Supplemental Information

For the paper "A generative neural network for maximizing fitness and diversity of synthetic DNA and protein sequences".

Supplemental Figures

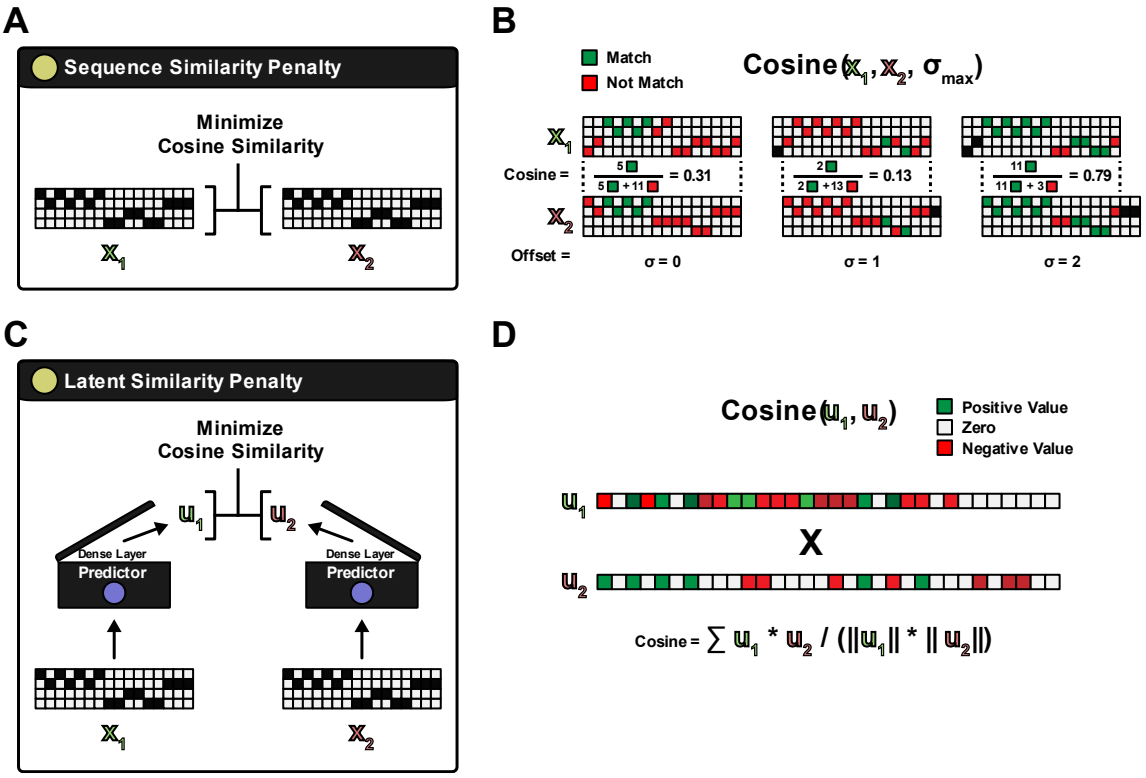


Figure S1. Diversity Cost Functions. Related to Figure 1. (A) Sequence similarity cost function: The cosine similarity penalty is applied directly on the one-hot representation of the generated sequence pair (x_1 and x_2). The sequence pair was produced by the generator given two input vectors (z_1 and z_2) as seeds. **(B)** Multiple offsets are considered when calculating the cosine similarity between one-hot sequences x_1 and x_2. For a given offset, x_1 and x_2 are multiplied elementwise, aggregated and normalized by the sequence length. Green squares indicate matching nucleotides between x_1 and x_2 (which contributes to the similarity). The worst-case (highest) value across all considered offsets is chosen as the cosine similarity. **(C)** Latent similarity cost function: The cosine similarity penalty is applied on the latent feature vectors (u_1 and u_2) produced by the activations of one of the fully connected layers within the predictor, when running the model on the generated sequence patterns x_1 and x_2. **(D)** Vectors u_1 and u_2 are multiplied elementwise and summed. The sum is normalized by the L2-norm of u_1 multiplied by the L2-norm of u_2, resulting in a cosine similarity metric.

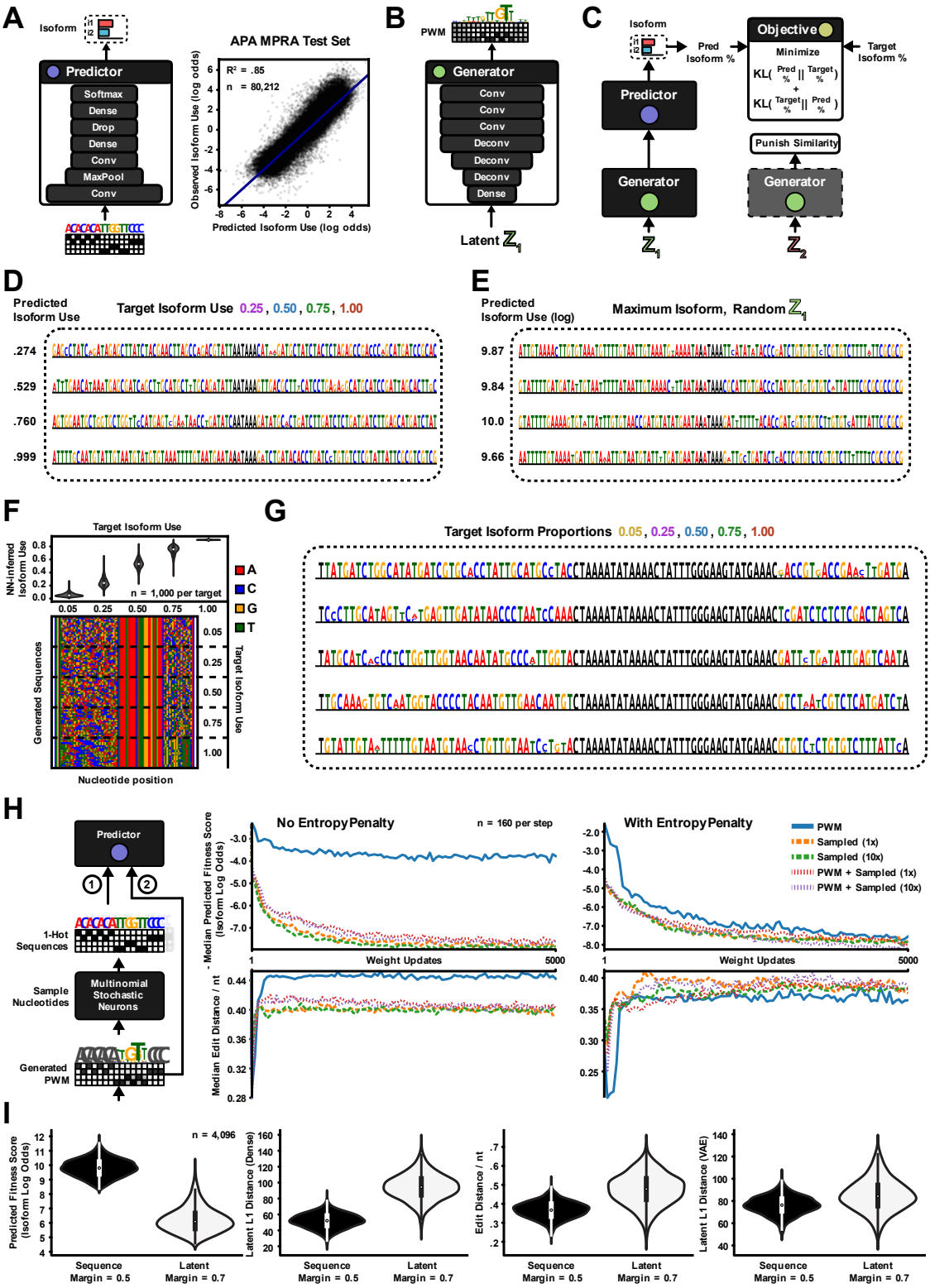

A
Isoform
Predictor
Softmax
Dense
Drop
Dense
Conv
MaxPool
Conv
APA MPRA Test Set
R² = .85
n = 80,212
Observed Isoform Use (log odds)
Predicted Isoform Use (log odds)
B
PWM
Generator
Conv
Conv
Conv
Deconv
Deconv
Deconv
Dense
Latent Z1
C
Isoform
Pred Isoform %
Objective
Minimize
Target Isoform %
Predictor
Generator
Z1
Punish Similarity
Generator
Z2
D
Predicted Isoform Use
Target Isoform Use 0.25, 0.50, 0.75, 1.00
.274
.529
.760
.999
E
Predicted Isoform Use (log)
Maximum Isoform, Random Z1
9.87
9.84
10.0
9.66
F
Target Isoform Use
NN-inferred Isoform Use
n = 1,000 per target
Generated Sequences
Target Isoform Use
Nucleotide position
A
C
G
T
G
Target Isoform Proportions 0.05, 0.25, 0.50, 0.75, 1.00
H
Predictor
1-Hot Sequences
Sample Nucleotides
Multinomial Stochastic Neurons
Generated PWM
No Entropy Penalty
n = 160 per step
With Entropy Penalty
PWM
Sampled (1x)
Sampled (10x)
PWM + Sampled (1x)
PWM + Sampled (10x)
- Median Predicted Fitness Score (Isoform Log Odds)
Weight Updates
Median Edit Distance / nt
I
Predicted Fitness Score (Isoform Log Odds)
n = 4,096
Latent L1 Distance (Dense)
Edit Distance / nt
Latent L1 Distance (VAE)
Sequence Margin = 0.5
Latent Margin = 0.7

