Designing DNA With Tunable Regulatory Activity Using Discrete Diffusion

Anirban Sarkar[*]
Simons Center for Quantitative Biology
Cold Spring Harbor Laboratory

Ziqi Tang[†]
Simons Center for Quantitative Biology
Cold Spring Harbor Laboratory

Chris Zhao
School of Biological Sciences
Cold Spring Harbor Laboratory

Peter K Koo[*]
Simons Center for Quantitative Biology
Cold Spring Harbor Laboratory

Abstract

Engineering regulatory DNA sequences with precise activity levels in specific cell types hold immense potential for medicine and biotechnology. However, the vast combinatorial space of possible sequences and the complex regulatory grammars governing gene regulation have proven challenging for existing approaches. Supervised deep learning models that score sequences proposed by local search algorithms ignore the global structure of functional sequence space. While diffusion-based generative models have shown promise in learning these distributions, their application to regulatory DNA has been limited. Evaluating the quality of generated sequences also remains challenging due to a lack of a unified framework that characterizes key properties of regulatory DNA. Here we introduce DNA Discrete Diffusion (D3), a score entropy discrete diffusion model for DNA sequences, to conditionally generate regulatory sequences with targeted functional activity levels. We develop a comprehensive suite of evaluation metrics that assess the functional similarity, sequence similarity, and regulatory composition of generated sequences. Through benchmarking on three high-quality functional genomics datasets spanning human promoters and fly enhancers, we demonstrate that D3 outperforms existing methods in capturing the diversity of cis-regulatory grammars and generating sequences that more accurately reflect the properties of genomic regulatory DNA. Furthermore, we show that D3-generated sequences can effectively augment supervised models and improve their predictive performance, even in data-limited scenarios.

1 Introduction

Designing DNA sequences that precisely control gene expression in specific cell types is a fundamental challenge with profound implications for a wide range of fields, from synthetic biology to gene therapy. However, the immense complexity of gene regulation and the vast space of possible DNA sequences have made this goal elusive. Traditionally, deep neural networks (DNNs) have been used to learn a mapping from DNA sequences to experimentally measured gene expression or other functional genomics readouts [1, 2, 3, 4, 5]. These DNNs serve as an oracle (i.e., scoring function) to predict the functional activity of proposed sequences [6, 7, 8]. This reframes the design task as an optimization along sequence space through the lens of the DNN. Recently proposed methods include gradient ascent [9] or in silico evolution [6, 7]. While somewhat successful, these approaches are limited to exploring local optima and struggle to broadly sample the diversity of functional sequences.

[*]Send correspondence to: asarkar@cshl.edu & koo@cshl.edu
[†]Currently research scientist at InstaDeep

Recent breakthroughs in generative modeling, particularly diffusion [10, 11, 12] and flow matching [13], offer an alternative approach by learning the underlying data distribution and enabling conditional generation based on desired attributes. These models have been introduced for DNA engineering recently, showing early promise. These include the Dirichlet Diffusion Score Model (DDSM) [14], DiscDiff [15], DNA-Diffusion [16] and DNA Flow Matching (DFM) [17]. Here we introduce DNA Discrete Diffusion (D3) which is adapted from Score Entropy Discrete Diffusion (SEDD) [18] to regulatory DNA sequences. SEDD was recently introduced as a natural language generative model that builds on diffusion processes by progressively adding noise to sentences and learning to reverse it, reconstructing coherent text. Unlike traditional autoregressive models, which predict one word at a time, SEDD diffuses noise across the entire sequence, providing a different approach for handling discrete data like words.

While conditional generation is supported by all these models for DNA sequence design, they differ in how they handle the generation process. D3 operates directly in the discrete nucleotide space, applying conditions to nucleotide transitions based on desired functional activity. In contrast, DDSM (which operates in the probability simplex), DiscDiff (latent space), DNA Diffusion (Euclidean space), and DFM (vector field) apply conditions in continuous spaces and require a post-processing step—quantization or rounding—to convert their outputs back into discrete sequences. D3's discrete approach is particularly effective for generating regulatory sequences, where sparse, informative motifs are surrounded by large stretches of non-informative "junk" DNA. By directly focusing on nucleotide transitions, D3 can refine key functional motifs without being overwhelmed by the surrounding noise. On the other hand, models like DiscDiff and DFM must reconstruct sequences from compressed latent spaces or continuous vector fields, which can lead to distortions during reconstruction. By contrast, we surmise that issue of class imbalance is mitigated through D3's selective mutations of nucleotides, much like how evolution fine-tunes functional properties of DNA through mutations at key positions while tolerating neutral drift in others.

All these existing diffusion based models for DNA [14, 15, 19, 16, 17] could allow directly generating regulatory sequences with specified activity levels and cell-type specificity. However, it is unclear whether they can reliably generate functional sequences that faithfully recreate the cell-type-specific grammars of cis-regulatory mechanisms inherent to genomic DNA. Here, we develop a comprehensive set of evaluations to assess whether generated sequences encode valid regulatory logic found in real genomes. By benchmarking DNA Discrete Diffusion (D3) against previous methods, we demonstrate that D3 generates DNA sequences that better capture the diversity of cis-regulatory grammars.

Our major contributions include:

- A new set of evaluations for designed regulatory sequences.
- Application of Score Entropy Discrete Diffusion to regulatory DNA sequences.
- D3 improves conditional sequence generation compared to previous DNA diffusion and flow matching models across three datasets, two of which are new to generative modeling.

2 Background

2.1 Biological Motivation

The human genome consists of 3 billion base pairs across 23 chromosome pairs [20]. While each cell shares the same DNA sequence, differential gene regulation gives rise to various cell types throughout the human body. Protein-coding genes comprise only 1-2% of the genome, while non-coding regions, which include promoters and enhancers, regulate transcription [21]. Promoters initiate transcription by recruiting RNA polymerase via transcription factors (TFs), proteins that bind to specific DNA sequence elements that are 4-19 nucleotides (nt) long. TF binding sites are referred to as motifs [22]. Enhancers are TF binding hotspots and interact with promoters to enhance transcription [23].

Experimental approaches like genome-wide chromatin profiling assays (e.g., CAGE-seq [24]) and massively parallel reporter assays (MPRAs [25]) have advanced quantitative measurements of regulatory sequence effects on gene expression. Supervised DNNs trained on these datasets have provided some insights [26, 3, 27, 28, 29, 30, 31, 32]. However, understanding the complex cis-regulatory code (i.e., TF binding rules, TF-TF interactions, and sequence context dependencies) remains an open challenge.

Current approaches for designing regulatory DNA involve using trained supervised DNNs to score sequences from local search strategies [6, 7, 33, 9]. Generative models, such as GANs [34, 35, 36], VAEs [37], and normalizing flows [38], among others [39], provide an avenue to sample sequence (design) space more broadly, but have yet to demonstrate reliable regulatory DNA generation. Similarly, BERT-style [40, 41, 42, 43, 44, 45] and autoregressive DNA language models [46, 47, 48, 49] have yet to demonstrate a comprehensive understanding of human regulatory sequences [50]. See Appendix A for a more detailed primer.

In contrast, deep generative models and language models have been successful when analyzing protein sequences or coding regions [51, 52, 53, 54]. The discrepancy with non-coding regulatory DNA may lie in the long stretches of nucleotides where only some positions are informative for TF binding. This is analogous to injecting random words between every meaningful word in a sentence and expecting a language model to learn the natural language structure within the noise. The sparsity of informative positions creates a class imbalance between informative and uninformative nucleotides, limiting the effectiveness of loss functions that aim to predict missing tokens or reconstruct input sequences, as these objectives can only be achieved through memorization rather than learning meaningful patterns.

2.2 Continuous Time Framework for Discrete Diffusion

Let $x_0 \in \mathcal{X}$ be a discrete sequence data with finite cardinality $\mathcal{S} = |\mathcal{X}|$ and $x_0 \sim q_{data}(x_0)$ for some discrete data distribution $q_{data}(x_0)$. In the continuous time framework, the discrete diffusion process can be described by a linear ODE [55, 56]: $\frac{dq_t}{dt} = \mathcal{R}_t q_t$, where q_t represents the family of distributions that evolves with time t, starting with $q_0 \approx q_{data}$, and $\mathcal{R}_t$ is the transition rate matrix. The reverse process starts from the marginal $q_T(x_T)$ and can lead to the data distribution $q_{data}(x_0)$ with a suitable transition rate matrix $\bar{\mathcal{R}}_t \in \mathbb{R}^{S\times S}$, which should have non-negative non-diagonal entries and columns summed to zero, restricting q_t to gain or lose any mass during the process. $\mathcal{R}_t$ can be analytically chosen as $\mathcal{R}_t = \beta(t)\mathcal{R}_b$ with a time-dependent scalar factor β_t and time independent base rate matrix $\mathcal{R}_b$ such that q_t approaches a stationary distribution p_{ref} as $t \to \infty$. With this formulation, the reverse transition matrix satisfies (for a perturbed $\bar{x}$):

$$\bar{\mathcal{R}}_t(x, \bar{x}) = \frac{q_t(\bar{x})}{q_t(x)} \mathcal{R}_t(\bar{x}, x) \tag{1}$$

and the reverse process ODE is given by[57, 58]:

$$\frac{dq_{T-t}}{dt} = \bar{\mathcal{R}}_{T-t} q_{T-t}. \tag{2}$$

The ratios $\frac{q_t(\bar{x})}{q_t(x)}$ are collectively known as the concrete score and are generally intractable. Previous methods attempted to learn the density ratios $\frac{q_t(\bar{x})}{q_t(x)}$ to construct the reverse diffusion process. Lou et al. [18] proposed a score matching network with Bregman divergence that restricts the density ratios to positive values and performs well for discrete data. We employ a similar method with a different function for Bregman divergence, leading to the same formulation as Lou et al. [18]. Hence, as explained in the next section, we adapt similar forward and backward transition ideas for genomic sequences, which are inherently different from text and require specific modifications. See Appendix B for additional details on the continuous time framework and C for previous approaches to learn density ratios.

3 Use of Score Entropy for Discrete Diffusion

Concrete score matching generalizes the standard Fisher divergence to learn a neural network $s_\theta(x, t)$ that can approximate the density ratios $\frac{q_t(\bar{x})}{q_t(x)}$ but does not perform well due to the inability of l_2 loss to penalize the non-positive values. On the other hand, score entropy is defined as Bregman divergence $D_F(s_\theta(x), \frac{q(\bar{x})}{q(x)})$ that enjoys convexity and non-negativity properties in general. Specifically, we consider divergence associated with $F(p)$ as $\sum p \log p$, i.e., the negative entropy function that inspired the name. Score entropy loss ($\mathcal{L}_{SE}$) is defined as:

$$\mathcal{L}_{SE} = \mathbb{E}_{x\sim q}\left[\sum_{\bar{x}\neq x} w_{x\bar{x}} \left\{ s_\theta(x)_{\bar{x}} - \frac{q(\bar{x})}{q(x)} \log s_\theta(x)_{\bar{x}} + K\left(\frac{q(\bar{x})}{q(x)}\right)\right\}\right] , \tag{3}$$

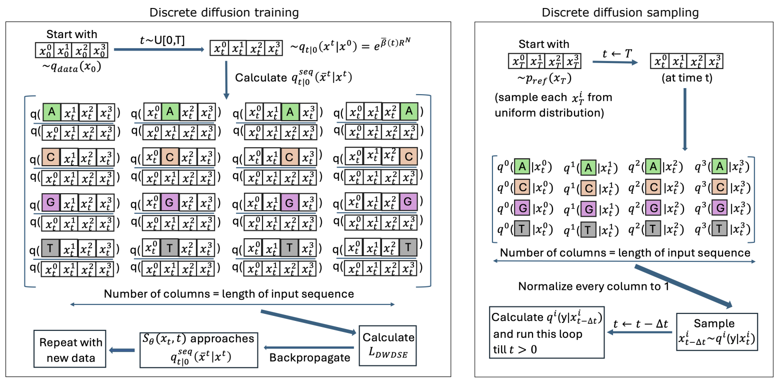
Discrete diffusion training
Start with
$\sim q_{data}(x_0)$
$t \sim U[0,T]$
$\sim q_{t|0}(x^t|x^0) = e^{\bar{\beta}(t)R^N}$
Calculate $q^{seq}_{t|0}(\bar{x}^t|x^t)$
Number of columns = length of input sequence
Repeat with new data
$S_\theta(x_t,t)$ approaches $q^{seq}_{t|0}(\bar{x}^t|x^t)$
Backpropagate
Calculate L_{DWDSE}
Discrete diffusion sampling
Start with
$\sim p_{ref}(x_T)$
(sample each x^i_T from uniform distribution)
$t \leftarrow T$
(at time t)
Number of columns = length of input sequence
Normalize every column to 1
Calculate $q^i(y|x^i_{t-\Delta t})$ and run this loop till $t > 0$
$t \leftarrow t - \Delta t$
Sample $x^i_{t-\Delta t} \sim q^i(y|x^i_t)$

Figure 1: Overview of D3 training and sampling. (*left*) Perturb the original sequence $x_0 \sim q_{data}$ (of length 4) to x_t (based on a uniformly sampled time t) following Eq. 6 and Eq. 7. Then, calculate density ratios $\frac{q(\bar{x}|x_0)}{q(x|x_0)}$ for sequences with 1-Hamming distance from x_t to calculate the score entropy loss. (*right*) Start with a random initial sequence, then calculate transition densities (Eq. 9), generating intermediate samples at time $t - \Delta t$, until $t > 0$; a data sample is generated at the last step.

where $K(l) = l(\log l - 1)$. The weights $w_{x\bar{x}}$ take a specific form using score entropy with diffusion models. However, $\mathcal{L}_{SE}$ contains unknown $\frac{q_t(\bar{x})}{q_t(x)}$. The ratio is replaced with $\frac{q(\bar{x}|x_0)}{q(x|x_0)}$ in the denoising score entropy loss $\mathcal{L}_{DSE}$, where q is a perturbation of a base density q_0 by a transition kernel $q(.|.)$ – see theorem 3.4 in [18] for a proof of equivalence. Please see Appendix B for how evidence lower bound (ELBO) is defined for likelihood-based training with a time-dependent score network $s_\theta(., t)$ [18]. Below, we discuss choices for the forward transition kernel that allow easy calculation of the score entropy loss.

