10.5281/zenodo.12693561.

ACKNOWLEDGMENTS

We thank E. Chanakira, D. Driggers, R. Dugan, H. Fritz, M. Iskender, A. Jain, M. LaPan, S. Marrs, S. Perelson, R. Rizun, J. Rojas, and D. Ugelstad for assistance with computational infrastructure. We thank S. Sternberg and C. Meers for providing covariance models to identify diverse ωRNAs. We thank M. White for contributing purification plasmids. We thank J. Adkins, J. Carvalho, D. Fu, J. Dunnmon, Y. Hwang, J. Kazaks, G. Machiraju, A. Merchant, A. Patel, A. Pawluk, C. Ricci-Tam, C. Theodoris, B. Viggiano, and A. Woodrow for helpful discussions and assistance with manuscript preparation. **Funding:** This work was supported by the Fannie and John Hertz Foundation (D.B.L.); National Science Foundation Graduate Fellowship Program (S.H.K. and G.B.); National Center for Advancing Translational Sciences of the National Institutes of Health, award no. UL1TR003142 (T.H.-B.); National Institutes of Health grant U54EB020405 (C.R.); National Science Foundation grants CCF2247015, CCF1763315, CCF1563078, and 1937301 (C.R.); US DEVCOM Army Research Laboratory grants W911NF-23-2-0184 and W911NF-21-2-0251 (C.R.); ONR grant N000142312633 (C.R.); Stanford HAI grant 247183 (C.R.); NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-GCP Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Meta, Google, and VMWare (C.R.); the Arc Institute (P.D.H. and B.L.H.); the Rainwater Foundation (P.D.H.); the Curci Foundation (P.D.H.); Rose Hill Investigators Program (P.D.H.); V. and N. Khosla (P.D.H.); S. Altman (P.D.H.); anonymous gifts to the Hsu laboratory (P.D.H.); V. Gupta (B.L.H.); and R. Tonsing (B.L.H.). **Author contributions:** E.N., P.D.H., and B.L.H. conceived the project. P.D.H. and B.L.H. supervised the project. E.N., M.P., and A.W.T. designed the model architecture. M.G.D. and B.L.H. curated and processed the pretraining and fine-tuning datasets. M.P. implemented the optimized training and generation code. E.N., A.W.T., G.B., and B.L.H. contributed to the optimized training and generation code. J.S. set up and managed the hardware infrastructure and training environment. E.N., M.P., and A.W.T. implemented and carried out the scaling laws analysis. B.L.H. evaluated pretrained and fine-tuned models on molecular prediction tasks. E.N., B.K., and B.L.H. conducted model fine-tuning. B.K. and P.D.H. sampled or analyzed CRISPR-Cas generations. B.K., D.K., and L.J.B. experimentally tested the CRISPR-Cas generations. M.G.D., D.B.L., and B.L.H. sampled or analyzed the IS200/IS605 generations. D.B.L. experimentally tested the IS200/IS605 generations. B.L.H. conducted the gene essentiality analysis. M.P., S.H.K., and B.L.H. conducted genome-scale sampling and analysis. M.P., A.W.T., and B.L.H. implemented the public Evo codebase. M.Y.N., A.Le., and T.H.-B. conducted the ethics and safety investigation and discussion. E.N., M.P., M.G.D., P.D.H., and B.L.H. wrote the first draft of the manuscript. All authors wrote the final draft of the manuscript. **Competing interests:** M.P. carried out the work during employment at Together AI. M.G.D. acknowledges outside interest in Stylus Medicine. D.K. acknowledges outside interest in Shape Therapeutics. C.R. acknowledges outside interest in Factory and Google Ventures. P.D.H. acknowledges outside interest as a cofounder of Stylus Medicine, Circle Labs, and Spotlight Therapeutics; serves on the board of directors at Stylus Medicine; is a board observer at EvolutionaryScale, Circle Labs, and Spotlight Therapeutics; is a scientific advisory board member at Arbor Biosciences and Veda Bio; and is an advisor to NFDG, Varda Space, and Vial Health. B.L.H. acknowledges outside interest in Prox Biosciences as a scientific cofounder. B.L.H., P.D.H., B.K., D.K., and L.J.B. are named as inventors on provisional patent application 63/688,826 applied for by Stanford University and Arc Institute related to EvoCas9-1. All other authors declare no competing interests. **Data and materials availability:** Code and models related to this study are publicly available at https://github.com/evo-design/evo and uploaded to Zenodo (*153*). The following models have been uploaded to Hugging Face under an open-source license: pretrained Evo model with 8k context (https://huggingface.co/togethercomputer/evo-1-8k-base); pretrained Evo model with 131k context (https://huggingface.co/togethercomputer/evo-1-131k-base); fine-tuned Evo model on CRISPR-Cas systems (https://huggingface.co/LongSafari/evo-1-8k-crispr); and fine-tuned Evo model on IS200/IS605 systems (https://huggingface.co/LongSafari/evo-1-8k-transposon). The OpenGenome dataset, including training, validation, and test splits, is publicly available on Hugging Face Datasets at https://huggingface.co/datasets/LongSafari/open-genome. We used the following publicly available datasets for pretraining: bacterial and archaeal genomes from the Genome Taxonomy Database (GTDB) v214.1 (*35*); curated prokaryotic viruses from the IMG/VR v4 database (*36*); and plasmid sequences from the IMG/PR database (*37*). In addition to the above datasets, we also used portions of the following datasets for fine-tuning: NCBI RefSeq (*102*), UHGG (*103*), JGI IMG (*104*), The Gut Phage Database (*105*), The Human Gastrointestinal Bacteria Genome Collection (*106*), MGnify (*107*), Animal gut metagenomes (*108*), MGRAST (*109*), and Tara Oceans samples (*110*). Additional details on these datasets are provided in Materials and methods. DNA, RNA, and protein sequences generated during our validation experiments are available in data S1. All newly created materials are available upon reasonable request to the corresponding authors.

SUPPLEMENTARY MATERIALS

science.org/doi/10.1126/science.ado9336
Supplementary Text
Figs. S1 to S28
Tables S1 to S7
References (*154*–*159*)
MDAR Reproducibility Checklist
Data S1

Submitted 27 February 2024; accepted 9 September 2024
10.1126/science.ado9336