Figure S2. Generator and Predictor Architectures, Generation Examples, Training Curves and Latent Diversity Costs. Related to Figure 2. (A) Left: The APA predictor. Convolutional, pooling, dropout and dense layers transform the one-hot coded input sequence into an APA isoform proportion prediction. Right: Predicted vs. observed isoform log odds on the test set of (Bogard et al., 2019), $R^2 = 0.85$, n = 80,212. **(B)** The generator follows a typical DC-GAN architecture. Dense and (de-)convolutional layers transform the input seed vector into a sequence PWM of nucleotide log probabilities. **(C)** The one-hot sequence outputted by the generator is passed as input to the APA fitness predictor, which in turn outputs an isoform proportion prediction. This prediction is used in a symmetric KL-divergence loss to fit the generator to a fixed target isoform proportion. **(D)** Example sequences generated by four of the five deep exploration network instances evaluated in Figure 2B: The 25%, 50%, 75% and 100% ('Max') generators. **(E)** Example sequences generated by the 100%-target ('Max') APA isoform DEN, using four different random seeds as input to the generator. **(F)** DENs were validated against real RNA-Seq measurements using a nearest neighbor approximation. Five new generators were trained to produce polyA signals for the APA isoform targets 5%, 25%, 50%, 75%, 100% ('Max'), but this time with a shorter (60 nt) freely tunable sequence so as to reduce "curse of dimensionality"-effects in the nearest neighbor search. The first four DENs were trained with 30% allowable sequence similarity margin. The Max-target DEN was trained with 50% similarity margin. The subset of measured APA sequences from (Bogard et al., 2019) with a TOMM5 3' UTR template (which only has 60 nt of randomized sequence) was collected (n = 125,949) and the first dense layer activations predicted by APARENT were used as features when storing the sequences in a nearest neighbor database. Next, 1,000 sequences were generated from each of the five DENs. The 50 nearest neighbors from the measured dataset were looked up for each of the 1,000 generated sequences, and the mean measured isoform proportion of those 50 neighbors were used as the estimate for each generated sequence. (Top) Measured mean isoform proportions (from RNA-Seq data) for the 50 nearest neighbors of each of the generated sequences (n = 1,000). Mean and Std dev of isoform proportions: (Target 5%) 6.00% +- 3.03%, (Target 25%) 25.1% +- 6.95%, (Target 50%) 53.2% +- 7.05%, (Target 75%) 73.7% +- 7.18%, (Target Max) 88.05% +- 0.05%. (Bottom) Sequence diversity for each of the five generators, illustrated by sampling 20 sequences per target/generator and plotting the nucleotides on a 2-dimensional pixel grid. The first four target-isoform generators have 0% duplication rate at 100,000 sampled sequences, and the 100%-target ('Max') generator has a 0.1% duplication rate at 100,000 samples. Hexamer entropy ranged between 7.89 and 9.00 bits depending on the generator (of 12 bits max). **(G)** Example sequences generated by the five DENs trained in Figure S2F. **(H)** Evaluation of three different sequence pattern representations: (1) Sampling a number of discrete one-hot-coded patterns from the generated PWM (1 or 10 samples) and using them as input to the predictor (gradients are propagated by a softmax straight-through gradient estimator), (2) Passing the softmax-relaxed PWM directly as input or (3) passing both representations as input to the predictor and walking down the average gradient. The methods were evaluated by training a DEN to generate maximally used (100%-target) polyA signals with a 50% sequence similarity margin, and tracking the predicted median fitness score (isoform log odds) and median normalized sequence edit distance across the first 5,000 weight updates. The metrics were computed on a random sample of 160 generator seeds per weight update (step). Note that, regardless of input representation, the metrics were always computed using the consensus one-hot-coded pattern extracted from the generated PWMs (guaranteeing well-formed input to the predictor). Reported are the median curves of five independent runs. (Left curves) The results of training the DEN without any entropy penalty. (Right curves) The results of training when enforcing an

entropy penalty on the generated PWM, with a target mean nucleotide conservation of 1.95 bits. **(I)** A comparison of the Max-target APA isoform DEN when trained with two different diversity costs: In one setting (black violin), the DEN was trained with the sequence cosine similarity penalty (see Figure S1A-B) with an allowable margin of 0.6 (allowing 60% of nucleotides to be identical without incurring cost). In the second setting (white violin), the DEN was trained with the latent cosine similarity penalty (see Figure S1C-D) with an allowable margin of 0.7 (allowing a 70% cosine similarity without incurring cost). The two DENs were used to generate 4,096 sequences each. Shown are the predicted fitness scores (isoform log odds), pairwise L1-distances of latent feature vectors from the penalized latent space, normalized sequence edit distances, and pairwise L1-distances in an independently trained APA variational autoencoder (see STAR methods for details on how the autoencoder is trained).

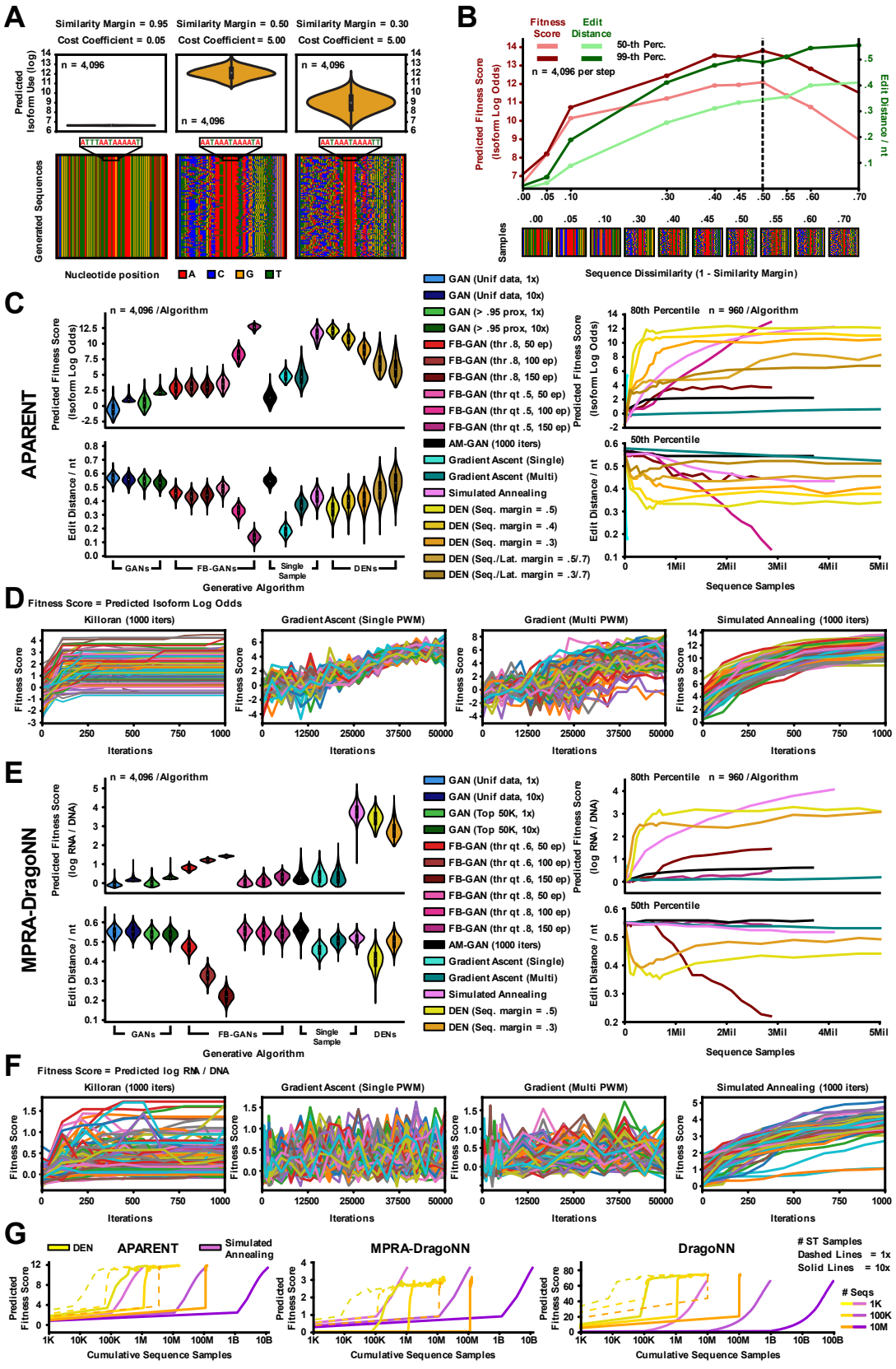
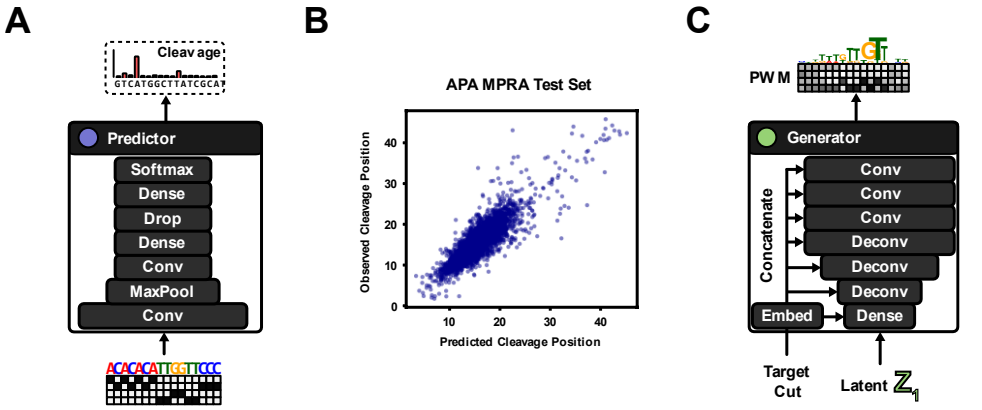
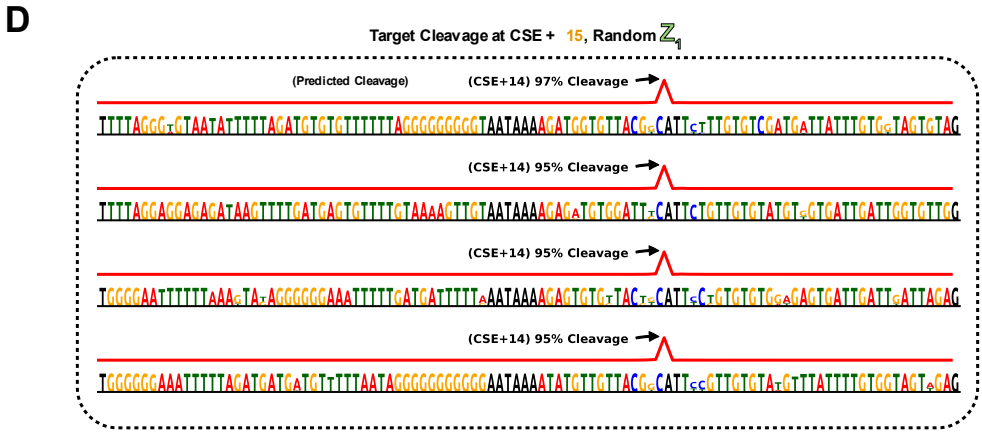