3.1 Choice of Forward Process (Specific to Genomics Data)

Modeling highly complex genomic sequences of the form $\mathbf{x} = x^1...x^d$ with a high value of d through score entropy incurs scaling challenges. Our state factorizes into sequences $\mathcal{X} = \{A, C, G, T\}^d$, where each position of the sequence represents one nucleotide, i.e., $x^i \in \{A, C, G, T\}$. We consider a sparse structured matrix $\mathcal{R}^N_t$ that can perturb every nucleotide independently, which is given by:

$$\mathcal{R}_t(x^1...x^i...x^d, x^1...\bar{x}^i...x^d) = \mathcal{R}^N_t(x^i, \bar{x}^i) \ . \tag{4}$$

We model all the density ratios for sequences with Hamming distance 1 following Lou et al. [18], which drastically reduces the span of perturbations to only sequences with one nucleotide change. This leads to the score network modeling a sequence-to-sequence map with $\mathcal{O}(4d)$ (reduced from $\mathcal{O}(4^d)$ for any sequence to any sequence transition) time as below:

$$(s_\theta(x^1...x^i...x^d, t))_{i,x\to\bar{x}} \approx \frac{q_t(x^1...\bar{x}^i...x^d)}{q(x^1...x^i...x^d)} \ . \tag{5}$$

The calculation for the forward transition $q^{seq}_{t|0}(.|.)$ with only one nucleotide change can be done in a single step by decomposing it into independent token perturbations:

$$q^{seq}_{t|0}(.|x) = \prod_{i=1}^{d} q^N_{t|0}(\bar{x}^i, x^i) \ . \tag{6}$$

Each $q^N_{t|0}(\bar{x}^i, x^i)$ can be calculated with a noise level β and a pre-specified transition matrix $\mathcal{R}^N$ as $\mathcal{R}^N_t = \beta(t)\mathcal{R}^N$. Considering $\bar{\beta}_t$ as the cumulative noise $\int_0^t \beta(t)dt$, we have $q^N_{t|0}(.|x) = x$-th column of $\exp(\bar{\beta}(t)\mathcal{R}^N)$. More specifically that implies

$$q^N_{t|0}(\bar{x}_t = j|x_0 = i) = \exp(\bar{\beta}(t)\mathcal{R}^N)(j, i) \ , \tag{7}$$

where (j, i) represents jth row, ith column entry of the matrix $\exp(\bar{\beta}(t)\mathcal{R}^N)$.

Here we consider special structures for the transition matrix, which removes the requirement of storing all possible edge weights $\mathcal{R}^N(i, j)$ [55, 59], resulting in a much lower memory and time requirement. Specifically, a uniform stationary base rate matrix $\mathcal{R} = \mathbb{1}\mathbb{1}^\top - \mathcal{S}\mathcal{I}$ can be a natural choice for categorical discrete distributions, where $\mathbb{1}\mathbb{1}^\top$ is a matrix of all ones, $\mathcal{I}$ is identity matrix. With this formulation, we can quickly calculate all values of $\mathcal{L}_{DWDSE}$ for model training. See Appendix B for additional details of the stationary matrix $\mathcal{R}$ and time-dependent version $\mathcal{R}_t^N$.

This formulation allows D3 to accept a conditioning signal, a scalar or vector, alongside the data as input to the score network, which learns the density ratios based on the provided information. This approach differs from classifier-free guidance as the conditioning signals consist of continuous-valued scores, aligning precisely with the formulation for conditional training. Previous methods [14, 17] employed this type of conditioning for designing human promoters (see Sec. 5).

3.2 Simulating Generative Reverse Process with Concrete Scores

Optimal denoising can be achieved only by knowing all density ratios $\frac{q_t(\bar{x})}{q_t(x)}$. If q_t follows the diffusion ODE (i.e., $\frac{dq_t}{dt} = \mathcal{R}_t q_t$), the true denoiser can be represented as a discrete version of Tweedie's theorem [60], as shown by Lou et al. [18]:

$$q_{0|t}(x_0|x_t) = \left(\exp(-t\mathcal{R}) \left[\frac{q_t(i)}{q_t(x_t)} \right]_{i=1}^{4} \right)_{x_0} \exp\left(t\mathcal{R}(x_t, x_0)\right) \ . \tag{8}$$

The generative reverse process can be efficiently simulated using tau-leaping and Tweedie denoiser analog, which leverage the concrete scores and consider only sequences with a 1-Hamming distance as follows:

$$q^i(x^i_{t-\Delta t}|x^i_t) = \left(\exp(-\beta(t)^{\Delta t}\mathcal{R}) s_\theta(x_t, t)_i\right)_{x^i_{t-\Delta t}} \exp(-\beta(t)^{\Delta t}\mathcal{R}(x^i_t, x^i_{t-\Delta t}) \ , \tag{9}$$

where $\beta(t)^{\Delta t} = (\bar{\beta}(t) - \bar{\beta}(t - \Delta t))$. Now $q^i(x^i_{t-\Delta t}|x^i_t)$ may need clamping for minimum 0 and normalizing the sum to 1, and $x_{t-\Delta t}$ can be obtained by sampling i-th token as $x^i_{t-\Delta t} \sim q^i(x^i_{t-\Delta t})$. See Appendix B for additional details on the general reverse process. The sampling process accepts conditioning signals with the uniformly sampled sequences and follows a similar formulation through computing the ratios incorporated in the conditionally trained score network outputs. Figure 1 shows an overview of the D3 training and sampling process.

4 Evaluation Framework for Regulatory DNA Sequences

We have incorporated standard practices and introduced new ones to assess the quality of generated sequences. Our evaluations are grouped into three main categories (Fig. 2): 1) functional similarity, 2) sequence similarity, and 3) compositional similarity. Functional similarity assesses whether generated sequences exhibit similar functional activity as real DNA from the original dataset. While experimental validation is ideal, we rely on an oracle—a DNN trained to predict functional genomics data—to provide "in silico measurements". Sequence similarity assesses the extent of memorization of training sequences and whether generated sequences exhibit similar properties to real ones. Compositional similarity assesses the similarity of motif composition between generated and real sequences, as regulatory function is primarily driven by motifs and their complex interactions.

4.1 Functional Similarity

Conditional generation fidelity measures how well-generated sequences yield the same functional activity as the conditioning value. An oracle predicts activities for a matched set of natural and generated sequences, and mean squared error is calculated [14, 17]. Lower values indicate higher fidelity.

Frechet distance is the 2-Wasserstein distance between embeddings given by an oracle for synthetic and real data, assuming a multivariate normal distribution. It assesses the quality and variety of generated sequences compared to real ones. Lower values indicate greater sample variety.

Predictive distribution shift is the Kolmogorov-Smirnoff test statistic, quantifying the largest deviation between the cumulative distribution functions of functional activity predictions for generated and natural sequences. Lower values indicate more similar distributions.

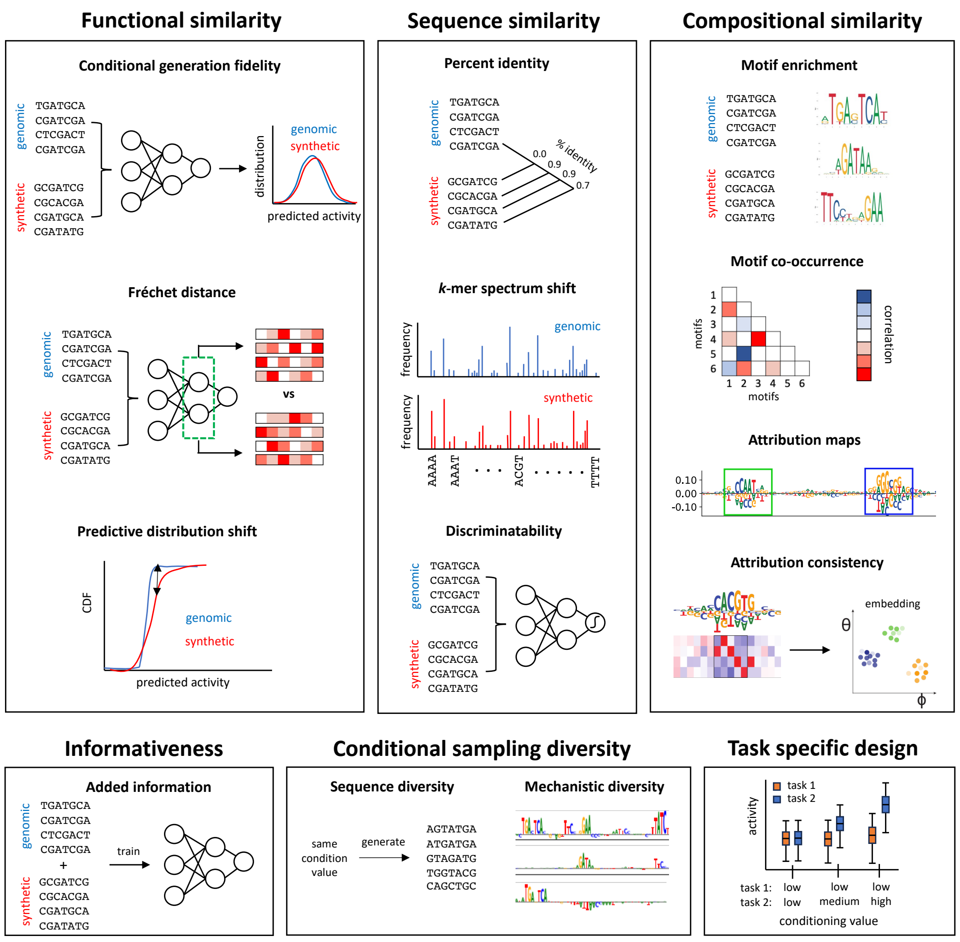

Figure 2: Schematic of evaluation framework.

4.2 Sequence Similarity

Percent identity measures the extent of memorization between two sequences using Hamming distance normalized to sequence length. A value of 1 indicates perfect memorization; lower values indicate that the model is generating unseen sequences. We calculate the maximum percent identity for each generated sequence against all training sequences (memorization) and among generated sequences (diversity). We summarize the percent identity distribution as an average maximum percent identity and a global maximum across all generated sequences.

k**-mer spectrum shift** measures the similarity of low-level sequence statistics between generated and natural sequences using the Jensen-Shannon distance between k-mer frequencies ($k = 1$ to $k = 7$). Lower values indicate more similar distributions.

Discriminatability tests how well a discriminator can classify generated sequences from real ones using the area under the receiver-operating characteristic curve (AUROC). A value of 0.5 suggests similar sequence properties; 1 indicates different properties.

4.3 Compositional Similarity

Motif enrichment is calculated by scanning sequences with position weight matrices [61] from the JASPAR database [62] using FIMO [63]. We calculate the total number of motif hits (879 unique motifs) across real and generated sequences and use Pearson's correlation to measure similarity. A correlation closer to 1 indicates similar proportions of motifs (up to a scaling factor).

Motif co-occurrence is assessed by calculating the motif-motif covariance matrix based on motif counts per sequence separately for real and generated sequences given by FIMO. The Frobenius norm between the two covariance matrices is calculated; lower values indicate higher similarity between motif co-occurrences. This provides a coarse-grained view of whether the generated sequences are learning co-occurring motifs, a necessary but not sufficient property of cis-regulatory mechanisms.

Attribution analysis scores the importance of single-nucleotide variants on model predictions, revealing important motif positions. Using an oracle, we employ `GradientSHAP` from Captum [64] with 5,000 random sequences as references. We perform gradient correction to each attribution map [65] and visualize attribution maps using Logomaker [66]. We calculated attribution maps for 2,000 generated and real sequences based on high functional activity and compared them qualitatively.

Attribution consistency quantifies the enrichment of k-mer patterns within high attribution scores using the Kullback-Leibler divergence (KLD) between an analytical, geometric embedding of mean attribution scores over $k = 6$ positions and an uninformative prior [67]. Higher values indicate more consistent patterns. We analyze 2,000 sequences with the highest functional activity. We also assess attribution consistency across generated and real sequences by analyzing a concatenated sequence set.

4.4 Other Evaluations

Informativeness refers to the **added information** provided by the generated sequences when combined with genomic sequences. We conditionally generate sequences with matched activities as the original training sequences, combine them with downsampled subsets of the original training data (100%, 75%, 50%, 25%, and 0%), and train a DNN on the hybrid dataset using the same procedure outlined in the original studies. If the generated sequences provide added information, the performance will improve compared to training on just the original training sequences.

Conditional sampling diversity is assessed according to the **sequence diversity**, which calculates percent identity on generated sequences sampled using the same conditioning value. For the fly enhancer dataset, we generate 500 samples using a fixed conditioning value and perform an all-vs-all percent identity comparison. A baseline is set using a matched set of dinucleotide shuffled sequences. We also assess the **mechanistic diversity**, that is, the variability of the cis-regulatory mechanisms encoded in the generated sequences sampled using the same conditioning value. Attribution maps are calculated and compared as an all-vs-all Euclidean distance. A higher value indicates that the generated sequences contain diverse mechanisms (i.e., variable arrangements of motifs).

Task-specific design assesses the ability to generate sequences with desired activity levels across tasks. We set the conditioning values for desired activities for each task, generate 500 sequences, and compare the distribution of oracle-predicted activities.

5 Experiments

To demonstrate the effectiveness of D3 in conditionally generating sequences, we benchmark its performance against previous methods using three high-quality experimental datasets: human promoters [14], fly enhancers [68], and cell-type-specific MPRA [33].

5.1 Designing human promoters

Dataset. The human promoter dataset is an established benchmark for DNA generative models [14]. It consists of 100,000 promoter sequences (1024 nt long, centered on transcription start sites) with corresponding transcription initiation signal profiles measured via CAGE-seq [24] from the FANTOM database [69]. The signal profiles represent transcription activity levels, which vary across genes depending on promoter content.

Baselines. We compared D3 against Bit-diffusion [70], D3PM [59], DDSM, and DFM. We used the same CNN specs and setup as the original DDSM study for a fair comparison. Sequences were conditionally generated using transcription initiation signal profiles as input. We also introduced and trained a new transformer architecture within the D3 and DFM framework (details in Appendix D).

Evaluation. In addition to conditional generation fidelity, which was previously established [14], we also evaluated the predictive distribution shift, percent identity, k-mer spectrum shift, motif enrichment, and motif co-occurrence, focusing on DFM- and D3-based models, using Sei [71] as an oracle. Generated sequences were conditioned using activities from test sequences, creating a matched set of generated and real sequences. See Appendix D for additional details.

Table 1: Performance of generated human promoter sequences.

