Absorb & Escape: Overcoming Single Model Limitations in Generating Genomic Sequences

Zehui Li[1,*], Yuhao Ni[1], Guoxuan Xia[1], William Beardall[1], Akashaditya Das[1], Guy-Bart Stan[1,*], Yiren Zhao[1,*]
[1]Imperial College London, {zehui.li22, harry.ni21, g.xia21, william.beardall15, akashaditya.das13, g.stan, a.zhao}@imperial.ac.uk
[*]Correspondence: {zehui.li22, g.stan, a.zhao}@imperial.ac.uk

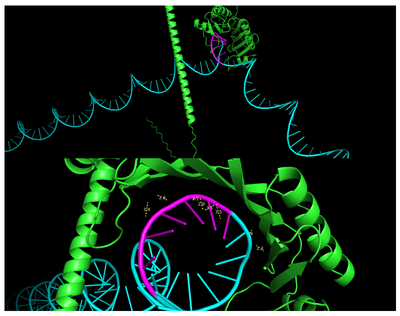
Abstract

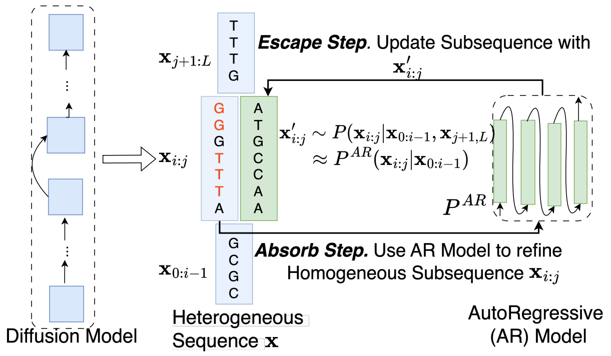
Recent advances in immunology and synthetic biology have accelerated the development of deep generative methods for DNA sequence design. Two dominant approaches in this field are AutoRegressive (AR) models and Diffusion Models (DMs). However, genomic sequences are functionally heterogeneous, consisting of multiple connected regions (e.g., Promoter Regions, Exons, and Introns) where elements within each region come from the same probability distribution, but the overall sequence is non-homogeneous. This heterogeneous nature presents challenges for a single model to accurately generate genomic sequences. In this paper, we analyze the properties of AR models and DMs in heterogeneous genomic sequence generation, pointing out crucial limitations in both methods: (i) AR models capture the underlying distribution of data by factorizing and learning the transition probability but fail to capture the global property of DNA sequences. (ii) DMs learn to recover the global distribution but tend to produce errors at the base pair level. To overcome the limitations of both approaches, we propose a post-training sampling method, termed **A**bsorb **&** **E**scape (A&E) to perform compositional generation from AR models and DMs. This approach starts with samples generated by DMs and refines the sample quality using an AR model through the alternation of the Absorb and Escape steps. To assess the quality of generated sequences, we conduct extensive experiments on 15 species for conditional and unconditional DNA generation. The experiment results from motif distribution, diversity checks, and genome integration tests unequivocally show that A&E outperforms state-of-the-art AR models and DMs in genomic sequence generation. A&E does not suffer from the slowness of traditional MCMC to sample from composed distributions with Energy-Based Models whilst it obtains higher quality samples than single models. Our research sheds light on the limitations of current single-model approaches in DNA generation and provides a simple but effective solution for heterogeneous sequence generation. Code is available at the Github Repo[1].

1 Introduction

DNA sequences, as the blueprint of life, encode proteins and RNAs and directly interact with these molecules to regulate biological activities within cells. The success of deep generative models in image [28], text [27], and protein design [35] has drawn the attention of deep learning researchers to the problem of DNA design, i.e. applying these models to genomic sequence generation [4, 34, 38]. However, one rarely explored issue is how well existing methods can handle the unique property of DNA sequences: heterogeneity. DNA sequences are highly heterogeneous, consisting of multiple

[1]https://github.com/Zehui127/Absorb-Escape


(a) **A DNA generated by A&E, interacting with TATA-binding protein.** The DNA sequences highlighted in magenta are the TATA-box motif. The confirmation is predicted by AlphaFold 3 [1]: the DNA bends at the TATA-box position.



Diffusion Model
$\mathbf{x}_{j+1:L}$
T T T G
$\mathbf{x}_{i:j}$
G G G T T T A
A T G C C A A
$\mathbf{x}_{0:i-1}$
G C G C
Escape Step. Update Subsequence with $\mathbf{x}'_{i:j}$
$\mathbf{x}'_{i:j} \sim P(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1}, \mathbf{x}_{j+1,L}) \approx P^{AR}(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1})$
P^{AR}
Absorb Step. Use AR Model to refine Homogeneous Subsequence $\mathbf{x}_{i:j}$
Heterogeneous Sequence $\mathbf{x}$
AutoRegressive (AR) Model

(b) The proposed framework Fast **A**bsorb **& E**scape (Fast A&E): The DM and AR models jointly optimize a given sequence by alternating between the A-step and the E-step. See Section 4 for a detailed explanation.