A
Similarity Margin = 0.95
Cost Coefficient = 0.05
Similarity Margin = 0.50
Cost Coefficient = 5.00
Similarity Margin = 0.30
Cost Coefficient = 5.00
Predicted Isoform Use (log)
n = 4,096
Generated Sequences
Nucleotide position
A C G T
B
Fitness Score
Edit Distance
50-th Perc.
99-th Perc.
n = 4,096 per step
Predicted Fitness Score (Isoform Log Odds)
Edit Distance / nt
Samples
Sequence Dissimilarity (1 - Similarity Margin)
C
APARENT
n = 4,096 /Algorithm
Predicted Fitness Score (Isoform Log Odds)
Edit Distance / nt
GANs
FB-GANs
Single Sample
DENs
Generative Algorithm
GAN (Unif data, 1x)
GAN (Unif data, 10x)
GAN (> .95 prox, 1x)
GAN (> .95 prox, 10x)
FB-GAN (thr .8, 50 ep)
FB-GAN (thr .8, 100 ep)
FB-GAN (thr .8, 150 ep)
FB-GAN (thr qt .5, 50 ep)
FB-GAN (thr qt .5, 100 ep)
FB-GAN (thr qt .5, 150 ep)
AM-GAN (1000 iters)
Gradient Ascent (Single)
Gradient Ascent (Multi)
Simulated Annealing
DEN (Seq. margin = .5)
DEN (Seq. margin = .4)
DEN (Seq. margin = .3)
DEN (Seq./Lat. margin = .5/.7)
DEN (Seq./Lat. margin = .3/.7)
80th Percentile n = 960 /Algorithm
50th Percentile
Sequence Samples
D
Fitness Score = Predicted Isoform Log Odds
Killoran (1000 iters)
Gradient Ascent (Single PWM)
Gradient (Multi PWM)
Simulated Annealing (1000 iters)
Fitness Score
Iterations
E
MPRA-DragoNN
n = 4,096 /Algorithm
Predicted Fitness Score (log RNA / DNA)
Edit Distance / nt
GANs
FB-GANs
Single Sample
DENs
Generative Algorithm
GAN (Unif data, 1x)
GAN (Unif data, 10x)
GAN (Top 50K, 1x)
GAN (Top 50K, 10x)
FB-GAN (thr qt .6, 50 ep)
FB-GAN (thr qt .6, 100 ep)
FB-GAN (thr qt .6, 150 ep)
FB-GAN (thr qt .8, 50 ep)
FB-GAN (thr qt .8, 100 ep)
FB-GAN (thr qt .8, 150 ep)
AM-GAN (1000 iters)
Gradient Ascent (Single)
Gradient Ascent (Multi)
Simulated Annealing
DEN (Seq. margin = .5)
DEN (Seq. margin = .3)
80th Percentile n = 960 /Algorithm
50th Percentile
Sequence Samples
F
Fitness Score = Predicted log RNA / DNA
Killoran (1000 iters)
Gradient Ascent (Single PWM)
Gradient (Multi PWM)
Simulated Annealing (1000 iters)
Fitness Score
Iterations
G
DEN
APARENT
Simulated Annealing
MPRA-DragoNN
DragoNN
ST Samples
Dashed Lines = 1x
Solid Lines = 10x
Seqs
1K
100K
10M
Predicted Fitness Score
Cumulative Sequence Samples

Figure S3. Detailed Comparison of Generative Models. Related to Figure 3. (A) Replicate analysis of Figure 2C, using the earth mover fitness cost instead of the KL-divergence cost to maximize APA isoform abundance. The Max-target APA isoform DEN was retrained three times, (1) with a low sequence diversity cost (left; cost coefficient = 0.05, allowable margin = 0.95), (2) with a medium-high diversity cost (middle; cost coefficient = 5.0, allowable margin = 0.5) and (3) with a high diversity cost (right; cost coefficient = 5.0, allowable margin = 0.3). (Top) Predicted isoform proportions for 4,096 generated sequences per generator instance. (Bottom) Sequence diversity illustrated by sampling 100 sequences and plotting them on a 2D pixel grid where rows denote sequences and columns denote nucleotide position. Low diversity penalty (left): Mean predicted isoform log odds = 6.67, 99.9% duplication rate at 100,000 sampled sequences. Medium penalty (middle): Mean predicted isoform log odds = 12.0, 0.4% duplication rate at 100,000 samples. High penalty (right): Mean predicted isoform log odds = 8.95, 0% duplication rate at 100,000 samples. **(B)** Replicate analysis of Figure 2D, using the earth mover fitness cost to maximize APA isoform abundance. The diversity cost coefficient was fixed at a high value (5.0) and the Max-target isoform DEN was retrained for different values of the allowable similarity margin (the *fraction* of nucleotides allowed to be identical without incurring cost; X-axis depicts *1 - fraction*). Plotted are the 50th and 99th percentile of predicted fitness scores (isoform log odds) and pairwise normalized edit distance for 4,096 generated sequences. **(C)** Benchmark comparison of 6 design methods for designing sequences with maximal APA isoform abundance: Generative Adversarial Networks (GAN), Feedback-GAN (FB-GAN; Gupta et al., 2019), Activation-maximization of GAN (AM-GAN; Killoran et al., 2017), PWM Gradient Ascent, Simulated Annealing and Deep Exploration Networks (DEN). Left: (Top) Predicted fitness score distribution (proximal isoform log odds) for a sample of 4,096 sequences per design method (higher is better). (Bottom) Normalized pairwise sequence edit distances (higher is better). Right: (Top) Trajectories of the 80th percentile of generated fitness scores for a sample of 960 sequences per algorithm and step. The X-axis shows the total number of sequences sampled during optimization (the sequence budget) in order to design 4,096 sequences. (Bottom) Trajectories of the 50th percentile of normalized sequence edit distances for a sample of 960 sequences per algorithm and step. Multiple configurations were tested for many of the methods. **GAN:** The GAN (Wasserstein-GAN with Gradient Penalty) was trained either on a random subsample of the APA MPRA data (“Unif data”), or on a subset of high-fitness sequences with measured proximal isoform proportions above 0.95 (“> .95 prox”). After training, we either directly sampled 4,096 sequences from the GAN for the benchmark (“1x”), or we sampled 40,960 sequences and selected the top 10% sequences with highest fitness (“10x”). **FB-GAN:** The FB-GAN was trained for either 50, 100 or 150 epochs, using a random subsample of 2,000 sequences from the APA MPRA as initial training data. The feedback threshold was either set to a fixed proportion of 0.8 throughout training (“thr .8”), or we adaptively raised the threshold to the 50th percentile of predicted isoform proportions in the newly generated dataset at the end of every epoch (“thr qt .5”). **AM-GAN:** The GAN trained on the uniform subsample of APA sequences were used for activation maximization. The latent generator seed was initialized and optimized by gradient ascent for 1,000 iterations. This procedure was repeated 4,096 times, resulting in 4,096 optimized sequences. **PWM Gradient Ascent:** A randomly initialized softmax sequence relaxation (PWM) was optimized for 50,000 iterations by gradient ascent. We either optimized just one PWM and sampled 4,096 sequences from it (“Single”), or we optimized 4,096 PWMs independently and selected the consensus sequences (“Multi”). **Simulated Annealing:** Single nucleotide substitutions were used to traverse sequence space, with the Metropolis acceptance criterion. The method started with a randomly initialized sequence and optimized