	Functional similarity		**Sequence similarity**			**Compositional similarity**	
	Conditional generation fidelity	Predictive distribution shift	Percent identity		k-mer spectrum shift	Motif enrichment	Motif co-occurrence
Model	(MSE ↓)	(KS test statistic ↓)	(Average)	(Max)	(JS distance ↓)	(Pearson's r ↑)	(Frobenius norm ↓)
Bit Diffusion* (Bit encoding)	0.0414	-	-	-	-	-	-
BIT Diffusion* (one-hot encoding)	0.0395	-	-	-	-	-	-
D3PM-Uniform*	0.0375	-	-	-	-	-	-
DDSM*	0.0334	-	-	-	-	-	-
DFM-Conv*	0.0269	0.399	7.09	12.4	0.346	0.831	23.699
DFM-Tran	0.0318	0.336	6.49	12.4	0.216	0.873	12.924
D3-Conv	0.0259	0.093	8.57	12.4	0.003	0.997	0.746
D3-Tran	0.0219	0.052	9.06	12.4	0.002	0.999	1.487
Shuffle	0.0617	0.300	7.97	12.4	0.096	0.891	2.545

Results. D3 with a CNN (D3-Conv) achieved lower *conditional generation fidelity* (MSE) compared to the state-of-the-art DFM with a CNN (FM-Conv) (Table 1). A transformer architecture further improved MSE for D3 but worsened results for DFM. All generative models outperformed dinucleotide shuffled sequences, suggesting they learned useful sequence properties (to varying extents). D3-based models yielded lower *predictive distribution shifts* compared to FM-based models. All models did not memorize training sequences, indicated by low *percent identity* on par with shuffled sequences. The *k-mer spectrum shift* was substantially lower for D3-based models across different k-mer sizes (Fig. 3, Appendix E), while FM-based models showed higher shift than shuffled sequences, suggesting unnatural sequence biases. Similar trends for *motif enrichment* and *motif co-occurrence* suggest that D3-generated sequences better reflect real sequence composition.

5.2 Context-specific Fly Enhancer Design

Data. The data consists of enhancer sequences with corresponding activity values in two contexts (housekeeping and developmental gene promoters), taken from the STARR-seq dataset [68]. The dataset contains 402,296 training, 40,570 validation, and 41,186 test sequences. Each 249 nt enhancer sequence has two scalar activity values for the developmental and housekeeping conditions. We repurposed this dataset to train and evaluate generative models for context-specific enhancer activity.

Baselines. We trained DFM-Conv, DFM-Tran, D3-Conv, and D3-Tran with conditioning using activities from the training set. Training details are in Appendix D.

Evaluation. We performed the evaluations outlined in Sec. 4 using a DeepSTARR model trained with EvoAug [72] as the oracle, which employs evolution-inspired data augmentations and has demonstrated state-of-the-art performance on this dataset (Appendix D). For each method, we generated sequences conditioned on the test sequence activities for both tasks to obtain a matched set of generated and real sequences. Dinucleotide shuffled test sequences were used as a reference.

Results. D3-based models outperformed DFM-based models on evaluations for functional, sequence, and compositional similarity (Table 2). D3-Tran achieved the lowest MSE (*conditional generation fidelity*) and KS test statistic (*predictive distribution shift*), while D3-Conv had a slightly lower Frechet distance (*representational similarity*). According to *percent identity*, no model memorized training sequences. D3-based models yielded lower *k-mer spectrum shift* and *discriminatability* compared to DFM-based models, suggesting DFM-generated sequences contain biases easily recognized by a discriminator. Compositionally, D3-generated sequences contained *motif enrichment* and *motif combinations* at similar levels as real genomic sequences. Moreover, D3-generated sequences produced attribution maps visually similar to real sequences, with short, consistent motif-like patterns. In contrast, DFM-generated sequences exhibited spurious attribution scores for arbitrary positions (Fig. 5, Appendix E). The *attribution consistency* score showed that D3-Tran captures patterns more consistently within its own set of generated sequences and in combinations with the real sequences.

According to *added information*, a supervised DeepSTARR improved performance when trained with a combination of real sequences and D3-generated sequences (Figs. 6, Appendix E), with further improvements when augmenting the training data two-fold (Figs. 7, Appendix E) and remarkably, training DeepSTARR using only D3-Tran-generated sequences achieved nearly the

Table 2: Performance for fly enhancers. See Sec. 4 for details of each metric. Attribution consistency* represents analyses using combined attribution maps from generated sequences with real sequences. Red highlights indicate best performance.

	Functional similarity			**Sequence similarity**			
	Conditional generation fidelity	Predictive distribution shift	Representational similarity	Percent identity		k-mer spectrum shift	Discriminatability
Model	(MSE $\downarrow$)	(KS test statistic $\downarrow$)	(Frechet distance $\downarrow$)	(Average)	(Max)	(JS distance $\downarrow$)	(AUROC $\uparrow$)
DFM-Conv	2.029	0.231	1.109	38.9 ± 0.01	50.2	0.303	0.979 ± 0.014
DFM-Tran	2.182	0.212	1.059	40.4 ± 0.03	51.0	0.195	0.982 ± 0.019
D3-Conv	1.192	0.025	0.022	40.3 ± 0.02	51.0	0.004	0.516 ± 0.017
D3-Tran	1.156	0.020	0.033	40.4 ± 0.02	51.0	0.004	0.552 ± 0.034
Shuffle	1.708	0.184	0.803	40.0 ± 0.02	51.0	0.085	0.970 ± 0.003
Test set	-	-	-	-	-	-	-

	Compositional similarity			
	Motif enrichment	Motif co-occurrence	Attribution consistency	Attribution consistency generated + real sequences
Model	(Pearson's r $\uparrow$)	(Frobenius norm $\downarrow$)	(KLD $\uparrow$)	(KLD $\uparrow$)
DFM-Conv	0.450	100.5	0.485	0.459
DFM-Tran	0.656	104.5	0.438	0.450
D3-Conv	0.977	36.4	0.504	0.503
D3-Tran	0.984	19.3	0.525	0.514
Shuffle	0.743	81.0	0.084	0.376
Test set	-	-	0.52	-

same performance as training on the actual set (Figs. 7e, Appendix E), suggesting D3-Tran samples informative sequences (beyond the observed training sequences), providing additional information that benefits supervised modeling.

To ensure broad sampling, we also considered the *sequence diversity* and *mechanistic diversity* of sequences generated using a fixed conditioning value. We found that generated sequences exhibit a diverse set of sequences (Table 4, Appendix E), and their underlying cis-regulatory mechanisms were diversely represented in attribution maps (Fig. 8). Finally, we assessed the ability for *task-specific design* by generating sequences conditioned with specific activity settings across tasks. Strikingly, D3-Tran-generated sequences better reflected the task conditional values compared to DFM-Conv (Fig. 9), demonstrating that D3-Tran has learned a meaningful density over sequence space that enables sampling unseen sequences with desired activity levels across tasks.

5.3 Cell-type-specific Regulatory Sequence Design

Data. The dataset, created by Gosai et al. [33], consists of cis-regulatory sequences with activity values measured via an MPRA in three human cell lines: K562 (erythroid precursors), HepG2 (hepatocytes), and SK-N-SH (neuroblastoma). It contains 640,029 training, 59,697 validation, and 63,958 test sequences. Each 200 nt sequence has three scalar activity values for each cell type. We used this dataset to train and evaluate generative models for cell-type-specific sequence activity.

Baselines. We used the same models as in the fly enhancer analysis (see Appendix D for details).

Evaluation. We evaluated the conditional generation fidelity from Sec. 4 and explored the ability to generate sequences with cell-type-specific activity levels. The oracle was a custom ResidualBind model [26, 73] with dilated convolutional blocks and residual connections (details in Appendix D).

Results. Similarly, D3-based models led to lower MSE in *conditional generation fidelity* (Table 3, Appendix E). For *task-specific design*, D3-Tran generated sequences with appropriate activities when the conditioning activity values across cell types were the same (Fig. 10a, Appendix E). However, D3-Tran yielded lower efficacy when generating sequences were conditioned with highly differential activities across cell types (Fig. 10b-d, Appendix E). Further investigation revealed that task labels in the MPRA training set were highly correlated, with high activity examples being extremely rare (Fig. 11, Appendix E). In contrast, the fly enhancer dataset exhibited lower activity correlation across tasks and contained more high-activity examples. This suggests that while D3-Tran has learned a useful distribution capturing many observed properties of sequence-function space, its ability to generate unseen combinations of task-specific activities is limited. However, this issue could be a limitation of the supervised oracle's inability to generalize under covariate shifts. Experimental evidence is needed to resolve this discrepancy, but it remains out of scope for this study.

6 Conclusion

Here, we introduced DNA Discrete Diffusion, a generative model for conditionally sampling regulatory DNA with targeted functional activity levels. Through extensive benchmarking, we demonstrated that D3 outperforms existing methods in generating sequences that more accurately reflect the functional, sequence, and compositional properties of the regulatory genome. Notably, D3-generated sequences effectively augmented supervised models, improving predictive performance even in data-limited scenarios. These results highlight the potential of diffusion-based generative models to learn complex cis-regulatory mechanisms underlying gene regulation, which opens new avenues for targeted sequence design in biomedicine and synthetic biology. Future work could explore model distillation to improve sampling efficiency, increase the scale of generated sequences to incorporate interactions across distal cis-regulatory elements, and improve out-of-distribution generation.

References

[1] Gökcen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

[2] Peter K Koo and Matt Ploenzke. Deep learning for inferring transcription factor binding sites. *Current opinion in systems biology*, 19:16–23, 2020.

[3] Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18(10):1196–1203, 2021.

[4] Kseniia Dudnyk, Donghong Cai, Chenlai Shi, Jian Xu, and Jian Zhou. Sequence basis of transcription initiation in the human genome. *Science*, 384(6694):eadj0116, 2024.

[5] Žiga Avsec, Melanie Weilert, Avanti Shrikumar, Sabrina Krueger, Amr Alexandari, Khyati Dalal, Robin Fropf, Charles McAnany, Julien Gagneur, Anshul Kundaje, et al. Base-resolution models of transcription-factor binding reveal soft motif syntax. *Nature genetics*, 53(3):354–366, 2021.

[6] Eeshit Dhaval Vaishnav, Carl G de Boer, Jennifer Molinet, Moran Yassour, Lin Fan, Xian Adiconis, Dawn A Thompson, Joshua Z Levin, Francisco A Cubillos, and Aviv Regev. The evolution, evolvability and engineering of gene regulatory dna. *Nature*, 603(7901):455–463, 2022.

[7] Ibrahim I Taskiran, Katina I Spanier, Hannah Dickmänken, Niklas Kempynck, Alexandra Pančíková, Eren Can Ekşi, Gert Hulselmans, Joy N Ismail, Koen Theunis, Roel Vandepoel, et al. Cell-type-directed design of synthetic enhancers. *Nature*, 626(7997):212–220, 2024.

[8] Bernardo P de Almeida, Christoph Schaub, Michaela Pagani, Stefano Secchia, Eileen EM Furlong, and Alexander Stark. Targeted design of synthetic enhancers for selected tissues in the drosophila embryo. *Nature*, 626(7997):207–211, 2024.

[9] Johannes Linder and Georg Seelig. Fast differentiable dna and protein sequence optimization for molecular design. *arXiv preprint arXiv:2005.11275*, 2020.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[11] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[12] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[13] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[14] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pages 1276–1301. PMLR, 2023.

[15] Zehui Li, Yuhao Ni, William AV Beardall, Guoxuan Xia, Akashaditya Das, Guy-Bart Stan, and Yiren Zhao. Discdiff: Latent diffusion model for dna sequence generation. *arXiv preprint arXiv:2402.06079*, 2024.

[16] Lucas Ferreira DaSilva, Simon Senan, Zain Munir Patel, Aniketh Janardhan Reddy, Sameer Gabbita, Zach Nussbaum, Cesar Miguel Valdez Cordova, Aaron Wenteler, Noah Weber, Tin M Tunjic, et al. Dna-diffusion: Leveraging generative models for controlling chromatin accessibility and gene expression via synthetic regulatory elements. *bioRxiv*, pages 2024–02, 2024.

[17] Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.

[18] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

[19] Dmitry Penzar, Daria Nogina, Elizaveta Noskova, Arsenii Zinkevich, Georgy Meshcheryakov, Andrey Lando, Abdul Muntakim Rafi, Carl de Boer, and Ivan V Kulakovskiy. Legnet: a best-in-class deep learning model for short dna regulatory regions. *Bioinformatics*, 39(8):btad457, 2023.

[20] Sergey Nurk, Sergey Koren, Arang Rhie, Mikko Rautiainen, Andrey V Bzikadze, Alla Mikheenko, Mitchell R Vollger, Nicolas Altemose, Lev Uralsky, Ariel Gershman, et al. The complete sequence of a human genome. *Science*, 376(6588):44–53, 2022.

[21] Tong Ihn Lee and Richard A Young. Transcriptional regulation and its misregulation in disease. *Cell*, 152 (6):1237–1251, 2013.

[22] François Spitz and Eileen EM Furlong. Transcription factors: from enhancer binding to developmental control. *Nature reviews genetics*, 13(9):613–626, 2012.

[23] Daria Shlyueva, Gerald Stampfel, and Alexander Stark. Transcriptional enhancers: from properties to genome-wide predictions. *Nature Reviews Genetics*, 15(4):272–286, 2014.

[24] Toshiyuki Shiraki, Shinji Kondo, Shintaro Katayama, Kazunori Waki, Takeya Kasukawa, Hideya Kawaji, Rimantas Kodzius, Akira Watahiki, Mari Nakamura, Takahiro Arakawa, et al. Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage. *Proceedings of the National Academy of Sciences*, 100(26):15776–15781, 2003.

[25] Justin B Kinney and David M McCandlish. Massively parallel assays and quantitative sequence–function relationships. *Annual review of genomics and human genetics*, 20:99–127, 2019.

[26] Peter K Koo, Antonio Majdandzic, Matthew Ploenzke, Praveen Anand, and Steffan B Paul. Global importance analysis: An interpretability method to quantify importance of genomic features in deep neural networks. *PLoS computational biology*, 17(5):e1008925, 2021.

[27] Alexander Karollus, Thomas Mauermeier, and Julien Gagneur. Current sequence-based models capture gene expression determinants in promoters but mostly ignore distal enhancers. *Genome Biology*, 24(1): 56, 2023.

[28] Shushan Toneyan and Peter K Koo. Interpreting cis-regulatory interactions from large-scale deep neural networks for genomics. *bioRxiv*, 2023.

[29] Surag Nair, Mohamed Ameen, Laksshman Sundaram, Anusri Pampari, Jacob Schreiber, Akshay Balsubramani, Yu Xin Wang, David Burns, Helen M Blau, Ioannis Karakikes, et al. Transcription factor stoichiometry, motif affinity and syntax regulate single-cell chromatin dynamics during fibroblast reprogramming to pluripotency. *bioRxiv*, 2023.

[30] Gabriella E Martyn, Michael T Montgomery, Hank Jones, Katherine Guo, Benjamin R Doughty, Johannes Linder, Ziwei Chen, Kelly Cochran, Kathryn A Lawrence, Glen Munson, et al. Rewriting regulatory dna to dissect and reprogram gene expression. *bioRxiv*, pages 2023–12, 2023.

[31] Evan E Seitz, David M McCandlish, Justin B Kinney, and Peter K Koo. Interpreting cis-regulatory mechanisms from genomic deep neural networks using surrogate models. *bioRxiv*, 2023.