Figure 1: (a) Generated DNA interacting with TATA-binding protein. (b) Proposed A&E framework.

connected functional regions (e.g. Promoter Regions, Exons, and Introns) in sequential order. While elements within each functional region might be homogeneous (coming from the same distribution), the overall sequence is non-homogeneous. This heterogeneity, along with the discrete nature of genomic sequences, poses challenges to popular deep generative methods.

Limitations of Existing Single-Model Approaches in Generating Genomic Sequences AutoRegressive Models (AR) [8, 17, 24] are one of the most dominant approaches for discrete sequence generation. To model the data distribution of a sequence x of length T, the probability of x is factorized as:

$$p^{AR}(\mathbf{x}) = \prod_{i=1}^{T} p_\theta(x_i|x_1, x_2, \dots, x_{i-1}). \tag{1}$$

An issue arises when modeling heterogeneous data, where the value of θ may vary significantly from one segment to another. Additionally, AR models assume a dependency between the current element and previous elements; this assumption may not hold true for heterogeneous sequences, potentially hindering the learning process (see Section 3 for details).

On the other hand, Diffusion Models (DMs), initially proposed by [30], have been dominant in image generation. In the probabilistic denoising view [16], DMs gradually add noise to the input data x_0, and a reverse diffusion (generative) process is trained to gradually remove the noise from the perturbed data x_t. DMs directly model the data distribution without AutoRegressive factorization, thereby avoiding the issues associated with AR models. However, it has been shown that DMs are less competent than AR models for discrete data generation [21, 36]. When it comes to modeling heterogeneous genomic sequences, it remains unclear how the performance of DMs compares to AR models within each homogeneous segment.

Model Composition As a Solution Balancing the ability of generative algorithms to capture both local and global properties of the data distribution is central to the problem. An obvious solution could be to combine these two types of models and perform generation using the composed models. However, this typically requires converting these two models into an Energy-Based Model and then sampling using Markov Chain Monte Carlo (MCMC), which can be inherently slow due to the sampling nature of the algorithm [10], and the potential long inference time of individual models. With the goal of accurate and efficient DNA generation, we aim to investigate two key questions in this work: (i) How well does a single AR model or DM perform in DNA generation, given the heterogeneous nature of genomic sequences? (ii) Is there an efficient algorithm to combine the benefits of AR models and DMs, outperforming a single model? In answering these two questions, *our contribution is three-fold*:

(a) We study the properties of AR models and DMs in heterogeneous sequence generation through theoretical and empirical analysis (Section 3).

(b) We design the theoretical framework **A**bsorb **&** **E**scape (A&E) to sample from the compositional distribution of an AR model and a DM (Section 4.1). Furthermore, as shown in Figure 1b, we propose an efficient post-training sampling algorithm termed Fast A&E to sample from the composed model, requiring at most one forward pass through the pretrained DM and AR model (Section 4.2).

(c) We design a comprehensive evaluation workflow for DNA generation, assessing the sequence composition, diversity, and functional properties of generated genomic sequences. Extensive experiments (15 species, 6 recent DMs, 1 AR model, and 3 types of evaluations) reveal: 1) the limitations of existing models in DNA generation (Section 5.3), and 2) that the proposed algorithm Fast A&E consistently outperforms state-of-the-art models as measured by motif distribution and functional property similarity to natural DNA sequences (Sections 4.2 and 5.3).

2 Preliminaries and Related Work

2.1 Problem Formulation: DNA Generation

DNA generation aims to produce synthetic sequences that functionally approximate real DNA sequences. Formally, let $\mathbb{N}_4 = \{1, 2, 3, 4\}$, where each element represents one of the four nucleotides: adenine (A), thymine (T), guanine (G), and cytosine (C). A DNA sequence of length L can be represented as $\mathbf{x} \in \mathbb{N}_4^L$, with each element/nucleotide denoted by $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L$.

Unconditional Generation: Given a dataset of real-world DNA sequences $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^N$ collected from some distribution $p(\mathbf{x})$, where each sequence $\mathbf{x}^{(n)} \in \mathbb{N}_4^L$ represents a chain of nucleotides, the objective is to develop a generative model $p_{\boldsymbol{\theta}}(\mathbf{x})$ of the data distribution $p(\mathbf{x})$ from which we can sample novel sequences $\tilde{\mathbf{x}} \sim p_{\boldsymbol{\theta}}(\mathbf{x})$. These sequences should be structured arrangements of A, T, G, and C, reflecting the complex patterns found in actual DNA. Earlier works applying Generative Adversarial Networks (GANs) [13] to generate protein-encoding sequences [14] and functional elements [33, 34, 38] fall into this category.

Conditional Generation: In this task, the dataset of DNA sequences $\mathcal{X} = \{\mathbf{x}^{(n)}, c^{(n)}\}_{n=1}^N$ is sampled from the joint distribution $p(\mathbf{x}, c)$, where c represents the condition associated with each sequence. The objective is to develop a model $p_{\boldsymbol{\theta}}(\mathbf{x}|c)$ that generates new DNA sequences $\tilde{\mathbf{x}}$ given condition c. Recently, a discrete diffusion model DDSM [4] and an AutoRegressive model RegML [20] have used expression level as the condition, while DNADiffusion [26] has used cell type as the condition.