for 1,000 iterations. This procedure was repeated to optimize 4,096 sequences. **DEN:** The DEN was trained and evaluated for a range of different diversity cost configurations, using either the sequence similarity penalty with an allowable margin of 50%, 40% or 30%, or using a combination of the sequence and latent similarity penalties with allowable margins of 50% or 30% for the sequence penalty and 70% for the latent penalty. See STAR methods for all implementation details. **(D)** Selection of 100 optimization trajectories for each of the per-sequence optimization methods. Each trajectory represents a single sequence being optimized by the respective algorithm. The Y-axis displays predicted fitness scores (proximal isoform log odds) and the X-axis displays the iteration count. **(E)** Benchmark comparison of 6 design methods for designing sequences with maximal transcriptional activity as predicted by MPRA-DragoNN (Movva et al., 2019): Generative Adversarial Networks (GAN), Feedback-GAN (FB-GAN; Gupta et al., 2019), Activation-maximization of GAN (AM-GAN; Killoran et al., 2017), PWM Gradient Ascent, Simulated Annealing and Deep Exploration Networks (DEN). Left: (Top) Predicted fitness score distribution (log fold change of RNA to DNA count) for a sample of 4,096 sequences per design method (higher is better). (Bottom) Normalized pairwise sequence edit distances (higher is better). Right: (Top) Trajectories of the 80th percentile of generated fitness scores for a sample of 960 sequences per algorithm and step. The X-axis shows the total number of sequences sampled during optimization (the sequence budget) in order to design 4,096 sequences. (Bottom) Trajectories of the 50th percentile of normalized edit distances for a sample of 960 sequences per algorithm and step. Multiple configurations were tested for several of the methods. See Figure S3C for descriptions of these configurations. For this design task, we trained FB-GAN with adaptive feedback thresholds of either the 60th percentile (“thr qt .6”) or 80th percentile (“thr qt .8”) of the predicted fitness scores in the newly generated dataset at the end of every epoch. The GAN biased for high-fitness was trained on 10,000 sequences randomly sampled from the 50,000 sequences of the gene enhancer MPRA (Ernst et al., 2016) with highest measured fitnes score (“Top 50K”). **(F)** Selection of 100 optimization trajectories for each of the per-sequence optimization methods. Each trajectory represents a single sequence being optimized by the respective algorithm. The Y-axis displays predicted fitness scores (log fold change of RNA to DNA count) and the X-axis displays the iteration count. **(G)** Trajectories of the cumulative number of iterations (total sequence budget) required by DEN and Simulated Annealing to design 1,000, 100,000 and 10,000,000 sequences. Each trajectory displays the median predicted fitness score (Y-axis) for a sample of 960 sequences when the method is allowed a total sequence budget as specified by the X-axis. Two versions of the DEN were evaluated: One version where 10 one-hot patterns were sampled from each generated PWM during training to estimate the straight-through gradient (solid lines), and another version where only a single sample was used for the straight-through approximation (dashed lines). This results in a 10x lower sequence budget.


A
Cleavage
Predictor
Softmax
Dense
Drop
Dense
Conv
MaxPool
Conv
B
APA MPRA Test Set
Observed Cleavage Position
Predicted Cleavage Position
C
PWM
Generator
Concatenate
Conv
Conv
Conv
Deconv
Deconv
Deconv
Embed
Dense
Target Cut
Latent z_1


D
Target Cleavage at CSE + 15, Random z_1
(Predicted Cleavage)
(CSE+14) 97% Cleavage
(CSE+14) 95% Cleavage
(CSE+14) 95% Cleavage
(CSE+14) 95% Cleavage

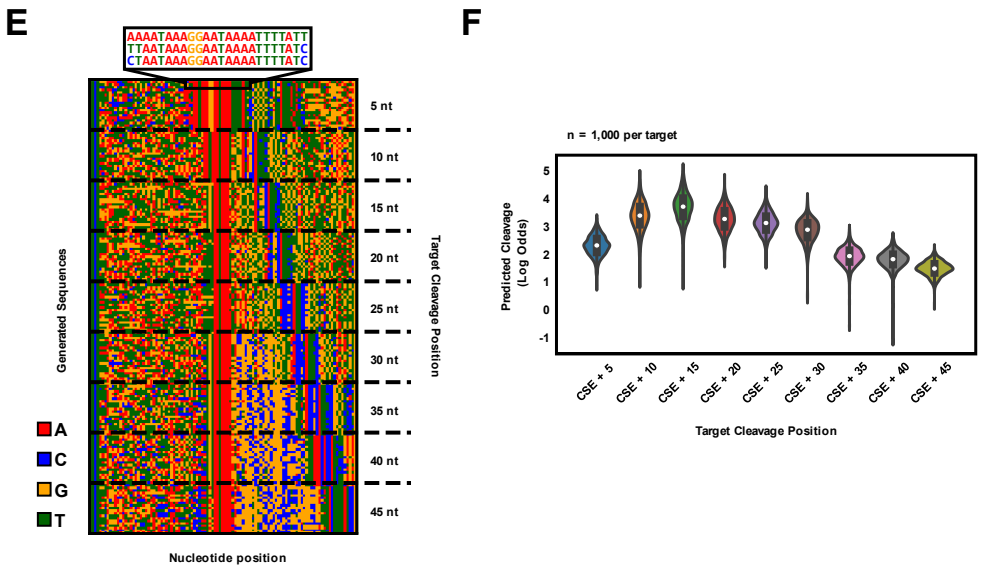

E
Generated Sequences
Target Cleavage Position
5 nt
10 nt
15 nt
20 nt
25 nt
30 nt
35 nt
40 nt
45 nt
A
C
G
T
Nucleotide position
F
n = 1,000 per target
Predicted Cleavage (Log Odds)
CSE + 5
CSE + 10
CSE + 15
CSE + 20
CSE + 25
CSE + 30
CSE + 35
CSE + 40
CSE + 45
Target Cleavage Position

Figure S4. 3' Cleavage Predictor Model and Example Sequences. Related to Figure 4. **(A)** The 3' cleavage predictor architecture. A set of convolutional layers, pooling layers, dropout layers and dense layers transform the one-hot-coded input sequence into a 3' cleavage distribution, predicting % Cleavage at nucleotide resolution. **(B)** Mean cut position prediction accuracy of APARENT. Shown are the predicted and measured mean cut positions of 5,854 test set sequences from (Bogard et al., 2019). Predicted vs. measured mean cut position $R^2 = 0.76$. **(C)** A target cut position (class index) is passed as input to the generator. The class index is transformed by a trainable embedding layer and concatenated onto every input tensor of every layer, enabling conditional generation. **(D)** Example sequences generated by the 3' cleavage DEN, using four different random seeds and the (+15 cut position)-class index as input to the generator. **(E)** Generator sequence diversity evaluated across all 9 target position classes, illustrated by randomly sampling 20 sequences per target position, and plotting them in a grid where rows denote sequences and columns denote nucleotide position. 0% duplication rate at 100,000 sampled sequences. Hexamer entropy = 9.07 of 12 bits. **(F)** Predicted cleavage log odds of 1,000 sampled sequences per target cleavage position. Mean and Standard deviation of cleavage log odds at each target cut position, in order: 2.10 (+- 0.34), 3.17 (+- 0.50), 3.54 (+- 0.47), 3.03 (+- 0.46), 2.93 (+- 0.38), 2.64 (+- 0.39), 1.73 (+- 0.32), 1.61 (+- 0.29), 1.31 (+- 0.24).