[32] Adam Y He and Charles G Danko. Dissection of core promoter syntax through single nucleotide resolution modeling of transcription initiation. *bioRxiv*, pages 2024–03, 2024.

[33] Sager J Gosai, Rodrigo I Castro, Natalia Fuentes, John C Butts, Susan Kales, Ramil R Noche, Kousuke Mouri, Pardis C Sabeti, Steven K Reilly, and Ryan Tewhey. Machine-guided design of synthetic cell type-specific cis-regulatory elements. *bioRxiv*, 2023.

[34] Nathan Killoran, Leo J Lee, Andrew Delong, David Duvenaud, and Brendan J Frey. Generating and designing dna with deep generative models. *arXiv preprint arXiv:1712.06148*, 2017.

[35] Jan Zrimec, Xiaozhi Fu, Azam Sheikh Muhammad, Christos Skrekas, Vykintas Jauniskis, Nora K Speicher, Christoph S Börlin, Vilhelm Verendel, Morteza Haghir Chehreghani, Devdatt Dubhashi, et al. Controlling gene expression with deep generative design of regulatory dna. *Nature communications*, 13 (1):5099, 2022.

[36] Ye Wang, Haochen Wang, Lei Wei, Shuailin Li, Liyang Liu, and Xiaowo Wang. Synthetic promoter design in escherichia coli based on a deep generative network. *Nucleic Acids Research*, 48(12):6403–6412, 2020.

[37] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[38] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[39] Johannes Linder, Nicholas Bogard, Alexander B Rosenberg, and Georg Seelig. A generative neural network for maximizing fitness and diversity of synthetic dna and protein sequences. *Cell systems*, 11(1): 49–62, 2020.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv 1810.04805*, 2018.

[41] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Hassan Sirelkhatim, Guillaume Richard, et al. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, 2023.

[42] Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv*, 2023.

[43] Gonzalo Benegas, Sanjit Singh Batra, and Yun S Song. Dna language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences*, 120(44):e2311219120, 2023.

[44] Veniamin Fishman, Yuri Kuratov, Maxim Petrov, Aleksei Shmelev, Denis Shepelin, Nikolay Chekanov, Olga Kardymon, and Mikhail Burtsev. Gena-lm: A family of open-source foundational models for long dna sequences. *bioRxiv*, pages 2023–06, 2023.

[45] Yanyi Chu, Dan Yu, Yupeng Li, Kaixuan Huang, Yue Shen, Le Cong, Jason Zhang, and Mengdi Wang. A 5 utr language model for decoding untranslated regions of mrna and function predictions. *Nature Machine Intelligence*, pages 1–12, 2024.

[46] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Callum Birch-Sykes, Michael Wornow, Aman Patel, Clayton Rabideau, Stefano Massaroli, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*, 2023.

[47] Eric Nguyen, Michael Poli, Matthew G. Durrant, Armin W. Thomas1, Brian Kang, Jeremy Sullivan, Madelena Y. Ng, Ashley Lewis, Aman Patel, Aaron Lou, Stefano Ermon, Stephen A. Baccus, Tina Hernandez-Boussard1, Christopher Re, Patrick D. Hsu, and Brian L. Hie. Sequence modeling and design from molecular to genome scale with evo. *bioRxiv*, 2024.

[48] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *arXiv 2403.03234*, 2024.

[49] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[50] Ziqi Tang and Peter K Koo. Evaluating the representational power of pre-trained dna language models for regulatory genomics. *bioRxiv*, pages 2024–02, 2024.

[51] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[52] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637), 2023.

[53] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, pages 1–8, 2023.

[54] Noelia Ferruz and Birte Höcker. Controllable protein design with language models. *Nature Machine Intelligence*, 4(6):521–532, 2022.

[55] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

[56] William J Anderson. *Continuous-time Markov chains: An applications-oriented approach*. Springer Science & Business Media, 2012.

[57] Frank P Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[58] Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

[59] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[60] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[61] Gary D Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000.

[62] Ieva Rauluseviciute, Rafael Riudavets-Puig, Romain Blanc-Mathieu, Jaime A Castro-Mondragon, Katalin Ferenc, Vipin Kumar, Roza Berhanu Lemma, Jérémy Lucas, Jeanne Chèneby, Damir Baranasic, et al. Jaspar 2024: 20th anniversary of the open-access database of transcription factor binding profiles. *Nucleic Acids Research*, 52(D1):D174–D182, 2024.

[63] Charles E Grant, Timothy L Bailey, and William Stafford Noble. Fimo: scanning for occurrences of a given motif. *Bioinformatics*, 27(7):1017–1018, 2011.

[64] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[65] Antonio Majdandzic, Chandana Rajesh, and Peter K Koo. Correcting gradient-based interpretations of deep neural networks for genomics. *Genome Biology*, 24(1):109, 2023.

[66] Ammar Tareen and Justin B Kinney. Logomaker: beautiful sequence logos in python. *Bioinformatics*, 36 (7):2272–2274, 2020.

[67] Antonio Majdandzic, Chandana Rajesh, Ziqi Tang, Shushan Toneyan, Ethan L Labelson, Rohit K Tripathy, and Peter K Koo. Selecting deep neural networks that yield consistent attribution-based interpretations for genomics. In *Machine Learning in Computational Biology*, pages 131–149. PMLR, 2022.

[68] Bernardo P de Almeida, Franziska Reiter, Michaela Pagani, and Alexander Stark. Deepstarr predicts enhancer activity from dna sequence and enables the de novo design of synthetic enhancers. *Nature genetics*, 54(5):613–624, 2022.

[69] AR Forrest, H Kawaji, M Rehli, JK Baillie, MJ de Hoon, V Haberle, T Lassmann, IV Kulakovskiy, M Lizio, M Itoh, et al. A promoter-level mammalian expression atlas. *Nature*, 507(7493):462–470, 2014.

[70] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

[71] Kathleen M Chen, Aaron K Wong, Olga G Troyanskaya, and Jian Zhou. A sequence-based global map of regulatory activity for deciphering human genetics. *Nature genetics*, 54(7):940–949, 2022.

[72] Nicholas Keone Lee, Ziqi Tang, Shushan Toneyan, and Peter K Koo. Evoaug: improving generalization and interpretability of genomic deep neural networks with evolution-inspired data augmentations. *Genome Biology*, 24(1):105, 2023.

[73] Shushan Toneyan, Ziqi Tang, and Peter K Koo. Evaluating deep learning for predicting epigenomic profiles. *Nature machine intelligence*, 4(12):1088–1100, 2022.

[74] Gherman Novakovsky, Nick Dexter, Maxwell W Libbrecht, Wyeth W Wasserman, and Sara Mostafavi. Obtaining genetics insights from deep learning via explainable artificial intelligence. *Nature Reviews Genetics*, 24(2):125–137, 2023.

[75] Martin Kavšček, Martin Stražar, Tomaž Curk, Klaus Natter, and Uroš Petrovič. Yeast as a cell factory: current state and perspectives. *Microbial cell factories*, 14:1–10, 2015.

[76] David Jullesson, Florian David, Brian Pfleger, and Jens Nielsen. Impact of synthetic biology and metabolic engineering on industrial production of fine chemicals. *Biotechnology advances*, 33(7):1395–1402, 2015.

[77] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[78] Alexander Karollus, Johannes Hingerl, Dennis Gankin, Martin Grosshauser, Kristian Klemon, and Julien Gagneur. Species-aware dna language models capture regulatory elements and their evolution. *bioRxiv*, 2023.

[79] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

[80] Cen Wan and David T Jones. Protein function prediction is improved by creating synthetic feature samples with generative adversarial networks. *Nature Machine Intelligence*, 2(9):540–550, 2020.

[81] Donatas Repecka, Vykintas Jauniskis, Laurynas Karpus, Elzbieta Rembeza, Irmantas Rokaitis, Jan Zrimec, Simona Poviloniene, Audrius Laurynenas, Sandra Viknander, Wissam Abuajwa, et al. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*, 3 (4):324–333, 2021.

[82] Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K Min, Kelly Brock, Yarin Gal, and Debora S Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 599(7883):91–95, 2021.

[83] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

[84] Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of chemical physics*, 115(4):1716–1733, 2001.

[85] Ricky TQ Chen and David K Duvenaud. Neural networks with cheap differential operators. *Advances in Neural Information Processing Systems*, 32, 2019.

[86] Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *Advances in Neural Information Processing Systems*, 35:34532–34545, 2022.

[87] Pierre H Richemond, Sander Dieleman, and Arnaud Doucet. Categorical sdes with simplex diffusion. *arXiv preprint arXiv:2210.14784*, 2022.

[88] Griffin Floto, Thorsteinn Jonsson, Mihai Nica, Scott Sanner, and Eric Zhengyu Zhu. Diffusion on the probability simplex. *arXiv preprint arXiv:2309.02530*, 2023.

[89] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022.

[90] Nathan C Frey, Daniel Berenberg, Karina Zadorozhny, Joseph Kleinhenz, Julien Lafrance-Vanasse, Isidro Hotzel, Yan Wu, Stephen Ra, Richard Bonneau, Kyunghyun Cho, et al. Protein discovery with discrete walk-jump sampling. *arXiv preprint arXiv:2306.12360*, 2023.

[91] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

[92] T Dao, DY Fu, S Ermon, A Rudra, and C Flashattention Ré. Fast and memory-efficient exact attention with io-awareness, 2022. *URL https://arxiv. org/abs/2205.14135*.

[93] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[94] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[95] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[96] Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[97] EA Feingold, PJ Good, MS Guyer, S Kamholz, L Liefer, K Wetterstrand, FS Collins, TR Gingeras, D Kampa, EA Sekinger, et al. The encode (encyclopedia of dna elements) project. *Science*, 306(5696): 636–640, 2004.

[98] Peter K Koo and Matt Ploenzke. Improving representations of genomic sequence motifs in convolutional networks with exponential activations. *Nature machine intelligence*, 3(3):258–266, 2021.

[99] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv 1412.6980*, 2014.

[100] Aniketh Janardhan Reddy, Michael H Herschl, Sathvik Kolli, Amy X Lu, Xinyang Geng, Aviral Kumar, Patrick D Hsu, Sergey Levine, and Nilah M Ioannidis. Pretraining strategies for effective promoter-driven gene expression prediction. *bioRxiv*, 2023.

A Appendix: brief primer on regulatory genomics

The human genome consists of DNA sequences comprising 23 pairs of chromosomes, totaling approximately 3 billion base pairs [20]. While most cells share the same DNA, different cell types (e.g., heart, neurons, muscle) arise due to differential gene regulation. Protein-coding genes constitute 1-2% of the genome; large regions of the non-coding genome regulate transcription, the process of converting DNA to mRNA for protein translation [21]. Important non-coding regions called cis-regulatory elements act as promoters and enhancers, among others [23].

Promoters initiate and maintain mRNA transcription. Transcription requires RNA polymerase recruited to genes through interactions with transcription factors (TFs) that bind short DNA sequences (4-19 nucleotides). Amino acid-DNA interactions determine TF binding specificity; TF binding sites are called sequence 'motifs'. TF binding is influenced by factors such as nucleotide content, neighboring context, other TFs, and cofactors [22], resulting in a complex and partially stochastic process. Human cells contain 1,500-2,000 TF types expressed at varying levels across cell types. Enhancers are DNA sequence elements (200-1kb long) that act as TF binding hotspots, interacting with promoters to enhance polymerase recruitment [23]. The cell-type-specific regulatory code governing which TFs bind where and how they interact to coordinate transcription remains largely unknown.

Experimental approaches have advanced quantitative measurements of how regulatory sequences impact gene expression via high-throughput chromatin profiling screens (e.g., CAGE-seq [24] measuring mRNA levels) or massively parallel reporter assays (MPRAs). High-throughput chromatin profiling screens measure the genome-wide occupancy of TFs, chromatin accessibility, or histone modifications, providing indirect readouts of regulatory activity. MPRAs measure the quantitative effects of sequence elements like promoters or enhancers on reporter expression (e.g., mRNA, fluorescence, barcodes), providing direct readouts of their regulatory activity [25].

While fitting deep learning models to functional assays in a supervised manner has enabled some insights into regulatory elements and motif interactions [74, 26, 29, 30, 28, 31, 32], our understanding of the complex cis-regulatory code governing cell-type-specific gene expression remains limited. Current methods provide only partial insights into the intricate mechanisms of transcriptional regulation.

The ability to precisely design regulatory sequences for desired expression patterns would revolutionize therapeutics via targeted mRNA delivery and industrial biotechnology through optimized protein production in bacteria or yeast [75, 76]. However, realizing these transformative applications requires going beyond interpreting the genomic regulatory code to generative modeling approaches that encode this complex biological knowledge. Despite recent advances, it remains to be seen whether the few generative models that have recently emerged can reliably learn and recapitulate the context-dependent interactions orchestrating how transcription factors, enhancers, promoters, and other elements control gene programs across diverse cellular environments.

The main approaches to designing regulatory sequences involve using trained, supervised models to score proposed sequences from search strategies that sample from or navigate along the design space. In genomics, search methods include in silico evolution (computational simulation of evolutionary processes) [6], gradient ascent (iterative optimization by following the gradient of a target function) [9], simulated annealing (a probabilistic optimization algorithm) [33], motif implanting (inserting known TF binding motifs into sequences) [7], and reinforcement learning methods like GFlowNets (learning a policy to generate sequences with desired properties) [77]. In terms of generative models, generative adversarial networks (GANs) have also been proposed but have yet to generate compelling sequences [34, 35], albeit no further follow-up studies have demonstrated the capabilities of GANs. Surprisingly, other generative AI methods like variational autoencoders [37] and normalizing flows [38] have not reliably established an ability to generate regulatory sequences. Recently, DNA language models have emerged [41, 46, 44, 42, 43, 78, 45, 47, 48, 49], demonstrating promise in learning gene features in the genome. However, careful evaluation of their representations has shown that they need to learn a foundational understanding of regulatory sequences in humans [50].

By contrast, generative models have found success in learning distributions of homologous protein sequences [79, 80, 81, 82] and large language models have learned functional and structural features from sequence alone [51, 52, 53, 54]. The distinction is that protein sequences exhibit deep conservation across distant species spanning millions of years, while non-coding genomic regions are highly divergent at the sequence level. Importantly, unlike natural languages or proteins, regulatory DNA sequences contain long stretches with only some positions informative for transcription factor binding, creating a large class imbalance of uninformative versus informative positions. Loss functions aiming to predict missing tokens or reconstruct input sequences are dominated by positions that require memorization due to their high entropy (i.e., low information content), limiting their effectiveness for modeling regulatory sequences.