2.2 Homogeneous vs. Heterogeneous Sequences

Homogeneous Generation Process In the context of sequence generation, a homogeneous Markov Chain is characterized by constant probabilistic rules for generating the sequence at each time step t. More generally, a process is defined as **homogeneous** if the transition probabilities are independent of time t. This means there exists a constant $P_{\mathbf{c},j}$ such that:

$$P_{\mathbf{c},j} = \Pr[\mathbf{x}_t = j \mid \mathbf{x}_{1:t-1} = \mathbf{c}] \tag{2}$$

holds for all times t, where $P_{\mathbf{c},j}$ is a constant, and $\mathbf{c}$ represents a specific sequence of past values, i.e., $\mathbf{c} = (c_1, c_2, \ldots, c_{t-1})$.

Heterogeneous Generation Process Assuming homogeneous properties simplifies modeling but can be overly restrictive for certain modalities, leading to inaccuracies. For example, DNA sequences consist of various functionally distinct regions, such as promoters, enhancers, regulatory regions, and protein-coding regions scattered across the genome [11]. Each region may be assumed to be homogeneous, but the overall sequence is non-homogeneous due to the differing properties of these elements.

For sequences like DNA, which consist of locally homogeneous segments, we define them as **heterogeneous sequences**. Formally, a heterogeneous sequence is defined as follows: Suppose a sequence $\mathbf{x}$ is divided into segments $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$, where each segment $\mathcal{S}_i$ is homogeneous. For each segment $\mathcal{S}_i = (\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots, \mathbf{x}_{t-k})$, there exists a constant $P_{\mathbf{c}_i,j}$ such that:

$$P_{\mathbf{c}_i,j} = \Pr[\mathbf{x}_t = j \mid \mathbf{x}_{t-k:t-1} = \mathbf{c}_i] \tag{3}$$

holds for all times t within segment $\mathcal{S}_i$, where $P_{\mathbf{c}_i,j}$ is a constant, and $\mathbf{c}_i = (c_{i,t-1}, c_{i,t-2}, \ldots, c_{i,t-k})$ represents values characterizing the history within that segment. While segment $\mathcal{S}_i$ is homogeneous, the entire sequence is non-homogeneous due to the varying properties across different segments.

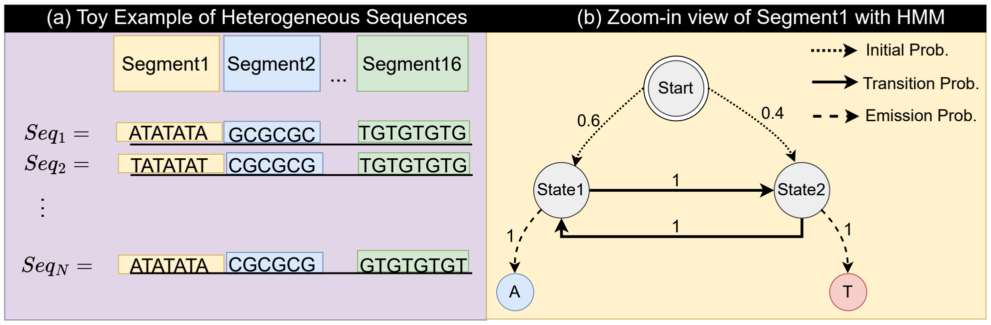

(a) Toy Example of Heterogeneous Sequences
Segment1
Segment2
...
Segment16
$Seq_1 =$ ATATATA GCGCGC TGTGTGTG
$Seq_2 =$ TATATAT CGCGCG TGTGTGTG
$Seq_N =$ ATATATA CGCGCG GTGTGTGT
(b) Zoom-in view of Segment1 with HMM
Start
0.6
0.4
State1
State2
1
1
1
1
A
T
Initial Prob.
Transition Prob.
Emission Prob.

Figure 2: A toy example with heterogeneous sequences: (a) The overall training set consists of $N = 50,000$ heterogeneous sequences, where each sequence further consists of 16 homogeneous segments. We apply an autoregressive and a diffusion model to learn the underlying distribution. (b) Within each segment, the sequences are generated with a simple Hidden Markov Model (HMM), with deterministic transition probability and emission probability.

3 Single Model Limitations in Heterogeneous Sequence Generation

How powerful are AR models and DMs in modelling heterogeneous sequences? We first provide a theoretical analysis, and then perform experiments on synthetic sequences to validate our assumption.

AutoRegressive (AR) Models Suppose a heterogeneous sequence $\mathbf{x}$ consist of two homogeneous segments of length k, then $\mathbf{x} = \{\{x_1, x_2, \cdots, x_k\}, \{x_{k+1}, x_{k+2}, \cdots, x_{2k}\}\}$. AR models factorize $p(\mathbf{x})$ into conditional probability in eq. (4); consider the case where the true factorisation of $p(x)$ follows eq. (5).