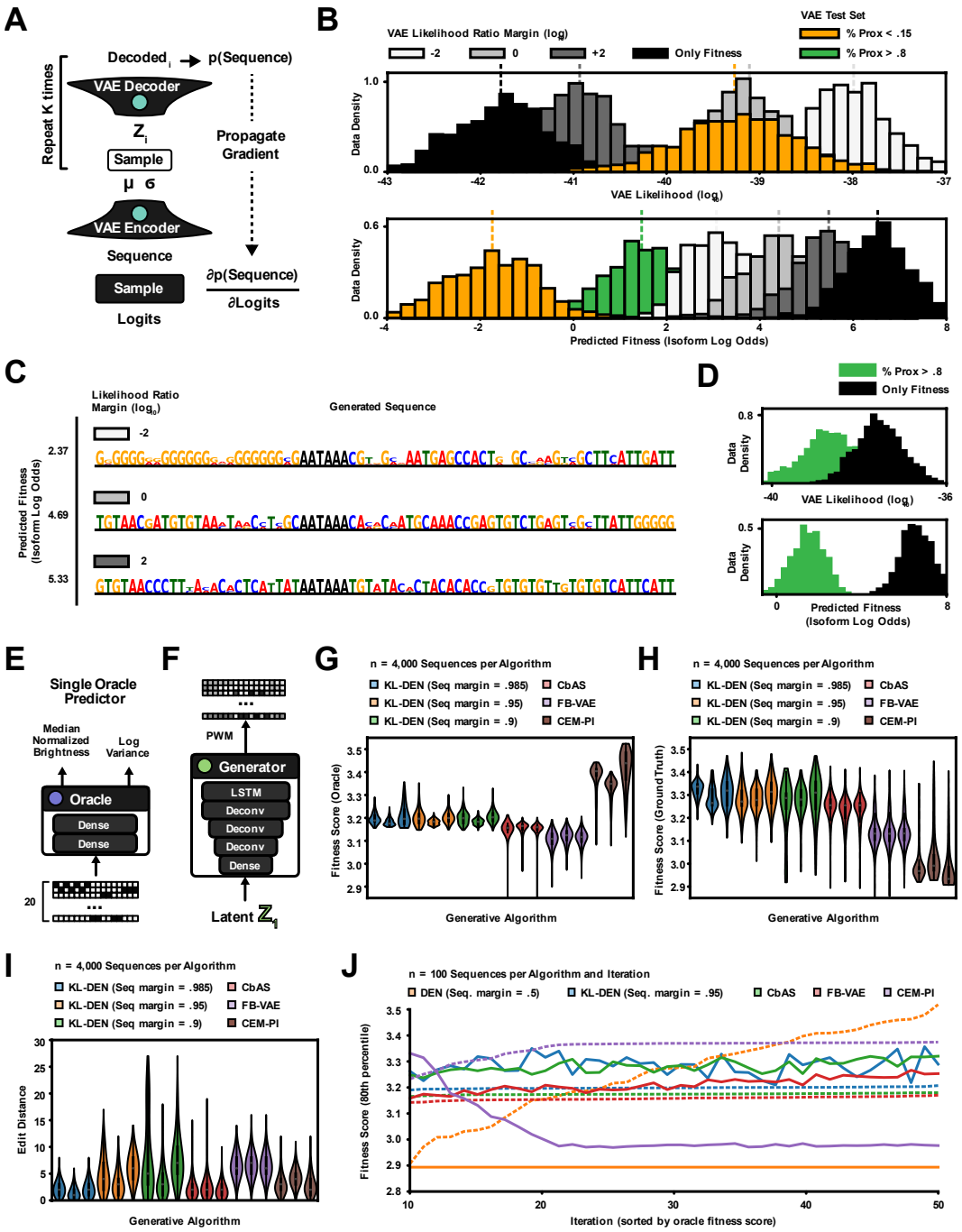

A
Decoded$_i$
p(Sequence)
VAE Decoder
Z_i
Sample
μ σ
VAE Encoder
Sequence
Sample
Logits
Repeat K times
Propagate Gradient
∂p(Sequence) / ∂Logits
B
VAE Likelihood Ratio Margin (log)
-2
0
+2
Only Fitness
VAE Test Set
% Prox < .15
% Prox > .8
Data Density
VAE Likelihood (log$_e$)
Predicted Fitness (Isoform Log Odds)
C
Likelihood Ratio Margin (log$_e$)
Generated Sequence
Predicted Fitness (Isoform Log Odds)
2.37
4.69
5.33
D
% Prox > .8
Only Fitness
Data Density
VAE Likelihood (log$_e$)
Predicted Fitness (Isoform Log Odds)
E
Single Oracle Predictor
Median Normalized Brightness
Log Variance
Oracle
Dense
Dense
20
F
PWM
Generator
LSTM
Deconv
Deconv
Deconv
Dense
Latent $\mathbb{Z}_1$
G
n = 4,000 Sequences per Algorithm
KL-DEN (Seq margin = .985)
KL-DEN (Seq margin = .95)
KL-DEN (Seq margin = .9)
CbAS
FB-VAE
CEM-PI
Fitness Score (Oracle)
Generative Algorithm
H
n = 4,000 Sequences per Algorithm
KL-DEN (Seq margin = .985)
KL-DEN (Seq margin = .95)
KL-DEN (Seq margin = .9)
CbAS
FB-VAE
CEM-PI
Fitness Score (GroundTruth)
Generative Algorithm
I
n = 4,000 Sequences per Algorithm
KL-DEN (Seq margin = .985)
KL-DEN (Seq margin = .95)
KL-DEN (Seq margin = .9)
CbAS
FB-VAE
CEM-PI
Edit Distance
Generative Algorithm
J
n = 100 Sequences per Algorithm and Iteration
DEN (Seq. margin = .5)
KL-DEN (Seq. margin = .95)
CbAS
FB-VAE
CEM-PI
Fitness Score (80th percentile)
Iteration (sorted by oracle fitness score)

Figure S5. Likelihood-bounded DENs, APA Analysis and Protein Engineering Details. Related to Figure 5. (A) The variational inference pipeline: The sequence is encoded into mean and- log-variance vectors. *K* latent vectors are sampled and decoded to estimate the VAE likelihood. Gradients are backpropagated through the sequence sample using a straight-through estimator. **(B)** The Likelihood-bounded DEN (KL-DEN) is tasked with designing polyadenylation signals which maximize the isoform proportion predicted by APARENT (See Figure 2). To show that regularization based on variational inference is tunable, we trained a VAE on 11,278 sequences from the APA MPRA (Bogard et al., 2019), selecting only weak examples with a measured proportion < 0.15. We hypothesized that the more we enforce the marginal likelihood of the VAE, the less fit the generated sequences can be, as they get pushed toward the low-fitness region of design space encoded by the VAE. We tested this by training three different KL-DENs: Each respective DEN was trained to generate maximally fit sequences that were 100-fold ($\log_{10}$ = -2; white histogram), 1-fold ($\log_{10}$ = 0; light gray histogram) and 1/100-fold ($\log_{10}$ = 2; dark gray histogram) more / less likely than the mean test set VAE likelihood (orange dashed line). 1,000 sequences were generated per likelihood ratio. Indeed, as the allowable likelihood ratio was lowered from 100 to 1 to 1/100 (white to light gray to gray), each DEN re-centered its generated sequence distribution to its least allowable likelihood due to the fitness objective (top histogram). At the same time, the predicted fitness distribution monotonically increased as the likelihood margin was lowered (bottom histogram). At the extreme, when optimizing only for fitness (no likelihood penalty), the generated sequences were nearly 1000-fold less likely than the low-fitness VAE test set sequences (orange histogram; < 0.15 isoform proportion). For reference, we also show the distribution of high-fitness VAE test sequences (green histogram; > 0.8 isoform proportion). **(C)** Example sequences for different likelihood ratio margins generated by the DENs from S6B. Shown are the sequences and their predicted fitness scores (isoform log odds). **(D)** We hypothesized that the gradual decrease in fitness observed in Figure S5B was only due to the low-fitness data distribution used for training the VAE; enforcing a large likelihood should not hurt the generated fitness distribution had the VAE been trained on high-fitness data. To test this, we re-trained the VAE on 11,055 sequences with proportions > 0.8 (green histogram). When optimizing only for fitness with no likelihood penalty (black histogram), the generated sequences (n = 1,000) became nearly 100-fold *more* likely than the test data evaluated on the high-fitness VAE. Compare to Figure S5B, where the sequences optimized only for fitness were 1000-fold *less* likely than the test data using the low-fitness VAE. **(E)** The oracle predictor architecture for the GFP design task, which is identical to the one used in (Brookes et al., 2019). Two fully connected layers with ELU activations in the hidden layer transform the 20-residue protein sequence one-hot pattern into two values: A mean normalized brightness prediction and a log-variance prediction. **(F)** The generator architecture used to train the GFP DENs. The network consists of a dense layer and three deconvolutional layers (same as previous generator architecture), followed by a recurrent LSTM layer (new for the GFP design task). **(G)** Predicted oracle fitness score distribution per generative algorithm for the GFP design task (see also Figure 5D, Left). n = 4,000 / algorithm. **(H)** Predicted "ground truth" score distributions (using the GP regression model) per generative algorithm (see also Figure 5D, Middle). n = 4,000 / algorithm. **(I)** Sequence edit distance distributions (see also Figure 5D, Right). n = 4,000 / algorithm. **(J)** The regular DEN (without VAE-likelihood penalty, similarity margin = 0.5; orange) was compared against the likelihood-bounded DEN (similarity margin = 0.95; blue), CbAS (green), FB-VAE (red) and CEM-PI (purple) on the GFP design task. Shown are the 80th percentile of oracle (dashed line) and ground truth (solid line) predicted fitness scores as a function of training epochs. The values (X-axis) are sorted on oracle scores.