B Appendix: Method details

B.1 Discrete Diffusion Process

Our aim is to model discrete sequence data $x_0 \in \mathcal{X}$ with finite cardinality $\mathcal{S} = |\mathcal{X}|$ and $x_0 \sim q_{data}(x_0)$ for some discrete data distribution $q_{data}(x_0)$. Similar to a continuous process, a forward noising process can be defined that transforms $p_{data}(x_0)$ to some terminal distribution $q_K(x_K)$ that closely approximates an easy-to-sample distribution $p_{ref}(x_K)$. Following the notion of discrete state spaces, a forward kernel can be defined as $q_{k+1|k}(x_{k+1|k})$ that tends to the stationary distribution as $k \to \infty$. A simple uniform kernel can be defined as $q_{k+1|k}(x_{k+1|k}) = \delta_{x_{k+1|k}}(1-\beta) + (1-\delta_{x_{k+1|k}})\beta/(\mathcal{S}-1)$ [83, 59] where δ is Kronecker delta with the stationary distribution being uniform distribution over all states. With a valid definition of $q_{k+1|k}$, a forward joint decomposition is defined as below:

$$q_{0:K}(x_{0:K}) = q_{data}(x_0) \prod_{k=0}^{K-1} q_{k+1|k}(x_{k+1|k})$$

The joint distribution can be decomposed with the reverse transitions as given below:

$$q_{0:K}(x_{0:K}) = q_K(x_K) \prod_{k=0}^{K-1} q_{k|k+1}(x_k|x_{k+1}) \text{ where } q_{k|k+1}(x_k|x_{k+1}) = \frac{q_{k+1|k}(x_{k+1|k})q_k(x_k)}{q_{k+1}(x_{k+1})}$$

With access to reverse transition kernel $q_{k|k+1}$, it is possible to sample from the stationary distribution $x_K \sim q_K(.)$ and then sample from the reverse kernels to produce $q_{data}(x_0)$ as the last step of the process. Generally, a parametric reverse kernel $p^{\theta}_{k|k+1}$ is employed to approximate the intractable $q_{k|k+1}$ and samples can be generated from $p_{data}(x_0)$ can be obtained by generative joint distribution

$$p^{\theta}_{k|k+1}(x_{0:K}) = p_{ref}(x_K) \prod_{k=0}^{K-1} p^{\theta}_{k|k+1}(x_k|x_{k+1})$$

where $q_K(x_K)$ is expected to approximate the stationary distribution $p_{ref}(x_K)$ closely for large K.

B.2 Continuous Time Framework

The transitions can occur at any time during $t = 0$ to $t = T$ following a continuous time Markov process, which is more flexible compared to the discrete-time scenario. This process starts with an initial distribution q_0 and is driven by a transition rate matrix $\mathcal{R}_t \in \mathbb{R}^{S \times S}$ that defines how the state changes during the process. The corresponding discrete diffusion process can be denoted by a linear ordinary differential $\frac{dq_t}{dt} = \mathcal{R}_t q_t$ where q_t represents the family of distributions that evolves with time t that starts with $q_0 \approx q_{data}$. The transition rate matrix $\mathcal{R}_t$ should have non-negative non-diagonal entries and columns summed to zero that restrict p_t to gaining or losing any mass during the process. $\mathcal{R}_t$ can be analytically chosen as $\mathcal{R}_t = \beta(t)\mathcal{R}_b$ with a time-dependent scalar factor β_t and time independent base rate matrix $\mathcal{R}_b$ such that p_t approaches a stationary distribution p_{ref} as $t \to \infty$.

$\mathcal{R}_t$ can also represent infinitesimal transition probability and corresponding small Δt Euler steps for the process at any time points $t - \Delta t$ and t as below:

$$q_{t|t-\Delta t}(x|\bar{x}) = \delta_{x,\bar{x}} + \mathcal{R}_t(\bar{x}, x)\Delta t + \mathcal{O}(\Delta t) \tag{10}$$

The reverse process starts from the marginal $q_T(x_T)$ and can lead to data distribution $p_{data}(x_0)$ with suitable transition rate matrix $\bar{\mathcal{R}}_t \in \mathbb{R}^{S \times S}$ as:

$$q_{t|t+\Delta t}(\bar{x}|x) = \delta_{\bar{x},x} + \bar{\mathcal{R}}_t(x, \bar{x})\Delta t + \mathcal{O}(\Delta t) \tag{11}$$

Also the reverse transition matrix satisfies

$$\bar{\mathcal{R}}_t(x, \bar{x}) = \frac{q_t(\bar{x})}{q_t(x)}\mathcal{R}_t(\bar{x}, x) \tag{12}$$

and the reverse process ODE can be given by

$$\frac{dq_{T-t}}{dt} = \bar{\mathcal{R}}_{T-t} q_{T-t} \tag{13}$$

The ratios $\frac{q_t(\bar{x})}{q_t(x)}$ are collectively known as concrete score, but are generally intractable. Their behavioral similarity to score function $\nabla \log q(x)$ for continuous distributions instigates learning the ratios for discrete data points (such as genomics data that we are trying to model).

B.3 Properties and Likelihood Bound

For any fully supported distribution q and positive weights $w_{x\bar{x}}$, the optimal θ^* minimizing $\mathcal{L}_{SE}$ for all $x, \bar{x}$ pairs satisfies $s_{\theta^*}(x)_{\bar{x}} = \frac{q(\bar{x})}{q(x)}$, with the minimum value being 0 at θ^*. This can be shown by taking derivative of $\mathcal{L}_{SE}$ (without considering $K(.)$ as it doesn't depend on θ) w.r.t. s_θ and setting it to 0 [18].

Unfortunately, the score entropy term contains the unknown $\frac{q_t(\bar{x})}{q_t(x)}$ term, which requires planning for an alternate solution to tractably calculate $\mathcal{L}_{SE}$. Following Theorem 3.4 in [18], we continue with calculating denoising score entropy as an equivalent (up to a constant independent of θ) of $\mathcal{L}_{SE}$. If q is a perturbation of a base density q_0 by a transition kernel $q(.|.)$ such as $q(x) = \sum_{x_0} q(x|x_0)q_0(x_0)$, $\mathcal{L}_{DSE}$ is given by

$$\mathcal{L}_{DSE} = \mathbb{E}_{x_0 \sim q_0, x \sim q(.|x_0)}[\sum_{\bar{x} \neq x} w_{x\bar{x}}(s_\theta(x)_{\bar{x}} - \frac{q(\bar{x}|x_0)}{q(x|x_0)} \log s_\theta(x)_{\bar{x}})] \tag{14}$$

Access to the perturbation kernel makes $\mathcal{L}_{DSE}$ scalable as the hope is to train a model $s_\theta(x)$ that can provide any $s_\theta(x)_{\bar{x}}$. We will discuss different perturbation techniques in the following subsection. Particularly, defining an ELBO and likelihood-based training requires defining a time-dependent score network $s_\theta(., t)$ with the parameterized reverse matrix as $\bar{\mathcal{R}}_t(\bar{x}, x) = s_\theta(x, t)\mathcal{R}_t(x, \bar{x})$ which is found by replacing the density ratios with scores in Equation 12. Also, the parameterized densities q_t^θ satisfy reverse process ODE in Equation 13. The log-likelihood of data points was derived for discrete diffusion models in [55], with a specific form for score entropy in [18] as

$$-log q_0^\theta(x_0) \leq \mathcal{L}_{DWDSE}(x_0) + D_{KL}(q_{T|0}(.|x_0)||p_{ref}) \tag{15}$$

where $q_T^\theta = p_{ref}$ for the parameterized densities q_t^θ, and $\mathcal{L}_{DWDSE}(x_0)$ is the diffusion weighted denoising score entropy for x_0, which is given by

$$\int_0^T \mathbb{E}_{x_t \sim q_{t|0}(.|x_0)} \sum_{\bar{x} \neq x_t} \mathcal{R}_t(\bar{x}_t, t) \left(s_\theta(x_t, t)_{\bar{x}} - \frac{q_{t|0}(\bar{x}|x_0)}{q_{t|0}(x|x_0)} \log s_\theta(x_t, t)_{\bar{x}} + K\left(\frac{q_{t|0}(\bar{x}|x_0)}{q_{t|0}(x|x_0)})\right)\right) dt \tag{16}$$

We follow the likelihood based training off of this upper bound with specific choices for forward transition kernel that allow easy calculation of $\mathcal{L}_{DWDSE}$ which we discuss now in detail.

B.4 Choice of Forward Process (specific to genomics data)

Modeling highly complex genomics sequences of the form $\mathbf{x} = x^1...x^d$ with a high value of d through score entropy incurs real scaling challenges. Our state factorizes into sequences $\mathcal{X} = \{A, C, G, T\}^d$ such that each position of the sequence represents one nucleotide i.e. $x^i \in \{A, C, G, T\}$, which is synonymous to tokens for language models or image pixels for computer vision tasks. $\mathcal{R}_t$ should be able to take any sequence to any other sequence which requires exponential time and space, specifically $\mathcal{O}(4^d)$. Here, we consider a sparse structured matrix $\mathcal{R}_t^N$ that can perturb every nucleotide independently, which is given by

$$\mathcal{R}_t(x^1...x^i...x^d, x^1...\bar{x}^i...x^d) = \mathcal{R}_t^N(x^i, \bar{x}^i) \tag{17}$$

Specifically, we model all the density ratios for sequences with a Hamming distance of 1, which drastically reduces the span of perturbations to only sequences with one nucleotide change. This leads to the score network to model a seq-to-seq map with $\mathcal{O}(4d)$ time as below:

$$(s_\theta(x^1...x^i...x^d, t))_{i, x\to\bar{x}} \approx \frac{q_t(x^1...\bar{x}^i...x^d)}{q(x^1...x^i...x^d)} \tag{18}$$

As we consider only one token change, the calculation for the forward transition $q^{seq}_{t|0}(.|.)$ can be done in a single step by decomposing it to independent token perturbation:

$$q^{seq}_{t|0}(.|x) = \prod_{i=1}^{d} q^{N}_{t|0}(\bar{x}^i, x^i) \tag{19}$$

Each $q^{N}_{t|0}(\bar{x}^i, x^i)$ can be calculated with a noise level β and a pre-decided transition matrix $\mathcal{R}^N$ as $\mathcal{R}^N_t = \beta(t)\mathcal{R}^N$. Considering $\bar{\beta}_t$ as the cumulative noise $\int_0^t \beta(t)dt$, we have $q^{N}_{t|0}(.|x) = x$-th column of $exp(\bar{\beta}(t)\mathcal{R}^N)$. More specifically, that implies

$$q^{N}_{t|0}(\bar{x}_t = j|x_0 = i) = exp(\bar{\beta}(t)\mathcal{R}^N)(j, i) \tag{20}$$

where (j, i) represents jth row,ith column entry of the matrix $exp(\bar{\beta}(t)\mathcal{R}^N)$. With this formulation, We follow the diffusion-weighted denoising score entropy loss $\mathcal{L}_{DWDSE}$ from [18] for our model training.

Storing all possible edge weights $\mathcal{R}^N(i, j)$ requires a high cost in terms of space and time. This issue can be addressed by considering special structures for transition matrix [55, 59]. In particular, a uniform stationary base rate matrix $\mathcal{R} = \mathbb{1}\mathbb{1}^\top - \mathcal{S}\mathcal{I}$ can be a natural choice for categorical discrete distributions where $\mathbb{1}\mathbb{1}^\top$ is a matrix of all ones, $\mathcal{I}$ is identity matrix. For our specific scenario, the token size is 4, and hence the corresponding time-dependent transition matrix is given by

$$\begin{bmatrix} \frac{1}{4} + \frac{3}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} \\ \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} + \frac{3}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} \\ \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} + \frac{3}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} \\ \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} - \frac{1}{4}e^{-4t} & \frac{1}{4} + \frac{3}{4}e^{-4t} \end{bmatrix}$$

This structured matrix perturbs the input sequence according to the uniformly sampled time t, with the idea that all the matrix entries tend to $\frac{1}{4}$ uniformly when $t \to \infty$. This formulation explained above enables us to calculate all values of $\mathcal{L}_{DWDSE}$ quickly, which makes the training process much faster. Specifically, $\mathcal{L}_{DWDSE}$ can be calculated as below:

$$\mathcal{L}_{DWDSE} = \beta(t) \sum_{i=1}^{d} \sum_{y=1}^{4} (1 - \delta_{\bar{x}^i_t}(y))(s_\theta(\bar{x}_t, t)_{i,y} - \frac{q_{t|0}(y|x^i_0)}{q_{t|0}(\bar{x}^i_t|x^i_0)} \log s_\theta(\bar{x}_t, t)_{i,y} \tag{21}$$

where $\beta(t)$ is noise schedule (with total noise $\bar{\beta}(t)$, and δ is Kronecker delta. While other options are available, such as absorbing state process with a stationary matrix of all zeros[55, 18], our experiments with uniform rate matrix showed exciting results, and we leave other options for future work.

B.5 Simulating Generative Reverse Process

The parametric generative reverse process following continuous time framework with rate matrix $\bar{\mathcal{R}}$ can be employed by taking a small Euler step shown in Eq. 11. While this process can be simulated using Gillespie's Algorithm [84], this is impractical for large sequence lengths as stepping through each transition individually will allow only one dimension change for each backward step [55]. Another popular approximate simulation method, tau-leaping [84], can be employed that applies all the transitions simultaneously. Particularly, a sequence $x_{t-\Delta t}$ can be constructed from x_t by sampling each token $x^i_{t-\Delta t}$ independently [18] by below probability:

$$q^i(x^i_{t-\Delta t}|x^i_t) = \delta_{x^i_t}(x^i_{t-\Delta t}) + \Delta t \bar{\mathcal{R}}_t(x^i_t, x^i_{t-\Delta t}) s_\theta(x_t, t)_{i, x^i_{t-\Delta t}} \tag{22}$$

B.6 Utilizing Concrete Scores

The optimal denoising can be achieved only by knowing all density ratios $\frac{q_t(\bar{x})}{q_t(x)}$. If q_t follows the diffusion ODE $\frac{dq_t}{dt} = \mathcal{R}_t q_t$, the true denoiser can be represented as a discrete version of Tweedie's theorem [60], as shown in [18]

$$q_{0|t}(x_0|x_t) = \left(exp(-t\mathcal{R}) \left[\frac{q_t(i)}{q_t(x_t)} \right]_{i=1}^{4} \right)_{x_0} exp(t\mathcal{R}(x_t, x_0) \tag{23}$$

As we only know the density rations for sequences with 1 Hamming distance away, a Tweedie denoiser analog of tau-leaping to construct transition densities can be obtained by replacing the token transition probabilities as below:

$$q^i(x^i_{t-\Delta t}|x^i_t) = \big(exp(-\beta(t)^{\Delta t}\mathcal{R})s_\theta(x_t,t)_i\big)_{x^i_{t-\Delta t}}\, exp(-\beta(t)^{\Delta t}\mathcal{R}(x^i_t, x^i_{t-\Delta t}) \tag{24}$$

where $\beta(t)^{\Delta t} = (\bar{\beta}(t) - \bar{\beta}(t-\Delta t))$. Now $q^i(x^i_{t-\Delta t}|x^i_t)$ may need clamping for minimum 0 and normalizing the sum to 1, and $x_{t-\Delta t}$ can be obtained by sampling i-th token as $x^i_{t-\Delta t} \sim q^i(x^i_{t-\Delta t})$.