$$p^{AR}(\mathbf{x}) = p_\theta(x_1)p_\theta(x_2|x_1)\cdots p_\theta(x_k|\mathbf{x}_{1:k-1}) \cdot p_\theta(x_{k+1}|\mathbf{x}_{1:k})p_\theta(x_{k+2}|\mathbf{x}_{1:k+1})\cdots p_\theta(x_{2k}|\mathbf{x}_{1:2k-1}) \tag{4}$$

$$p^{data}(\mathbf{x}) = \underbrace{p_1(x_1)p_1(x_2|x_1)\cdots p_1(x_k|\mathbf{x}_{1:k-1})}_{\text{Segment 1}} \cdot \underbrace{p_2(x_{k+1})p_2(x_{k+2}|\mathbf{x}_{k+1})\cdots p_2(x_{2k}|\mathbf{x}_{k+1:2k-1})}_{\text{Segment 2}} \tag{5}$$

AR factorisation allows the accurate modelling of the first homogeneous segment; however, it may struggle to disassociate the elements of the second segment from the first segment. More precisely, sufficient data is needed for AR model to learn that $p_\theta(x_{k+1}), p_\theta(x_{k+2}), \cdots, p_\theta(x_{2k})$ should be independent to the elements $\{x_1, x_2, \cdots, x_k\}$ in the first segments. Secondly, when the context length of the AR model is shorter than the sequence length $2k$, it could struggle to capture the difference between p_1 and p_2 with a single set of parameters θ.

Diffusion Models (DMs) On the other hand, DMs estimate the overall probability distribution $p(\mathbf{x})$ without factorization. The elements of $\mathbf{x}$ are usually generated in parallel. Thus, they do not suffer from the conditional dependence assumption. However, the removal of the conditional dependence assumption may also decrease the accuracy of generation within each homogeneous segment compared to AR models, as DMs do not explicitly consider previous elements.

3.1 A Toy Example

To evaluate the performance of Autoregressive (AR) models and Diffusion Models (DMs) in generating heterogeneous sequences, we consider a toy example with 50,000 heterogeneous sequences $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{50000}$. Each sequence contains 16 segments, as illustrated in Figure 2(a), and each segment comprises 16 elements, resulting in a total sequence length of 256 ($\mathbf{x} \in \mathbb{N}_4^{256}$). A simple Hidden Markov Model (HMM) is used to generate each segment, as shown in Figure 2(b), with deterministic transition and emission probabilities that ensure homogeneity within each segment. The emitted tokens differ from one segment to another, mimicking the properties of real DNA sequences. Whilst it is possible to use more complex distributions for each segment, doing so could complicate the evaluation of the generated sequences.

Evaluation Under our toy HMM setup, a generative model could make two types of mistakes within each segment: **1) Illegal Start Token**: The generated sequence starts with a token which has zero emission probability. E.g. in Figure 2(b), the starting token could only be $\{A, T\}$. $\{G, C\}$ at the beginning of the sequence are classified as illegal start tokens. **2) Incorrect Transition**: The generated sequence contains tokens with zero transition probability. E.g. in Figure 2(b) given the start of the sequence is (A, T, A, T), the next token must be A, any other tokens such as $\{T, G, C\}$ are classified as incorrect transitions. We use the number of incorrect tokens as the metric for evaluation.

Experiment We use HyenaDNA [20, 24] as the representative autoregressive (AR) model. For the diffusion model, we develop a simple latent Discrete Diffusion model termed DiscDiff. It resembles the design of StableDiffusion [28], a latent diffusion model for image generation. DiscDiff consists of a CNN-based Variational Encoder-decoder, trained with cross entropy, to map the discrete DNA data into a latent space, and a standard 2-D UNet as the denoising network (detailed in appendix A). The training dataset consists of $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{50,000}$. For detailed training procedures see Appendix B. We generate 4,000 sequences from each model and present the evaluation results in Table 1. As expected, the diffusion model DiscDiff makes fewer errors regarding Illegal Start (IS) tokens but tends to generate more Incorrect Transition (IT) tokens. Conversely, while the AR model HyenaDNA generates some IS token errors, it produces significantly fewer IT token errors. This motivates the question: *can we combine the strengths of both algorithms to achieve better sequence generation?*

Table 1: **Number of Incorret Tokens on Synthetic Dataset**. The performance metrics used are the number of Illegal Start (IS) Tokens and Incorrect Transition (IT) Tokens. Note that there are a total of $4,000 \times 256 = 1024,000$ tokens.

	HyenaDNA	DiscDiff
# IS Tokens ↓	812	**0**
# IT Tokens ↓	**3,586**	110,192

4 Method

Algorithm 1 Absorb & Escape Algorithm

```
Require: Pretrained AutoRegressive model $p_\theta^{AR}(\mathbf{x})$ and pretrained Diffusion Model $p_\beta^{DM}(\mathbf{x})$
 1: Initialize $\tilde{\mathbf{x}}^0 \sim p_\beta^{DM}(\mathbf{x})$
 2: Set $t = 0$
 3: Assume $\mathbf{x} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$, where each $\mathbf{s}_k = \{\mathbf{x}_i, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_j\}$ is a segment
 4: while not converged do
 5:     Sample a segment $\mathcal{S} \in \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$
 6:     Set $i$ = start index of $\mathbf{s}_k$, $j$ = end index of $\mathbf{s}_k$
 7:     Absorb step:
 8:     $\tilde{\mathbf{x}}'_{i:j} \sim p(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1}, \mathbf{x}_{j+1:L}) \approx p_\theta^{AR}(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1})$ //Refine segment $\mathbf{s}_k$ using the AR model
 9:     Escape step:
10:     $\tilde{\mathbf{x}}^t_{i:j} = \tilde{\mathbf{x}}'_{i:j}$ //Update $\tilde{\mathbf{x}}^t$
11:     Increment $t = t + 1$
12: end while
13: Output: Final sample $\tilde{\mathbf{x}}^t$ with improved quality
```