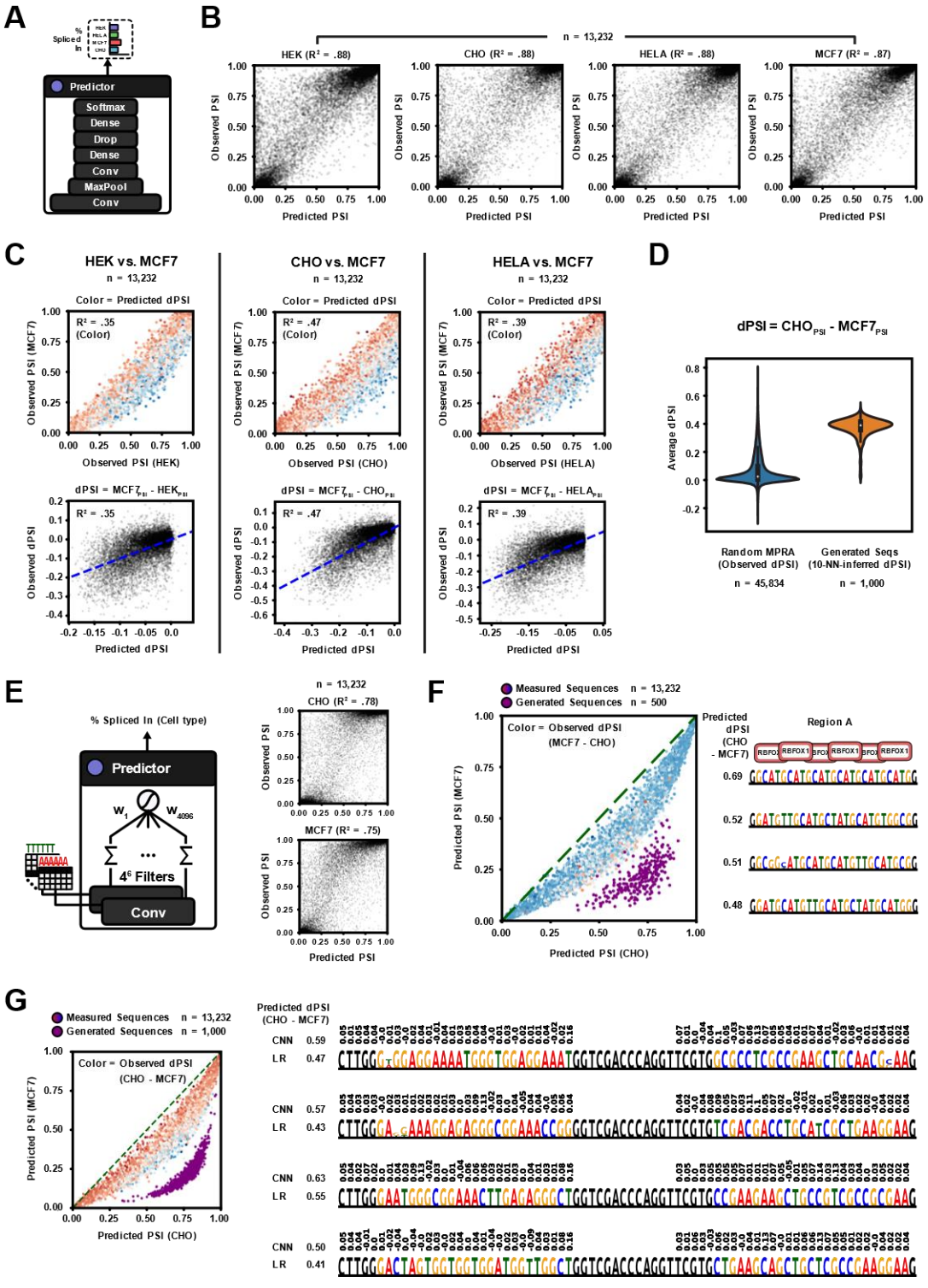

A
% Spliced In
HEK
HELA
MCF7
CHO
Predictor
Softmax
Dense
Drop
Dense
Conv
MaxPool
Conv
B
n = 13,232
HEK (R² = .88)
CHO (R² = .88)
HELA (R² = .88)
MCF7 (R² = .87)
Observed PSI
Predicted PSI
C
HEK vs. MCF7
CHO vs. MCF7
HELA vs. MCF7
n = 13,232
Color = Predicted dPSI
R² = .35 (Color)
R² = .47 (Color)
R² = .39 (Color)
Observed PSI (MCF7)
Observed PSI (HEK)
Observed PSI (CHO)
Observed PSI (HELA)
dPSI = MCF7PSI - HEKPSI
dPSI = MCF7PSI - CHOPSI
dPSI = MCF7PSI - HELAPSI
R² = .35
R² = .47
R² = .39
Observed dPSI
Predicted dPSI
D
dPSI = CHOPSI - MCF7PSI
Average dPSI
Random MPRA (Observed dPSI)
n = 45,834
Generated Seqs (10-NN-inferred dPSI)
n = 1,000
E
% Spliced In (Cell type)
Predictor
W1
W4096
4⁶ Filters
Conv
n = 13,232
CHO (R² = .78)
MCF7 (R² = .75)
Observed PSI
Predicted PSI
F
Measured Sequences n = 13,232
Generated Sequences n = 500
Color = Observed dPSI (MCF7 - CHO)
Predicted PSI (MCF7)
Predicted PSI (CHO)
Predicted dPSI (CHO - MCF7)
Region A
RBFOX1
0.69
0.52
0.51
0.48
G
Measured Sequences n = 13,232
Generated Sequences n = 1,000
Color = Observed dPSI (CHO - MCF7)
Predicted PSI (MCF7)
Predicted PSI (CHO)
Predicted dPSI (CHO - MCF7)
CNN 0.59
LR 0.47
CNN 0.57
LR 0.43
CNN 0.63
LR 0.55
CNN 0.50
LR 0.41

Figure S6. Organism-Specific Splicing Predictor Models and Sequence Generation Validation. Related to Figure 6. (A) The neural net predictor architecture. Convolutional, pooling, dropout and dense layers transform an input sequence into cell line-specific splice donor usage (PSI) predictions. **(B)** Predicted vs. measured PSI on a held-out test set of the splicing MPRA (n = 13,232). Predictions were made using a convolutional neural net with cell line-specific PSI outputs. **(C)** Top: Comparison of measured PSIs between pairs of cell lines/organisms in the splicing MPRA test set (n = 13,232). Measured PSIs are displayed on the X/Y axes, and color intensity indicates the neural network-predicted dPSIs (blue/red = more/less PSI in cell line X than Y). Bottom: Measured vs. predicted dPSIs for each cell line pair. **(D)** The max-differential splicing DEN was validated against real RNA-Seq measurements of the measured splicing MPRA using a nearest neighbor approximation. Sequences of the splicing MPRA with a minimum read count of 50 (n = 45,834) were transformed into 256-dimensional feature vectors, using the fitness predictor up until the first dense layer as a feature transform, and then stored in a nearest neighbor database. The feature transform reduces “curse of dimensionality”-effects in the nearest neighbor search and provides a degree of local motif invariance. Next, 1,000 sequences sampled from the DEN were transformed into 256-dimensional feature vectors according to the predictor and looked up in the nearest neighbor database. The measured PSIs of the 10 nearest neighbors per generated sequence were used to estimate average PSIs. Shown are the measured and NN-inferred MCF7-CHO dPSIs of the MPRA and generated sequences respectively. Mean MPRA dPSI = 0.07 (+- 0.10). Mean dPSI of generated sequences = 0.38 (+- 0.06). **(E)** Left: A differentiable relaxation of a 6-mer logistic regression model. A convolutional layer with 4096 filters, each encoding a distinct 6-mer (filter weight = 6, bias = -5), result in 4096 activation maps which after a summation over positions become hexamer counts. The counts are combined with cell line-specific weights and squashed through a sigmoid. Right: Predicted vs. measured PSI on a held-out test set of the splicing MPRA (n = 13,232). **(F)** Evaluation of a deep exploration network, using a hexamer regression predictor, when tasked with generating 500 maximally differentially spliced sequences between cell lines MCF7 and CHO. (Left) PSI as predicted by the regression model on all MPRA test set sequences in MCF7 and CHO. Color intensity indicates measured dPSI. Purple dots indicate the 500 generated sequences. (Inline) Measured vs. Predicted dPSI R^2 on test set = 0.27. (Right) Selection of maximally differentially spliced generated sequences. The generator has learned to exploit the additive independence of hexamer scores, by populating the sequences with a very differential motif, GCATGC (RBFOX1 binding site). **(G)** Both the neural network and hexamer regression splicing models were used as fitness predictors, and the DEN was retrained to maximize differential splicing between MCF7 and CHO according to the average response of both predictors. The generator was used to sample 1,000 sequences, and their average predicted PSIs (purple) were plotted alongside the predicted PSIs of the splicing MPRA test set (color intensity indicates measured dPSI). Mean predicted dPSI of test sequences = 0.08 (+- 0.07). Mean predicted dPSI of generated sequences = 0.54 (+- 0.04). (Right) The top four differentially spliced sequences generated by the DEN are shown with the neural net (CNN) and hexamer regression (LR) dPSI predictions. Shown above each sequence are the hexamer weights.