C Appendix: Related Work

Discrete Diffusion Models. There are previous methods that aimed to learn the density ratios $\frac{q_t(\bar{x})}{q_t(x)}$ and accordingly construct the reverse process stated above. Austin et al. [59], Campbell et al. [55] followed the strategy from [10] to learn the reverse density $q_{0|t}$ which can recover the density ratios [18]. This framework does not perform well empirically, as learning a density is harder than a value (or density ratios). On the other hand, the ratio matching method Sun et al. [58] follows maximum likelihood training to learn marginal probabilities, which is inherently different from score matching and requires expensive architecture for better performance [85]. Meng et al. [86] employs Fisher divergence to learn density ratios through score matching network that outputs $s_\theta(x, t)$ approximating $\frac{q_t(\bar{x})}{q_t(x)}$. As l^2 loss does not constrain the density ratios to be zero or negative, this method (concrete score matching) suffers in producing good performance. Lou et al. [18] recently proposed a score matching network with Bregman divergence that restricts the density rations to only positive values and empirically performs well for discrete data (text). We employ a similar method with a different function for Bregman divergence that ultimately leads to the same formulation as shown in Lou et al. [18]. Hence, we follow the similar forward and backward transition ideas tweaked for genomic sequences, which are inherently different from the text and require very specific modifications, as explained in the next section.

Simplex Based Models. Some of the previous diffusion-based approaches were designed for modeling discrete data residing at vertices of a probability simplex. Richemond et al. [87], Floto et al. [88] proposed methods that start from a Dirichlet prior over whole simplex and converge to a Dirichlet distribution with densities accumulated to one of the vertices of the simplex. Dirichlet diffusion score matching (DDSM) [14] follows the Jacobi diffusion process with a stick-breaking transform that converges to a Dirichlet distribution. Dirichlet flow matching (DFM) [17] designed a simplex flow method that transports data points residing in the simplex to the vertices with the points constrained inside the simplex at every intermediate step.

Continuous Diffusion Approaches. Another class of discrete diffusion approaches modeled discrete data without restricting them as discrete points and brought them into continuous space by adding Gaussian noise with the stationary distribution as Gaussian distribution [89, 70, 90, 16]. Dieleman et al. [91], Li et al. [15] proposed moving the continuous diffusion process to the latent space and considering another set of encoder-decoder networks for data space to latent space forward and backward transfer. While these methods are very similar to standard continuous diffusion processes, approaches explicitly designed for discrete data work better in practice.

D Appendix: Experiment details

D.1 Diffusion Details

We provide all our code at `https://anonymous.4open.science/r/D3-DNA-Discrete-Diffusion-3E42/` for reference to the reviewers. Please note that the training and sampling code is primarily influenced by the code shared with [18] with changes required for DNA sequence modeling.

Method Details. We follow the diffusion training closely from [18] with the geometric noise distribution $\bar{\beta}(t) = \beta_{min}^{1-t}\beta_{max}^{t}$, where $\bar{\beta}$ moves from $\beta_{min} = 0$ ($1e^{-3}$ to be specific) to $\beta_{max} = 1$ with increasing time step t. The noise schedules are chosen such that the prior loss $\mathcal{D}_{KL}(q_{T|0}(.|x_0)||p_{ref}$ are negligible and $q_{\bar{\beta}(0)}$ remains close to q_{data}. The uniform transition matrix is scaled down by $\frac{1}{N}$ ($N = 4$ here), and we consider p_{ref} as uniform.

Convolution Model Details. We use the same architecture for D3-Conv as the architecture used in Stark et al. [17] and Avdeyev et al. [14] for human promoter sequences. In place of the time step embedding with Gaussian random feature projection, we use similar embedding to parameterize the network with the total noise level instead of the time t. Here, the results (MSE) are compared with other previous methods, for which we referred to [17]. We follow a similar setup for DeepSTARR and MPRA datasets, too, for D3-Conv. As DFM [17] and other previous methods did not experiment with these two datasets, we trained the DFM-conv model (with a similar setup provided for the promoter) for proper performance comparison. Please note that we only considered DFM for comparison as their results are closest to D3-conv.

Transformer Model Details. We follow the similar transformer architecture for D3-Tran as used for SEDD small [18]. We used flash attention[92] with fused kernels and adaLN-zero time information network from [93] with 128 hidden dimensions, which is based on incorporating time conditioning into a standard encoder-only transformer architecture[94, 40]. A small change on this setup is adding rotary positional embedding [95] for performance improvement. D3-Tran has 12 transformer blocks and 12 attention heads per block. Training D3-Tran with this transformer architecture yielded even better performance than D3-Conv for all three datasets, i.e., promoter, DeepSTARR, and MPRA, as explained in Section5. Surprisingly, the performance of DFM with the transformer degraded more than its convolution counterpart, and this trend remained the same for all three datasets. We do not rule out the possibility of some minor architectural or methodical change to improve the performance of DFM for transformers, which may be explored in the near future.

Toward a more complete performance comparison of D3 with DFM, we trained DFM with the same transformer architecture with the only change of time step embedding (with Gaussian random feature projection) instead of embedding on noise level as already mentioned for convolution model details. For a fair comparison, we did not change any other part of the transformer architecture. We followed a similar overall training setup to train DFM as explained in [17].

Training Details. All of our models were trained with a learning rate 3×10^{-4}, a batch size of 128 for convolution architecture and 256 for transformer architecture. Following [18], our gradient norm is clipped to 1 and a linear warmup schedule for the first 2000 iterations. We set the training for 500000 steps with a validation step after every 50000 training step. The best models were found after different training steps for different datasets and methods. For example, DFM yielded the best model after 345730 steps, which is 400000 steps for D3 for convolution architecture with DeepSTARR. We employ an exponential moving average of 0.9999 for all our D3-Conv and D3-Tran training. All the training was performed on a single node with 4 A100 80GB GPUs.

Hyperparameter Search. As mentioned in Lou et al. [18], no hyperparameter or architecture search was performed for our models. The convolution architecture was chosen from the previous works [14, 17] and the transformer architecture from DDiT[93], with the rotary embedding as was used in [96]. The learning rate of 3×10^{-4} and the EMA of 0.9999 were selected as a standard choice from the previous well-known diffusion training recipes.

Conditioning Details. As mentioned in the main paper, we consider datasets with continuous-valued vectors as conditioning signals. Therefore, we send these signals with the true sequences to the score function, which is trained to learn density ratios based on the signals. For the promoter dataset, we have transcription initiation signal profiles, a vector of the same length as the sequences, i.e., 1024. Here, we concat (similar functionality as *torch.cat* for PyTorch framework) the signal profiles with the sequences before forwarding to the score function. As the fly enhancer sequences have corresponding activity values in 2 contexts (housekeeping and developmental gene promoters), we employ a 2-layered fully connected network that outputs an embedding (of the same length as sequence, i.e., 249) from the 2 activity values. Then, we concat the embeddings with the sequences and follow the same training and sampling process. For cell-type specific regulatory sequences, there are three cell-type specific activity levels. Hence, we use a similar 2-layered fully connected network to find an embedding (of length 200 to match the sequence length) that we concat to the sequences.

D.2 Oracle details

Human promoters. The oracle was Sei [71], a multi-task CNN that predicts 21,907 chromatin profiles across >1,300 cell lines/tissues from DNA sequences. Active promoter predictions were obtained from the H3K4me3 chromatin mark prediction. The Sei oracle and weights were acquired from the DDSM repository (`https://github.com/jzhoulab/ddsm`).

Fly enhancers. The DeepSTARR dataset [68] is a multi-task regression problem that predicts enhancer activity for two promoters, representing a developmental and housekeeping gene in *D. melanogaster* S2 cells. The input consists of genomic sequences with a length of 249 nucleotides, and the output comprises two scalar values indicating the activity of developmental and housekeeping enhancers, experimentally measured using STARR-seq. Sequences containing N characters were removed, which had a minimal impact on the dataset size (reducing it by approximately 0.005%). The dataset [72] was analyzed using the original DeepSTARR model architecture, described as follows:

- Input
- 1D convolution (256 filters, size 7, stride 1)
- BatchNorm + ReLU activation
- Max-pooling (size 2, stride 2)
- 1D convolution (60 filters, size 3, stride 1)
- BatchNorm + ReLU activation
- Max-pooling (size 2, stride 2)
- 1D convolution (60 filters, size 5, stride 1)
- BatchNorm + ReLU activation
- Max-pooling (size 2, stride 2)
- 1D convolution (120 filters, size 3, stride 1)
- BatchNorm + ReLU activation
- Max-pooling (size 2, stride 2)
- Fully-connected layer (256 units)
- BatchNorm + ReLU activation
- Dropout (0.4)
- Fully-connected layer (256 units)
- BatchNorm + ReLU activation
- Dropout (0.4)
- Fully-connected output layer (2 units, linear activation)

The DeepSTARR model was trained using EvoAug, an evolution-inspired data augmentations that was shown to lead to SOTA performance. We acquired weights from `DeepSTARR_combo1_finetune_2.ckpt` located on `https://zenodo.org/records/7767325`. The augmentation hyperparameters for this model were set to:

- mutation: mutate_frac = 0.05

- translocation: shift_min = 0, shift_max = 20
- insertion: insert_min = 0, insert_max = 20
- deletion: delete_min = 0, delete_max = 30
- reverse-complement: rc_prob = 0
- noise: noise_mean = 0, noise_std = 0.3

with a maximum number of augmentations set to 2. Dataset was acquired from `https://zenodo.org/records/7265991`.

Cell-type-specific cis-regulatory sequences. The MPRA dataset was curated by [33], which is a compilation of several MPRA experiments for K562, HepG2, and SK-N-SH cell lines from ENCODE [97]. We acquired and processed the data from Supplementary Table 2 of [33]. The input data are 200 nt genomic sequences with reference or alternate alleles for genomic variants, and the target is a scalar value (for each cell line) representing experimentally measured activities. Sequences from chromosomes 7 and 13 were used for testing (63,958 total), chromosomes 19, 21, and X were for validation (59,697 total), and all of the other sequences were used for training (640,029 total).

Our custom model is a variation of ResidualBind [26, 73], consisting of convolutional layers, residual blocks, and dense layers. The model takes an input with a specified shape and applies the following layers:

- Input layer
- Conv1D(196, 15) - BatchNorm - Activation('exponential') - Dropout(0.1)
- Residual Block(196, 3, 'relu', num_dilation_factors=4)
- Dropout(0.2) - MaxPooling1D(4)
- Conv1D(256, 5) - BatchNorm - Activation('relu') - Dropout(0.1)
- Residual Block(256, 3, 'relu', num_dilation_factors=3)
- MaxPooling1D(4) - Dropout(0.2)
- Conv1D(256, 3) - BatchNorm - Activation('relu') - Dropout(0.1)
- Residual Block(256, 3, 'relu', num_dilation_factors=1)
- MaxPooling1D(2) - Dropout(0.2)
- Flatten()
- Dense(512) - BatchNorm - Activation('relu') - Dropout(0.5)
- Dense(256) - BatchNorm - Activation('relu') - Dropout(0.5)
- Dense(1, activation='linear')

The model uses exponential activations in the first layer as it has been shown to lead to more interpretable convolutional kernels and improved efficacy with attribution analysis [98]. The residual block consists of a series of convolutional layers with varying dilation rates, batch normalization, and activation functions, according to:

- Input layer
- Conv1D(num_filters, filter_size, dilation_rate=1) - BatchNorm - Activation('relu')
- Dropout(0.1)
- For each dilation factor(1, 2, 4, 8)[:num_dilation_factor]:
  - Conv1D(num_filters, filter_size, dilation_rate=factor) - BatchNorm - Activation('relu')
- Add input layer and the output of the last convolutional layer
- Activation('relu')

where the number of dilation factors subsamples the dilation factor list, the output of the residual block is then added to the input layer, creating a residual connection.

ResidualBind was trained for 100 epochs using the Adam optimizer [99] with an initial learning rate of 1×10^{-3}. The learning rate was decayed by a factor of 0.2 when the validation loss did not improve for five epochs. Early stopping with a patience of 10 epochs was used. The model checkpoint with the lowest validation loss was chosen for downstream analysis. A separate model was trained for each cell line. Notably, the performance of ResidualBind for K562 (Pearson's r = 0.839), HepG2 (Pearson's r = 0.839), and SK-N-SH (Pearson's r = 0.827) was higher than what was achieved by the models explored in [100]. Nevertheless, nuances in data processing and training make direct comparisons not straightforward (even with the same data splits).