Algorithm 2 Fast Absorb & Escape Algorithm

```
Require: Absorb Threshold $T_{Absorb}$, Pretrained AutoRegressive model $p_\theta^{AR}(\mathbf{x})$ and pretrained Diffusion Model $p_\beta^{DM}(\mathbf{x})$
 1: Initialize $\tilde{\mathbf{x}}^0 \sim p_\beta^{DM}(\mathbf{x})$
 2: for $i$ in $len(\tilde{\mathbf{x}})$ do
 3:     if $p^{DM} < T_{Absorb}$ then
 4:         Absorb step:
 5:         j = i+1
 6:         $\tilde{\mathbf{x}}'_j \sim p_\theta^{AR}(\mathbf{x}_j|\mathbf{x}_{0:i})$
 7:         while $p^{AR}(\tilde{\mathbf{x}}'_j) > p^{DM}(\tilde{\mathbf{x}}_j)$ do
 8:             Increment $j = j + 1$
 9:             $\tilde{\mathbf{x}}'_j \sim p_\theta^{AR}(\mathbf{x}_j|\mathbf{x}_{0:i}, \mathbf{x}_{i:j-1})$ //Refine Inaccurate region of the sequence token by token
10:         end while
11:         Escape step:
12:         $\tilde{\mathbf{x}}_{i:j} = \tilde{\mathbf{x}}'_{i:j}$ //Update $\tilde{\mathbf{x}}$
13:         Increment i = i + j
14:     end if
15: end for
16: Output: $\tilde{\mathbf{x}}$ with improved quality
```

4.1 The Absorb & Escape Framework

Given a pretrained AutoRegressive model $p_\theta^{AR}(\mathbf{x})$ and a Diffusion Model $p_\beta^{DM}(\mathbf{x})$, we aim to generate a higher quality example $\tilde{\mathbf{x}}$ from the composed distribution $p_{\theta,\beta}^{C}(\mathbf{x}) = p_\theta^{AR}(\mathbf{x}) \circ p_\beta^{DM}(\mathbf{x})$. However, directly computing $p_{\theta,\beta}^{C}(\mathbf{x})$ is generally intractable, as both the autoregressive factorizations from p^{AR} and score functions from p^{DM} are not directly composable [9, 10]. We propose the **A**bsorb **& E**scape (A&E) framework, as shown in Algorithm 1, to efficiently sample from $p_{\theta,\beta}^{C}(\mathbf{x})$.

Absorb-Step Inspired by Gibbs sampling [12], which iteratively refines each dimension of a single sample and moves to higher density areas, our algorithm starts with a sequence $\mathbf{x}^0 \sim p^{DM}(\mathbf{x})$, generated by the diffusion model and then refines the samples through the Absorb step and Escape step. By exploiting the heterogeneous nature of the sequence, we assume that $\mathbf{x}^0$ can be factorized into multiple segments $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$. For each segment $\mathbf{s}_k$, we set i and j as the start and end indices, respectively. During the Absorb step, we sample a subset of segments $\mathcal{S} \subseteq \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$ and refine each segment $\mathbf{s}_k$ by sampling $\tilde{\mathbf{x}}'_{i:j} \sim p(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1}, \mathbf{x}_{j+1:L}) \approx p_\theta^{AR}(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1})$, using the autoregressive model to approximate the conditional probability.

Escape-Step After refining the segment in the Absorb step, we proceed with the Escape step where we update the refined segment $\tilde{\mathbf{x}}^t_{i:j}$ to $\tilde{\mathbf{x}}'_{i:j}$. This iterative process continues for each selected segment $\mathbf{s}_k$, with t incrementing after each update. By leveraging the ability of the diffusion model to capture the overall data distribution and the autoregressive model to refine homogeneous sequences within each segment, our algorithm efficiently improves the quality of the generated samples. The final output $\tilde{\mathbf{x}}^t$ is hereby closer to the true data distribution $p(\mathbf{x})$ compared to the initial sample $\tilde{\mathbf{x}}^0$. A proof for the convergence in Proposition 1 is provided in Appendix C.

Proposition 1. *The Absorb & Escape (A&E) algorithm converges to the target distribution $p_{\theta,\beta}^{C}(\mathbf{x}) = p_\theta^{AR}(\mathbf{x}) \circ p_\beta^{DM}(\mathbf{x})$, under the assumptions that both models are properly trained, the segments of $\mathbf{x}$ are homogeneous, the subset of segments is chosen randomly, and the conditional distribution $p(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1}, \mathbf{x}_{j+1:L})$ is accurately approximated by $p_\theta^{AR}(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1})$.*

4.2 Practical Implementation: Fast A&E

While A&E offers a method to sample from a compositional distribution, two practical issues remain unresolved. Firstly, the algorithm may take a considerable amount of time to converge. Secondly, a crucial step in Line 3 of Algorithm 1 involves splitting $\mathbf{x}$ into homogeneous segments $\{\mathbf{s_1}, \mathbf{s_2}, \cdots, \mathbf{s_n}\}$ and then sampling a subset of these segments. Segmentation is straightforward when the boundaries of functional regions of the DNA sequence are known, such as protein-coding regions, exons, or introns, where each region naturally forms a homogeneous segment. However, this information is often unavailable in practice.