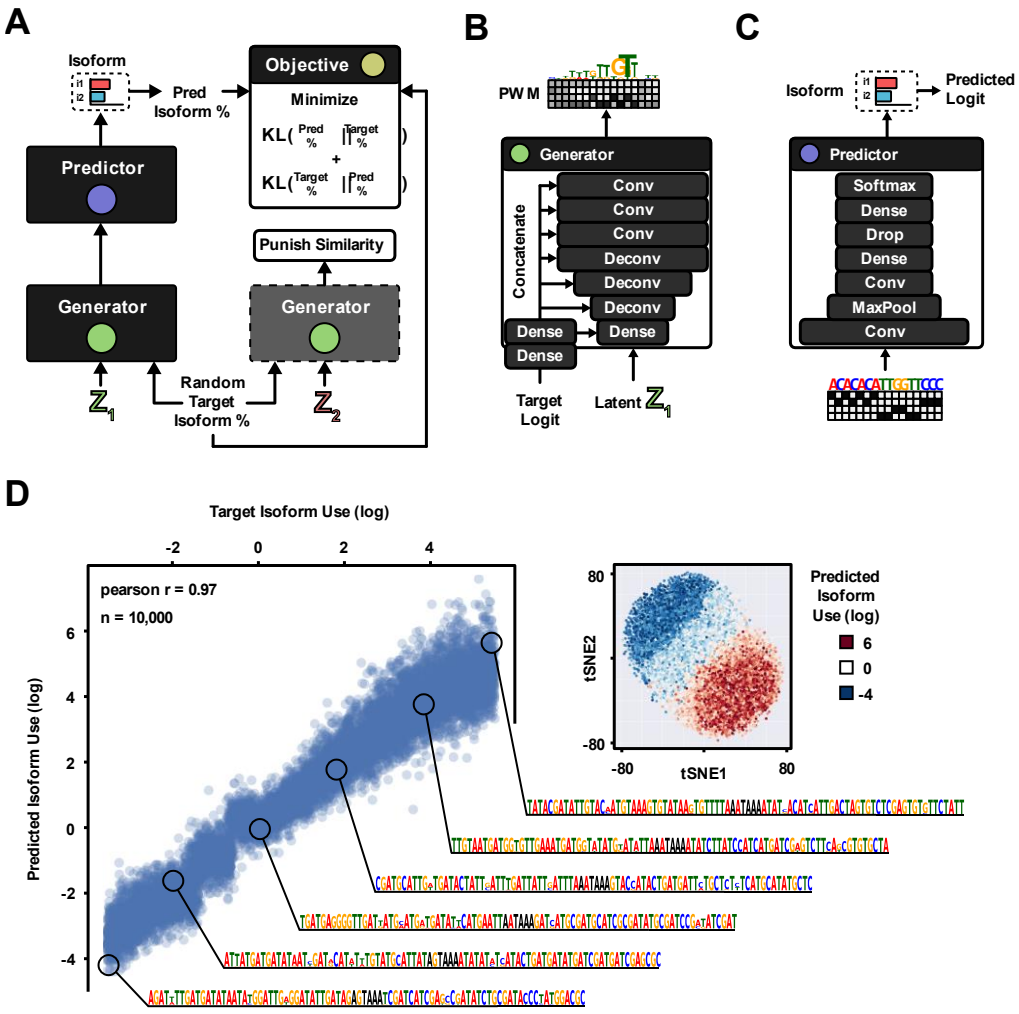


Figure S7. An Inverse Regression Model of APA. Related to STAR Methods. (A) The Inverse Regression DEN architecture: Target isoform proportions are randomly sampled and passed to both the generator and objective function. The generator is optimized to output sequences which are predicted to conform to the sampled target proportions. **(B)** A target isoform logit is passed as input to the generator in addition to the random seed. The logit is transformed into a high-dimensional, trainable embedding. The embedding is concatenated to the input tensor of every layer, enabling conditional learning and generation. See STAR methods for details on the implementation. **(C)** The APA fitness predictor architecture. A set of convolutional layers, pooling layers, dropout layers and dense layers transform the one-hot-coded input sequence into an APA isoform proportion. The predicted logit is computed and passed to the fitness cost. **(D)** The DEN was trained to generate polyadenylation signals according to randomly sampled target isoform logits in the uniform range -4 to 6, with 35% sequence similarity margin. Left: 10,000 sequences were sampled from the trained generator and their predicted isoform log odds (Y-axis) were plotted against the corresponding supplied targets (X-axis). The predicted isoform log odds were highly correlated with their targets (pearson r = 0.97), and the generator had a sequence duplication rate of 0% at 100,000 samples, indicating high diversity. Right: The sequences were projected in two-dimensional space using tSNE and colored based on their predicted isoform log odds. We find only a single tSNE cluster, where the sequences smoothly transition from one edge to the other based on the isoform log odds.

Supplemental Tables

Table S1. APA qPCR Primer Sequences. Related to STAR Methods. Table containing the forward (qPCR_FWD) and reverse (qPCR_REV_upstream, qPCR_REV_dnstream_A - qPCR_REV_dnstream_D) primer sequences used to amplify each APA reporter.

Primer ID	**Reporter**	**Sequence**
qPCR_FWD	For all reporters.	ACAACGAGGACTACACCATCGTGGAACAGTAC
qPCR_REV_upstream	For all reporters. Amplifies proximal and distal isoforms.	GGATGCGAGTAATGAATGCCATAGAAAGAGCG
qPCR_REV_dnstream_A	Amplifies the distal isoform of DEN-PAS 1 vs. Gradient Ascent-PAS 1 and DEN-PAS 2 vs. Gradient Ascent-PAS 1.	AGGCTTAATTGGCTGAAAATAAATGACAC
qPCR_REV_dnstream_B	Amplifies the distal isoform of Gradient-Ascent PAS 1 vs. DEN-PAS 1 and Gradient Ascent-PAS 2 vs. DEN-PAS 1.	CAGGCTTAATTGGCTGAATAAAAAAACACACAC
qPCR_REV_dnstream_C	Amplifies the distal isoform of DEN-PAS 1 vs. Gradient Ascent-PAS 2 and DEN-PAS 2 vs. Gradient Ascent-PAS 2.	GGCTTAATTGGCTGAAAAAATGAAGAC
qPCR_REV_dnstream_D	Amplifies the distal isoform of Gradient-Ascent PAS 1 vs. DEN-PAS 2 and Gradient-Ascent PAS 2 vs. DEN-PAS 2.	TACAGGCTTAATTGGCTGAAAAAAAGACGAC

Table S2. APARENT Predictor Architecture. Related to STAR Methods. Note: Conv(F, W, S) denotes a convolutional layer with F filters of width W, with stride S. MaxPool(P) denotes a max pooling layer with a pool size of P. Dense(N) denotes a fully connected layer with N neurons. Dropout(R) denotes a dropout layer with a drop rate of R.

Predictor	
Inputs: Sequence Input (One-hot Encoding, $\{0, 1\}^{205\times4}$)	
Conv(96, 8, 1) ReLU()	
MaxPool(2)	
Conv(128, 6, 1) ReLU()	
Dense(256) ReLU() Dropout(0.2)	
Dense(1) Sigmoid()	Dense(206) Softmax()

Table S3. VAE Decoder Architecture. Related to STAR Methods. Conv(F, W, S) denotes a convolutional layer with F filters of width W, with stride S. Dense(N) denotes a fully connected layer with N neurons. ResBlock(F, W, S) denotes a residual block with F filters of width W, with stride S.

Decoder
Inputs: Latent Seed ($\mathbb{R}^{100}$)
Dense(3072)
Reshape(8, 384)
ResBlock(384, 3, 2)
ResBlock(256, 3, 2)
ResBlock(128, 3, 2)
ResBlock(64, 3, 2)
ResBlock(32, 3, 1)
Conv(4, 1, 1)

Table S4. VAE Decoder ResBlock. Related to STAR Methods. Deconv(F, W, S) and Conv(F, W, S) denote either a deconvolutional or convolutional layer with F filters of width W, with stride S.