E Appendix: Supporting Figures and Tables

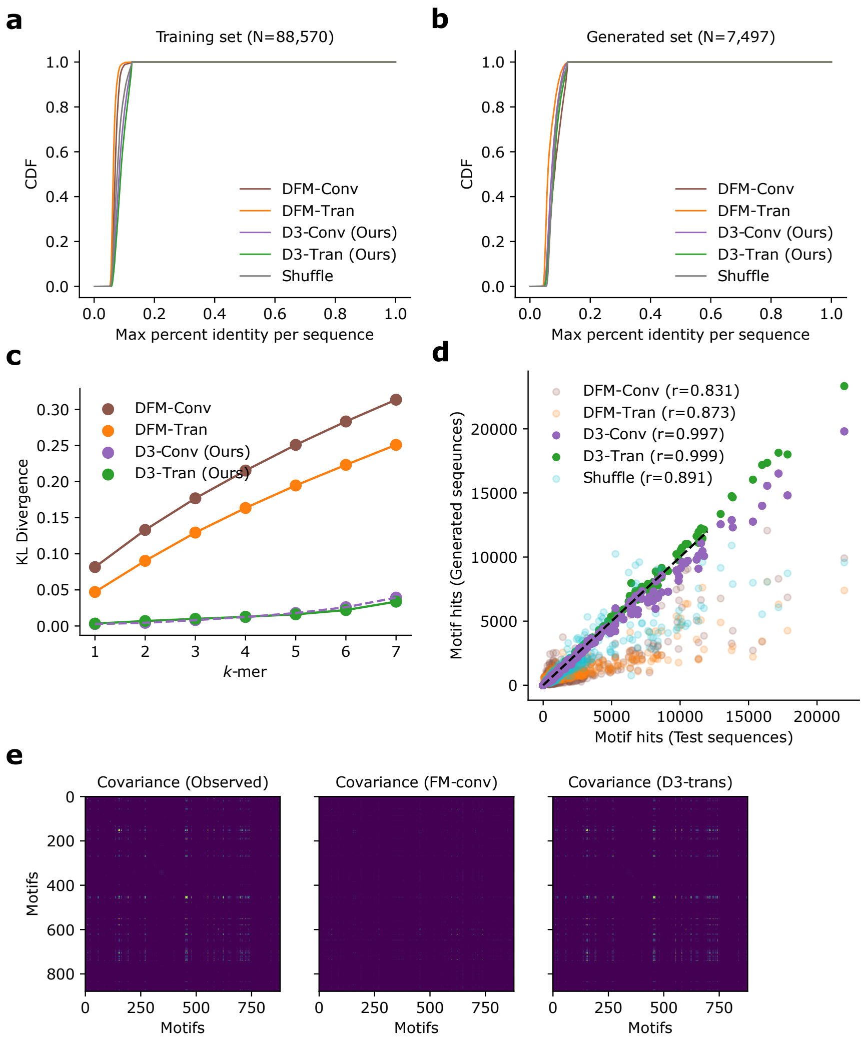

Figure 3: Sequence evaluation for human promoters.The cumulative distribution function of the maximum percent identity for each generated sequence against the entire training set (**a**) to assess memorization and among the generated set (**b**) to assess sequence diversity. Different methods are shown in different colors and comprise analysis for 7,497 sequences in each curve. Shuffle represents dinucleotide shuffled test sequences. **c** Shows the k-mer spectrum shift for each method according to the KL divergence for various k-mer values. D3-Conv is shown in dashed lines to see the separation from D3-Tran visually. **d** Scatter plot of the JASPAR motif hits using FIMO for generated sequences versus test sequences. Each dot represents the total enrichment of a different motif across the entire set. Pearson's r is shown in the legend. **e** the observed motif-motif covariance matrix (based on motif hits per sequence) is shown for test sequences (observed), FM-conv, and D3-trans.

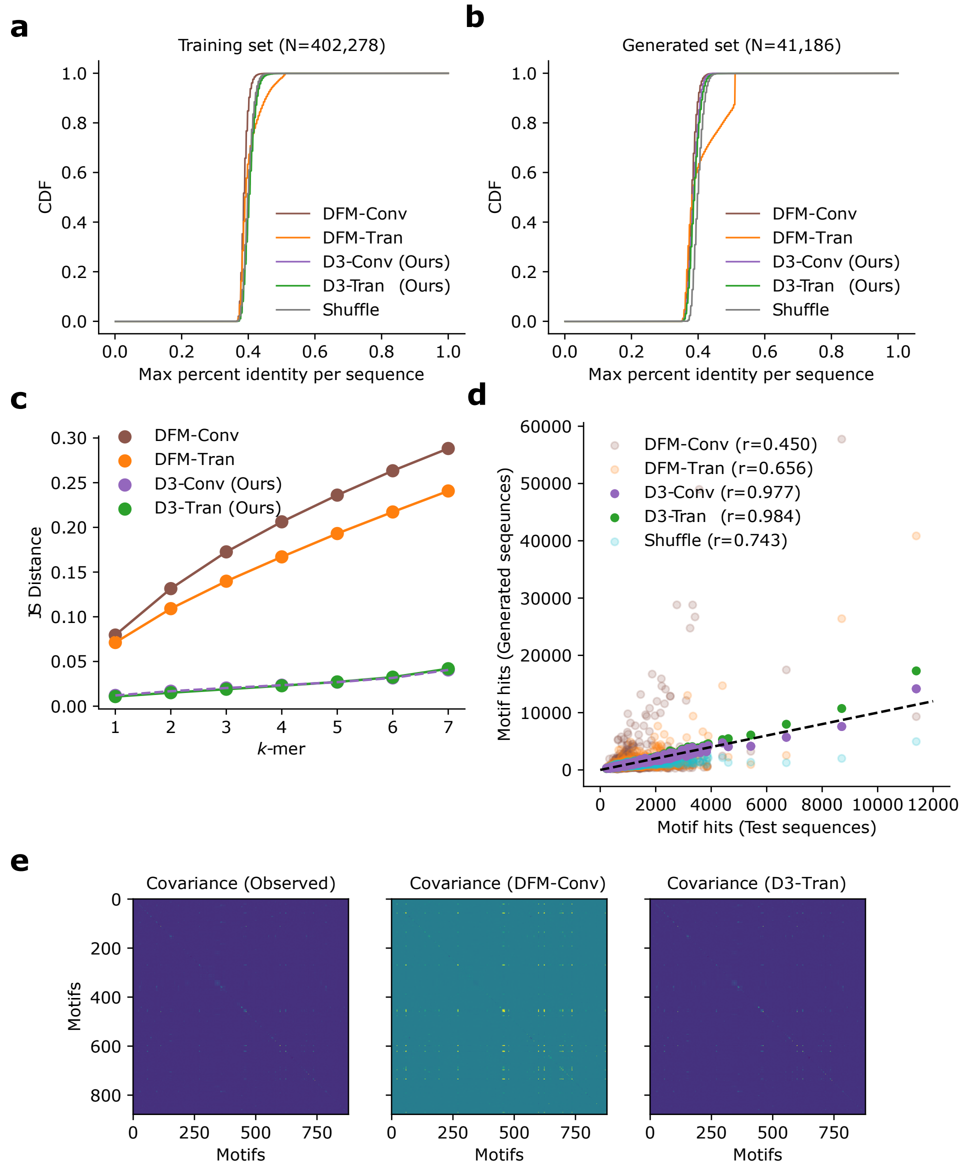

Figure 4: Sequence evaluation for fly enhancers. The cumulative distribution function of the maximum percent identity for each generated sequence against the entire training set (**a**) to assess memorization and among the generated set (**b**) to assess sequence diversity. Different methods are shown in different colors and comprise analysis for 7,497 sequences in each curve. Shuffle represents dinucleotide shuffled test sequences. **c** Shows the k-mer spectrum shift for each method according to the KL divergence for various k-mer values. D3-Conv is shown in dashed lines to show the separation from D3-Tran visually. **d** Scatter plot of the JASPAR motif hits using FIMO for generated sequences versus test sequences. Each dot represents the total enrichment of a different motif across the entire set. Pearson's r is shown in the legend. **e** the observed motif-motif covariance matrix (based on motif hits per sequence) is shown for test sequences (observed), FM-conv, and D3-trans.

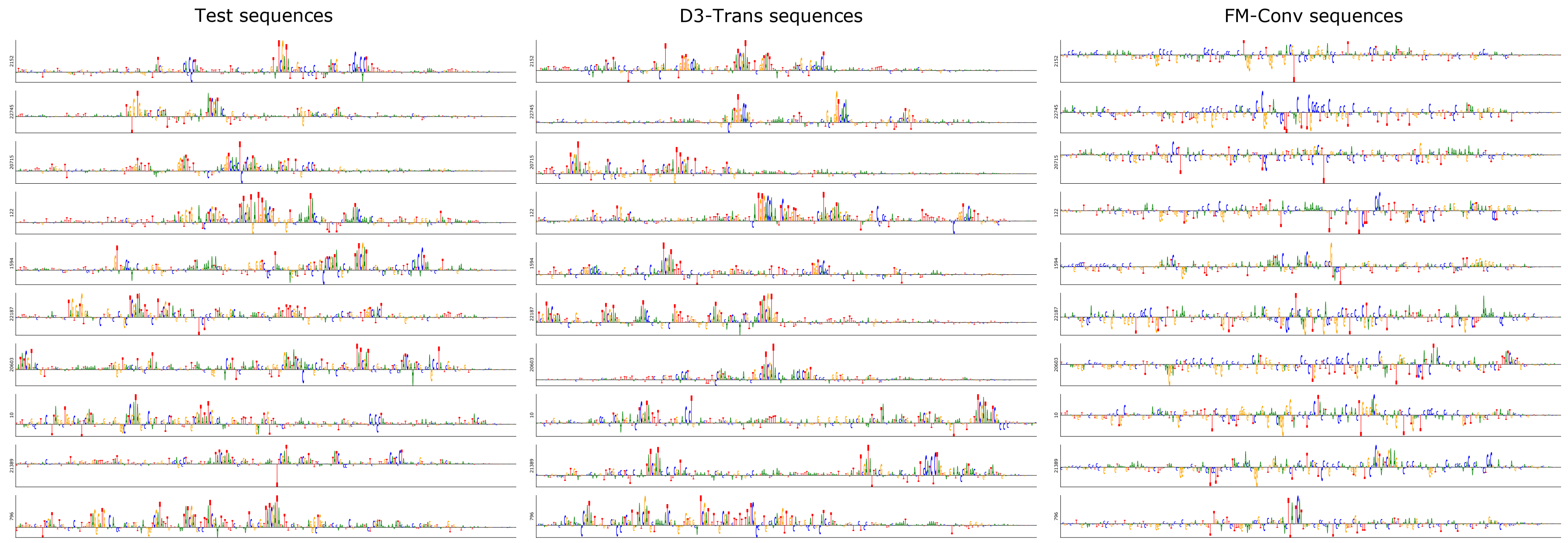

Figure 5: Attribution maps for test sequences (left), D3-Tran-generated sequences (middle), and DFM-Conv-generated sequences (right), with each row representing matched conditional activities based on the test sequence. Test and D3-Tran-generated sequences show contiguous motif-like patterns, while DFM-Conv-generated sequences have more spurious attribution noise. Attribution maps were calculated using GradientSHAP from Captum with an EvoAug-trained DeepSTARR oracle and visualized using Logomaker.

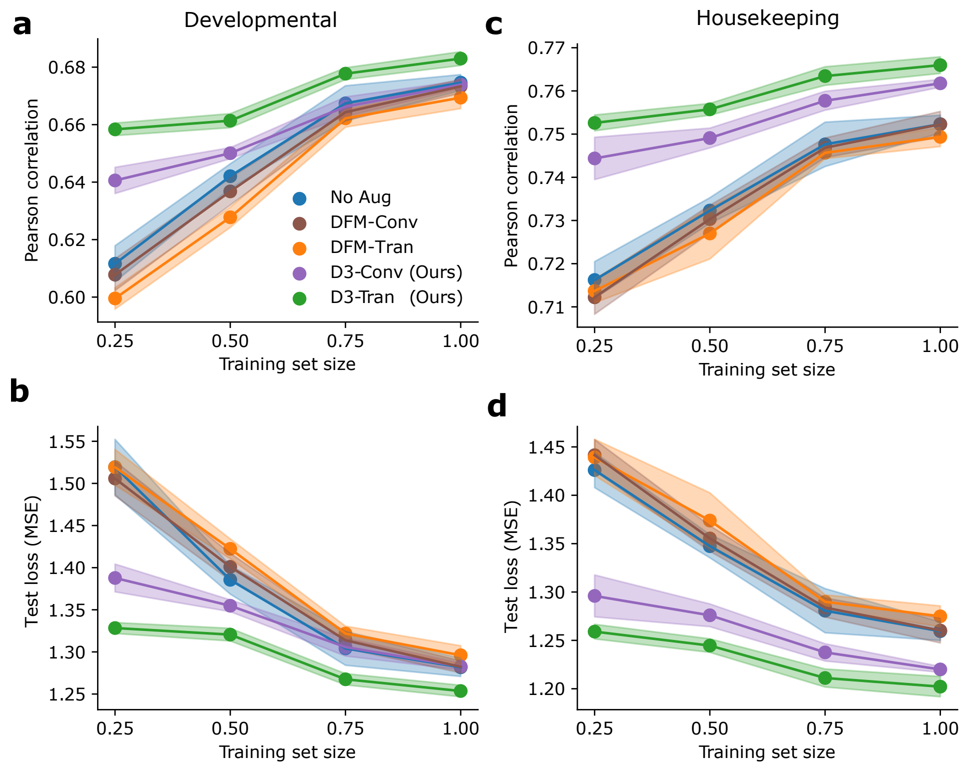

Figure 6: Augmentation analysis for synthetic fly enhancers. The performance of training a DeepSTARR model using different downsampled sets of training sequences (No Aug) and with the addition of synthetic sequences generated conditionally by different methods to match the activity of the full training set (shown in a different color) according to the Pearson correlation for developmental (**a**) and housekeeping (**c**) contexts and the test loss for developmental (**b**) and housekeeping (**d**) contexts. Plots display average values; shaded regions represent the standard deviation across five trials, each with a random initialization.

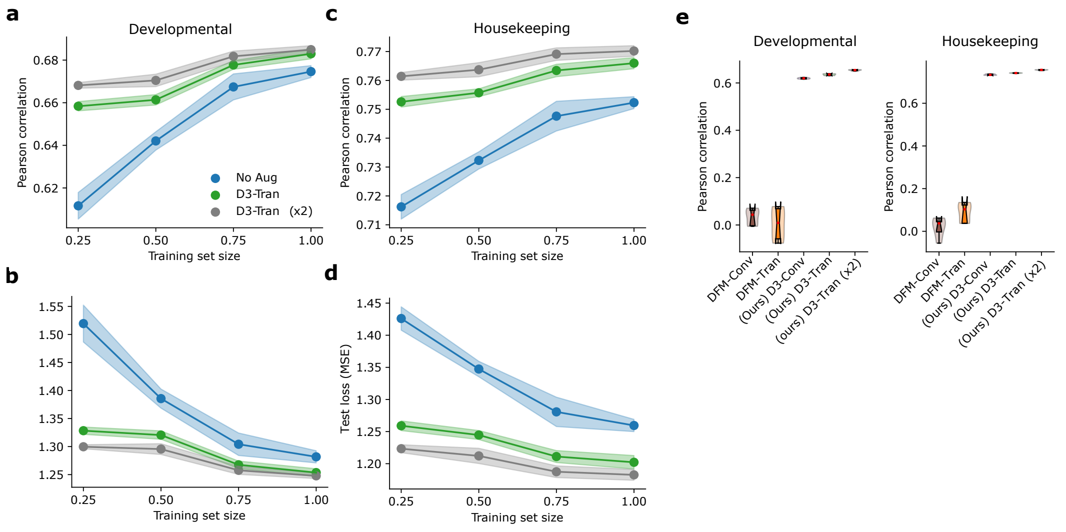

Figure 7: Augmentation analysis for synthetic fly enhancers. The performance of training a DeepSTARR model using different downsampled sets of training sequences (No Aug) and with the addition of synthetic sequences generated conditionally to match the activity of the full training set by D3-Tran with different multiples of the generated training set (shown in a different color) according to the Pearson correlation for developmental (**a**) and housekeeping (**c**) contexts and the test loss for developmental (**b**) and housekeeping (**d**) contexts. Plots display average values; shaded regions represent the standard deviation across five trials, each with a random initialization. **e** Boxplot of Pearson correlation for DeepSTARR models trained using only synthetic sequences from various methods, evaluated on test sequences.