To address these challenges, we propose a practical implementation termed Fast A&E. For generating a sequence $\mathbf{x} \in \mathbb{N}_4^L$, it requires at most L forward passes through the AR model. As shown in Algorithm 2 and Figure 1b, Fast A&E adopts a heuristic-based approach to select segments for refinement. It scans the sequence from left to right, identifying low-quality segments through a thresholding mechanism. Tokens with predicted probabilities smaller than the T_{absorb} threshold trigger the absorb action, while the autoregressive process terminates once the probability of a token generated by the AR model $p^{DM}(\tilde{\mathbf{x}}'_j)$ is smaller than that of the diffusion model $p^{AR}(\tilde{\mathbf{x}}_j)$. In this manner, Fast A&E corrects errors made by the diffusion model with a maximum running time of $O(T_{DM} + T_{AR})$, where T_{DM} and T_{AR} are the times required for generating a single sequence from the diffusion model and autoregressive model, respectively.

5 Experiment

5.1 Transcription Profile (TP) conditioned Promoter Design

We first evaluate Fast A&E in the task of TP-conditioned promoter design, following the same evaluation procedures and metrics as used by DDSM [4] and Dirichlet Flow Matching (DFM) [32].

Data Format & Evaluation Metric: Each data point in this task is represented as a (DNA, signal) pair, where *signal* corresponds to the CAGE values for a given DNA sequence, providing a quantitative measure of gene expression around the transcription start site (TSS). Both the DNA sequence and the signal have a length of 1024. The goal of this task is to generate DNA sequences conditioned on specified signals. During evaluation, for a given test set data point (x, c), the generated sequence $\tilde{\mathbf{x}}$ and the ground truth sequence are processed through the genomic neural network Sei [37]. The performance metric is the mean squared error (MSE) between $Sei(x)$ and $Sei(\tilde{\mathbf{x}})$.

Table 2: **Evaluation of transcription profile conditioned promoter sequence design.** A&E achieves the smallest MSE with Language Model and DFM distilled being the AR and DM components.

Method	**MSE↓**
Bit Diffusion (bit-encoding)*	.0414
Bit Diffusion (one-hot encoding)*	.0395
D3PM-uniform*	.0375
DDSM*	.0334
Language Model*	.0333
Linear FM*	.0281
Dirichlet FM (DFM)*	.0269
Dirichlet FM distilled (DFM distilled)*	.0278
A&E (Language Model+Dirichlet FM distilled)	**.0262**

Results: As shown in Table 2, we ran Fast A&E on this task with a default threshold of $T_{\text{absorb}} = 0.85$, using the pretrained model as both the autoregressive (AR) model and the denoising model (DM) component. The evaluation followed the same procedure as described in the DFM repository. The Fast A&E model, comprising AR and DM components (i.e., the language model and the distilled DFM checkpoints provided in the DFM repository), achieved state-of-the-art results with an MSE of 0.0262 on the test split.

5.2 Multi-species Promoter Generation

5.2.1 Experimental Setup

Dataset Construction Prior efforts in DNA generation have been constrained by small, single-species datasets [19, 34]. To better evaluate the capability of various generative algorithms in DNA generation, we construct a dataset with 15 species from the Eukaryotic Promoter Database (EPDnew)[23]. Table 3 compares our EPD dataset with those used in previous studies, including DDSM[4], ExpGAN [38], and EnhancerDesign [33]. The key advantage of EPD is its diversity in both species types and DNA sequence types. Additionally, although the number of sequences in EPD is on a similar scale to that of DDSM, EPD offers greater uniqueness: each sequence corresponds to a unique promoter-gene combination, a guarantee not provided by the other datasets.

Baseline Model We evaluate the state-of-the-art diffusion models for DNA sequence generation: *DDSM* [4], *DNADiffusion* [26], *DDPM* [2, 3], and a AR model *Hyena* [20, 24]. In addition, we implement a VAE with a CNN-based encoder-decoder architecture. Adding UNet as the denoising network to VAE results in another baseline latent diffusion model termed *DiscDiff*. For a fair evaluation, we maximally scale up the denoising networks of each diffusion model to fit into an 40GB NVIDIA A100. Additionally, we adapt four pretrained *Hyena* models from HuggingFace for comprehensive fine-tuning. The additional details of the network architectures are shown in Appendix D.

Table 3: **A comparison of DNA generation datasets.** EPD used in this work is significantly larger in size and contains fifteen species. Reg. represents the regulatory regions, and Prot. represents the protein encoding region.

Dataset	# DNA	Multi Species	DNA Regions
EPD (Ours)	**160,000**	✓	Reg. & Prot.
DDSM [4]	100,000	×	Reg. & Prot.
ExpGAN [38]	4238	×	Reg.
EnhancerDesign [33]	7770	×	Reg.