<table>
<tr><th colspan="2">ResBlock(F, W, S)</th></tr>
<tr><td colspan="2">Inputs:
Sequence Tensor ($\mathbb{R}^{L \times C}$)</td></tr>
<tr><td>BatchNorm()
ReLU()
Deconv(F, W, S)</td><td rowspan="2">Deconv(F, W, S)</td></tr>
<tr><td>BatchNorm()
ReLU()
Conv(F, W, 1)</td></tr>
<tr><td colspan="2">Add()</td></tr>
</table>

Table S5. VAE Encoder Architecture. Related to STAR Methods. Conv(F, W, S) denotes a convolutional layer with F filters of width W, with stride S. Dense(N) denotes a fully connected layer with N neurons. ResBlock(F, W) denotes a residual block with F filters of width W.

Encoder	
Inputs: Sequence Input (One-hot Encoding, $\{0,1\}^{205\times4}$)	
Conv(32, 1, 1)	Conv(32, 1, 1)
ResBlock(32, 8)	
ResBlock(32, 8)	
ResBlock(32, 8)	
ResBlock(32, 8)	
Conv(32, 1, 1)	
Add()	
Dense(100) (Z Mean)	Dense(100) (Z Log-var)

Table S6. VAE Encoder ResBlock. Related to STAR Methods. Conv(F, W, S) denotes a convolutional layer with F filters of width W, with stride S. The input tensor is sent directly to the addition (merge) layer along the right path of the graph.

ResBlock(F, W)	
Inputs: Sequence Tensor ($\mathbb{R}^{L\times C}$)	
BatchNorm() ReLU() Conv(F, W, 1)	No Operation (Pass through)
BatchNorm() ReLU() Conv(F, W, 1)	
Add()	

Table S7. DEN Generator Architecture. Related to STAR Methods. Deconv(F, W, S) and Conv(F, W, S) denote either a deconvolutional or convolutional layer with F filters of width W, with stride S. Dense(N) denotes a fully connected layer with N neurons. Reshape(L, C) reshapes the tensor into a 1-dimensional signal of length L with C channels.

Generator
Inputs: Latent Seed ($\mathbb{R}^{100}$)
Dense(8064)
Reshape(21, 384)
Deconv(256, 7, 2) BatchNorm() ReLU()
Deconv(192, 8, 2) BatchNorm() ReLU()
Deconv(128, 7, 2) BatchNorm() ReLU()
Conv(128, 8, 1) BatchNorm() ReLU()
Conv(64, 8, 1) BatchNorm() ReLU()
Conv(4, 8, 1)

Table S8. Inverse Regression DEN Architecture. Related to STAR Methods. Deconv(F, W, S) and Conv(F, W, S) denote either a deconvolutional or convolutional layer with F filters of width W, with stride S. Dense(N) denotes a fully connected layer with N neurons. Reshape(L, C) reshapes the tensor into a 1-dimensional signal of length L with C channels. We store the logit embedding as variable **T** and reuse it as input to every layer (**T** is concatenated to the channel dimension and replicated across spatial positions).

<table>
<tr><td colspan="2">Generator</td></tr>
<tr><td colspan="2">Inputs:</td></tr>
<tr><td>Latent Seed ($\mathbb{R}^{100}$)</td><td>Target Logit ($\mathbb{R}$)</td></tr>
<tr><td rowspan="2">No Operation (Pass through)</td><td>Dense(256)
ReLU()</td></tr>
<tr><td>Dense(100)
Tanh()
(Store as Tensor Variable T)</td></tr>
<tr><td colspan="2">Concatenate()</td></tr>
<tr><td colspan="2">Dense(8064)</td></tr>
<tr><td>Reshape(21, 384)</td><td>Reuse Tensor Variable T
Broadcast to Compatible Shape</td></tr>
<tr><td colspan="2">Concatenate()</td></tr>
<tr><td>Deconv(256, 7, 2)
BatchNorm()
ReLU()</td><td>Reuse Tensor Variable T
Broadcast to Compatible Shape</td></tr>
<tr><td colspan="2">Concatenate()</td></tr>
<tr><td>Deconv(192, 8, 2)
BatchNorm()
ReLU()</td><td>Reuse Tensor Variable T
Broadcast to Compatible Shape</td></tr>
<tr><td colspan="2">Concatenate()</td></tr>
<tr><td>Deconv(128, 7, 2)
BatchNorm()
ReLU()</td><td>Reuse Tensor Variable T
Broadcast to Compatible Shape</td></tr>
<tr><td colspan="2">Concatenate()</td></tr>
<tr><td>Conv(128, 8, 1)
BatchNorm()
ReLU()</td><td>Reuse Tensor Variable T
Broadcast to Compatible Shape</td></tr>
<tr><td colspan="2">Concatenate()</td></tr>
<tr><td>Conv(64, 8, 1)</td><td>Reuse Tensor Variable T</td></tr>
</table>

BatchNorm() ReLU()	Broadcast to Compatible Shape
Concatenate()	
Conv(4, 8, 1)	

Table S9. Class-conditional DEN Architecture. Related to STAR Methods. Deconv(F, W, S) and Conv(F, W, S) denote either a deconvolutional or convolutional layer with F filters of width W, with stride S. Dense(N) denotes a fully connected layer with N neurons. Reshape(L, C) reshapes the tensor into a 1-dimensional signal of length L with C channels. We store the class embedding as variable **C** and reuse it as input to every layer (**C** is concatenated to the channel dimension and replicated across spatial positions).

Generator	
Inputs:	
Latent Seed ($\mathbb{R}^{100}$)	Target Cut Position (1-Hot Encoding of C classes, $\{0,1\}^C$, Store as Tensor Variable **C**)
Concatenate()	
Dense(8064)	
Reshape(21, 384)	Reuse Tensor Variable **C** Broadcast to Compatible Shape
Concatenate()	
Deconv(256, 7, 2) BatchNorm() ReLU()	Reuse Tensor Variable **C** Broadcast to Compatible Shape
Concatenate()	
Deconv(192, 8, 2) BatchNorm() ReLU()	Reuse Tensor Variable **C** Broadcast to Compatible Shape
Concatenate()	
Deconv(128, 7, 2) BatchNorm() ReLU()	Reuse Tensor Variable **C** Broadcast to Compatible Shape
Concatenate()	
Conv(128, 8, 1) BatchNorm() ReLU()	Reuse Tensor Variable **C** Broadcast to Compatible Shape
Concatenate()	
Conv(64, 8, 1) BatchNorm() ReLU()	Reuse Tensor Variable **C** Broadcast to Compatible Shape
Concatenate()	
Conv(4, 8, 1)	

Table S10. Protein Design DEN Architecture. Related to STAR Methods. Deconv(F, W, S) and Conv(F, W, S) denote either a deconvolutional or convolutional layer with F filters of width W, with stride S. Dense(N) denotes a fully connected layer with N neurons. LSTM(N) denotes an LSTM layer with N Units. Reshape(L, C) reshapes the tensor into a 1-dimensional signal of length L with C channels.

Generator
Inputs: Latent Seed ($\mathbb{R}^{100}$)
Dense(12288)
Reshape(32, 384)
Deconv(256, 8, 2) BatchNorm() ReLU()
Deconv(192, 8, 2) BatchNorm() ReLU()
Deconv(128, 8, 2) BatchNorm() ReLU()
LSTM(20)
BatchNorm() Conv(20, 1, 1)

Table S11. Splicing CNN Predictor Architecture. Related to STAR Methods. Conv(F, W, S) denotes a convolutional layer with F filters of width W, with stride S. MaxPool(P) denotes a max pooling layer with a pool size of P. Dense(N) denotes a fully connected layer with N neurons. Dropout(R) denotes a dropout layer with a drop rate of R.

Predictor	
Inputs:	
Sequence Input A (1-Hot Encoding, $\{0,1\}^{35\times4}$)	Sequence Input B (1-Hot Encoding, $\{0,1\}^{35\times4}$)
Conv(96, 8, 1) ReLU()	Conv(96, 8, 1) ReLU()
MaxPool(2)	MaxPool(2)
Conv(128, 6, 1) ReLU()	Conv(128, 6, 1) ReLU()
Concatenate()	
Dense(256) ReLU() Dropout(0.2)	
Dense(4) Sigmoid()	

Table S12. Splicing Hexamer Regression Predictor Architecture. Related to STAR Methods. Conv(F, W, S) denotes a convolutional layer with F filters of width W, with stride S. Dense(N) denotes a fully connected layer with N neurons.

Predictor	
Inputs:	
Sequence Input A (1-Hot Encoding, $\{0,1\}^{35\times4}$)	Sequence Input B (1-Hot Encoding, $\{0,1\}^{35\times4}$)
Conv(4096, 6, 1) ReLU() Sum() (Across positions)	Conv(4096, 6, 1) ReLU() Sum() (Across positions)
Concatenate()	
Dense(4) Sigmoid()	