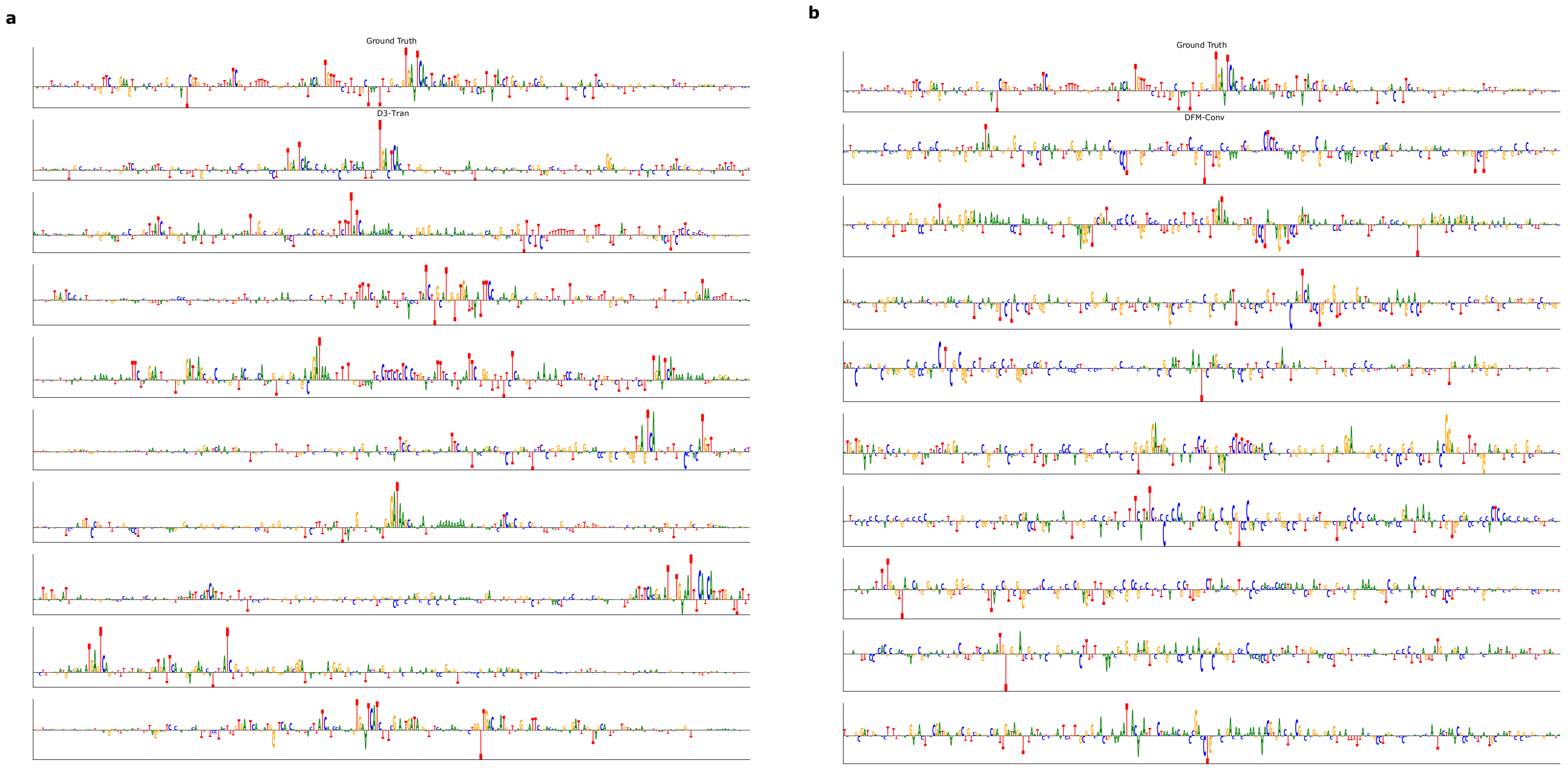

Figure 8: Attribution maps of generated sequences using the same conditional values for D3-Tran (**a**) and DFM-Conv (**b**). D3-Tran-generated sequences show contiguous motif-like patterns, while DFM-Conv-generated sequences have more spurious attribution noise. An attribution map for the wild type sequence is shown at the top for reference. Attribution maps were calculated using GradientSHAP from Captum with an EvoAug-trained DeepSTARR oracle and visualized using Logomaker. Importantly, the attribution maps in each row show very different underlying motif patterns, suggesting a different cis-regulatory mechanism that drives comparable activity levels.

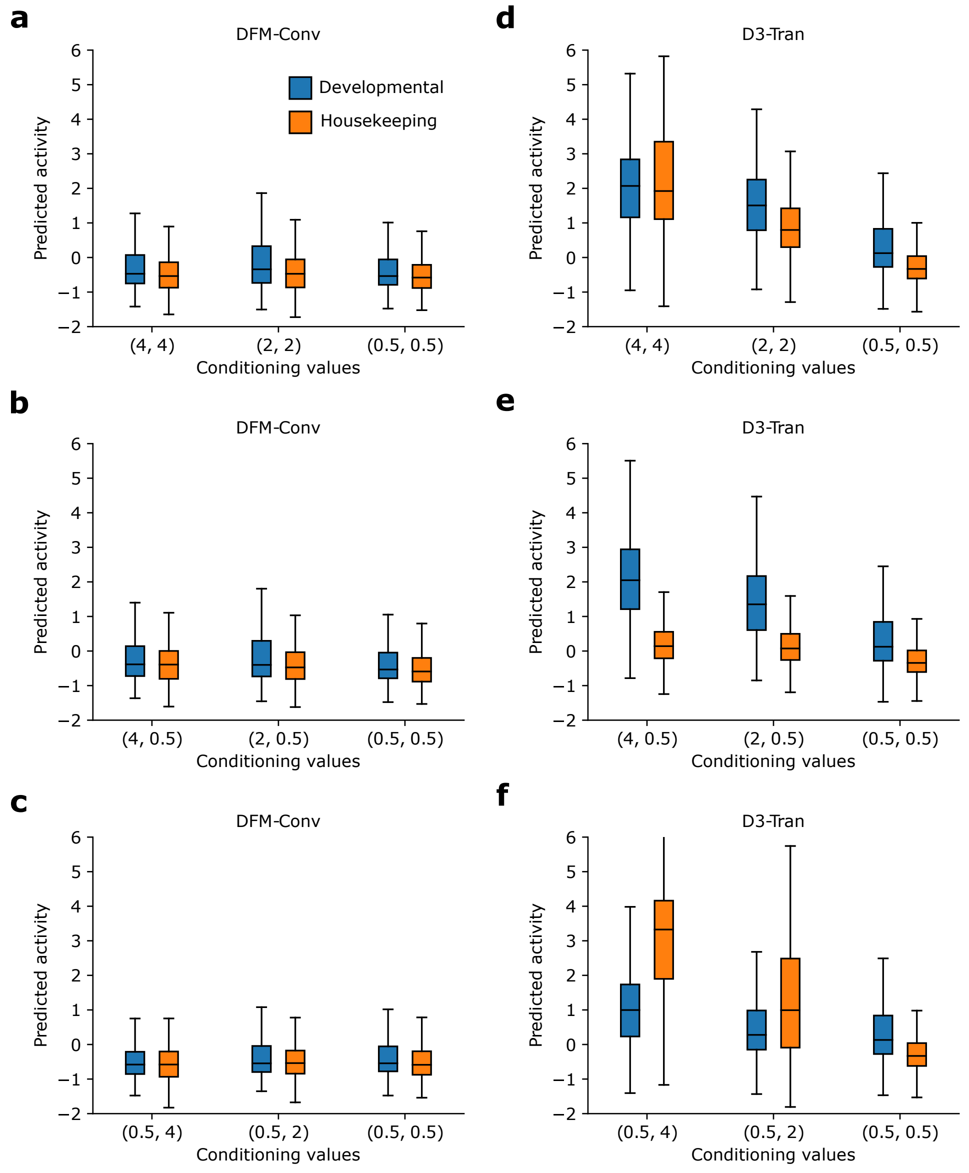

Figure 9: Task-specific design for fly enhancers. Comparison of predicted activities by EvoAug-trained DeepSTARR model for generated sequences conditioned on task-specific activities for DFM-Conv (left) and D3-Tran (right). Task activity values used for conditioning are shown on the x-ticks as (activity for developmental tasks and activity for housekeeping tasks). Each boxplot represents the task-specific predictions for 500 generated sequences. (**a**,**d**) Results when conditional activities are similar across cell types. (**b**,**c**,**e**,**f**) Results with differential conditional activities. D3-Tran generates sequences whose predicted activity more closely tracks the conditioning value trends, albeit the magnitude of the high activities is not as precise.

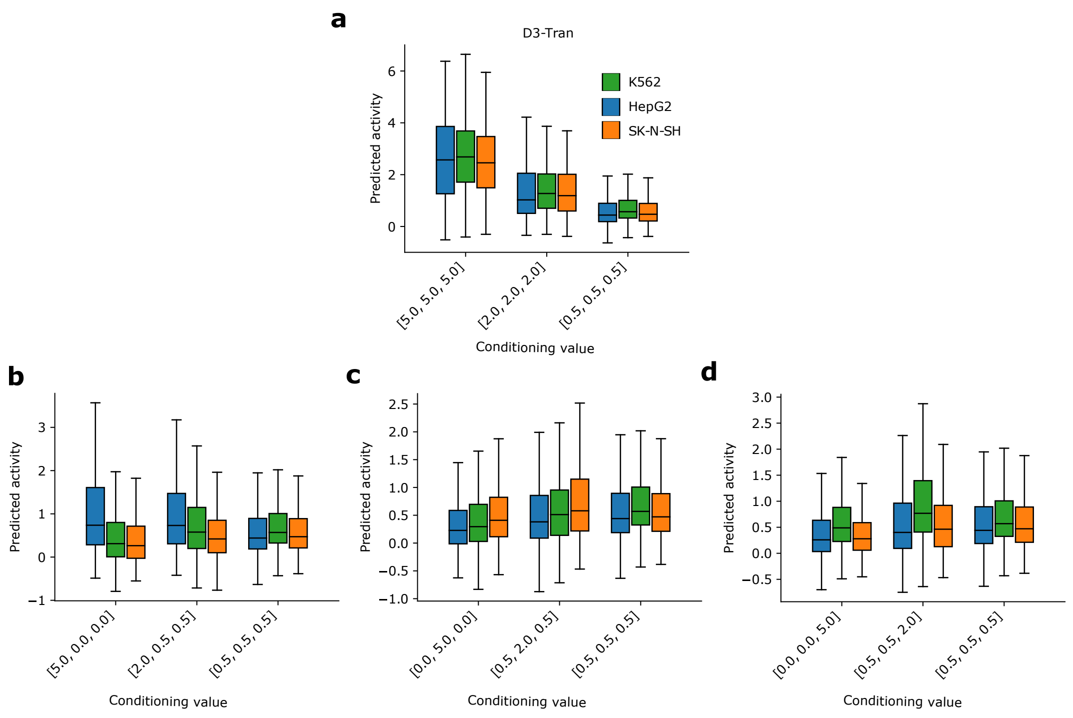

Figure 10: Task-specific design for cell-type-specific cis-regulatory sequences from MPRA data. Boxplots show oracle-predicted activities for sequences generated by D3-Tran, conditioned on values indicated by (K562 activity, HepG2 activity, SK-N-SH activity) on the x-axis. Each boxplot represents the corresponding cell type activities from 500 generated sequences. (**a**) Results when conditional activities are similar across cell types. (**b**-**d**) Results with differential conditional activities.

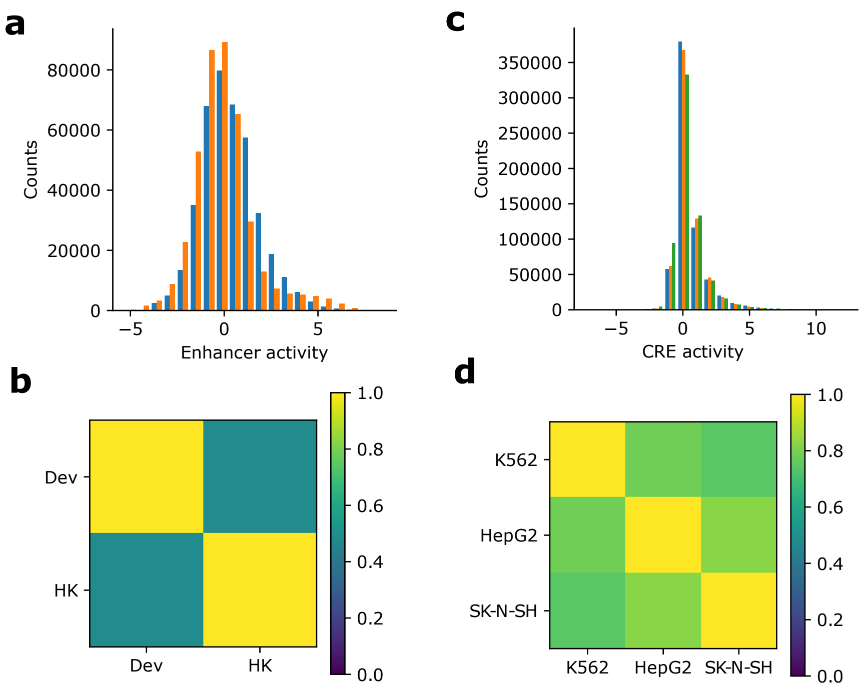

Figure 11: Label statistics for fly enhancer and cell-type-specific MPRA data. Histogram of label activities for each task within fly enhancers dataset (**a**) and cell-type-specific regulatory sequences (**c**). Correlation matrix of target activities of the training data for fly enhancers (**b**) and cell-type-specific regulatory sequences (**d**).

Table 3: Results for cell-type-specific generation from MPRA data.

Model	Conditional generation fidelity (MSE)
DFM-Conv	2.243
DFM-Tran	1.620
D3-Conv	0.937
D3-Tran	0.929
Shuffle	1.824

Table 4: Conditional sampling diversity for fly enhancers. 500 sequences were generated using a fixed conditional value of $[7.045,\ 1.973]$ for [developmental task, housekeeping task] (from test sequence index=21,106) and assessed according to sequence diversity and mechanistic diversity.

Model	Sequence Diversity (percent identity)	Mechanistic diversity (Euclidean distance)
DFM-Conv	23.9	0.428
DFM-Tran	25.0	0.425
3D-Conv	26.6	0.456
3D-Tran	26.5	0.472
Shuffle	29.5	0.436