Model Training All the models are implemented in Pytorch and trained on a NVIDIA A100-PCIE-40GB with a maximum wall time of 48 GPU hours per model; most of the models converged within the given time. Adam optimizer [7] is used together with the CosineAnnealingLR [22] scheduler. The learning rate of each model are detailed in Appendix D. For the evaluation of various diffusion models in unconditional generation (see Section 5.2.2), we sample 50,000 sequences from each model. For the conditional generation across 15 species (see Section 5.3), we generate 4,000 sequences. In both cases, we use the DDPM sampler [31] with 1,000 sequential denoising steps.

5.2.2 Evaluating Diffusion Models on Mammalian Model Organisms

Table 4: Comparison of diffusion models on unconditional generation evaluated on EPD (256 bp) and EPD (2048 bp). Metrics include S-FID, Cor_{TATA}, and MSE_{TATA}. The **best** and <u>second-best</u> scores are highlighted in bold and underlined, respectively.

	EPD (256 bp)			EPD (2048 bp)		
Model	S-FID ↓	Cor_{TATA} ↑	MSE_{TATA} ↓	S-FID ↓	Cor_{TATA} ↑	MSE_{TATA} ↓
Random (Reference)	119.0	-0.241	8.21	106.0	0.030	1.86
Sample from Training Set	0.509	1.0	0	0.100	0.999	0
VAE	295.0	-0.167	26.5	250.0	0.007	9.40
BitDiffusion	405	0.058	5.29	100.0	0.066	5.91
D3PM (small)	<u>97.4</u>	0.0964	4.97	<u>94.5</u>	0.363	**1.50**
D3PM (large)	161.0	-0.208	<u>4.75</u>	224.0	0.307	8.49
DDSM (Time Dilation)	504.0	<u>0.897</u>	13.4	1113.0	<u>0.839</u>	2673.7
DiscDiff (Ours)	**57.4**	**0.973**	**0.669**	**45.2**	**0.858**	<u>1.74</u>

One prerequisite of Fast A&E (Algorithm 2) is that the diffusion model P_β^{DM} should be properly trained and provide accurate approximations of underlying data distribution. We first evaluate existing Diffusion Models on a subset of EPD datasets. This subset includes sequences from four mammalians *H. Sapiens* (human), *Rattus Norvegicus* (rat), *Macaca mulatta*, and *Mus musculus* (mouse), which collectively represent 50% of the total dataset. Training on this subset allows for a more precise assessment of the generative algorithm's accuracy in a unconditional generation setting.

Metrics

1. **Motif Distribution Correlation (Cor_M)** and **Mean Square Error (MSE_M)**: Cor_M is the Pearson correlation between the motif distributions of generated and natural DNA sequences for motifs like TATA-box, GC-box, Initiator, and CCAAT-box. MSE_M is the average squared differences between these motif distributions.
2. **S-FID (Sei Fréchet Inception Distance)**: Measures the distance between distributions of generated and natural DNA sequences in latent space similar to the FID metric [15] for images, replacing the encoder with the pre-trained genomic neural network, Sei [37].

The results are presented in Table 4, indicating that most existing models perform worse than the simple baseline DiscDiff proposed here, as measured by S-FID, Cor_{TATA}, and MSE_{TATA}. TATA is one of the most fundamental motifs for gene transcription – a special type of protein called transcription factors binds to TATA tokens on the DNA as shown in Figure 1a. It changes the shape of DNA and then enables the gene transcription. The failure of existing diffusion models to capture the TATA-box distribution indicates a potential gap in existing research. In the following, we hereby use DiscDiff as the p_β^{DM} to initialize the A&E algorithm.

5.3 Multi-species DNA Sequences Generation

We compare our model Fast A&E with *Hyena* [24] and the best-performing diffusion model from Section 5.2.2, *DiscDiff*, on the task of generating species-specific DNA sequences.

Motif-centric Evaluation We consider four types of motifs closely related to promoter activities $\{\text{TATA-box}, \text{GC-box}, \text{Initiator}, \text{CCAAT-box}\}$. We calculate 4 types of motif distributions for 15 species across 3 models, resulting in 180 frequency distributions.

Figure 3 shows the average MSE and Correlation between generated and natural DNA distributions for each model and motif type across 15 species. Fast A&E improves upon *Hyena* and *DiscDiff*, generating the most realistic sequences across almost all species and motif types. It achieves the lowest MSE and highest Correlation across all four motifs. This pattern is consistent across all 15 species. The motif plots for all 15 species are provided in Appendix F. As an example, Figure 4 shows the motif distributions of sequences generated by the three models versus real DNA sequences

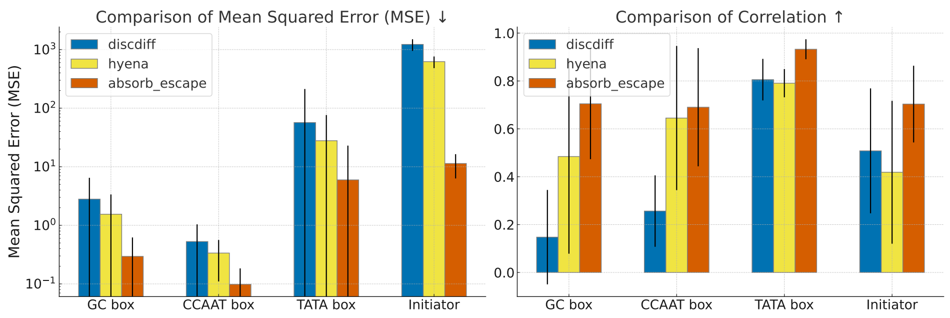


Figure 3: **The average MSE and Correlation between generated and natural DNA distributions for each model and motif type across 15 species.** Fast A&E outperforms *Hyena* and *DiscDiff*, generating the most realistic sequences with the lowest MSE and highest Correlation across four motif types. This pattern is consistent across all 15 species.

for *macaque*. Fast A&E closely resembles the natural motif distribution, especially for the TATA and GC box motifs, while *Hyena* and *DiscDiff* fail to capture the values or trends accurately.

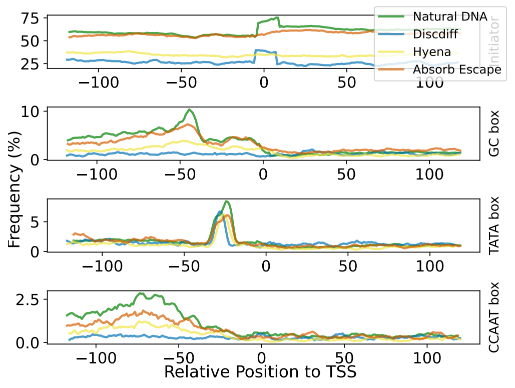


Figure 4: **Motif distributions in macaque DNA compared across natural DNA, FAST A&E, DiscDiff, and Hyena**. FAST A&E closely aligns with natural DNA, especially for the TATA and GC motifs.

Sequence Diversity To assess the diversity of the generated sequences, we applied BLASTN [18] to check (1) the similarity between the training set (Natural DNA) and generated sequences from three models, and (2) the similarity within the generated sequences. BLASTN takes a query DNA sequence and compares it with a database of sequences, returning all the aligned sequences in the database that are similar to the query. For each alignment, an alignment score, the alignment length (AlignLen), and statistical significance (eValue) are provided to indicate the quality of the alignment, where a larger alignment score, a smaller statistical significance (eValue), and a longer alignment sequence (AlignLen) indicate a higher similarity between the query sequence and the database sequences. Ideally, when using generated sequences to query the training dataset, a good generative sequence should align better than a random sequence, but not replicate the sequences in the training set.

Table 5 shows the results of the BLASTN algorithm. From the table, *DiscDiff*, *Hyena*, and A&E all satisfied the mentioned criteria. In terms of the diversity within the groups, none of the algorithms generated repetitive sequences. Furthermore, A&E tends to lie between Hyena and DiscDiff in terms of the diversity of the generated sequences, implying that A&E may combine the properties of AR and DM models. One notable fact is that the alignment scores are very high within the natural sequences, potentially indicating that natural sequences naturally have repetitive properties (conservative motifs), while the generative sequences do not have this characteristic.

Table 5: **BLASTN Results**

Query vs. Database	Score↑	eValue↓	AlignLen↑
Random vs. Natural DNA	17.78	1.1769	24.2
A&E vs. Natural	21.39	0.1695	35.9
Hyena vs. Natural	22.89	0.2895	40.1
DiscDiff vs. Natural	20.25	0.2098	31.4
A&E vs. A&E	20.14	0.0968	33.69
Hyena vs. Hyena	19.57	0.0843	28.7
DiscDiff vs. DiscDiff	20.95	0.1029	37.6
Natural vs. Natural DNA	57.06	0.0633	77.6

Genome Integration with Promoter Sequences As shown in Figure 5, to evaluate the functional properties of sequences generated by *Hyena*, *DiscDiff*, and *A&E*, we inserted the generated promoter sequences of length 128 bp upstream (5') of three commonly studied genes in oncology: **TP53**,

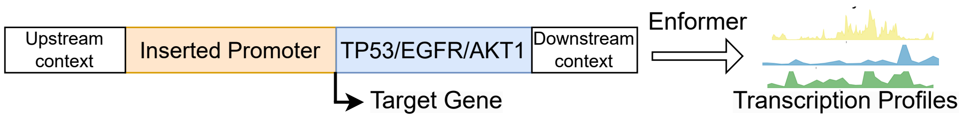


Figure 5: **Evaluation of Generated Promoters for gene regulation through Genome Integration EGFR**, and **AKT1** [6, 25, 29], which are closely related to tumor activities. Our goal was to determine which model generates promoter sequences that produce gene expression levels closest to those of natural promoters when reinserted into the human genome.

Table 6: **Sum of Squared Errors (SSE) of Transcription Profiles between Real and Generated Sequences**

	TP53↓	EGFR↓	AKT1↓
Random	278.18	8.09	65.70
A&E	**17.21**	**0.28**	**1.65**
Hyena	36.25	0.89	2.88
DiscDiff	124.03	2.17	25.50

We use Enformer [5] to predict transcription profiles. Enformer takes a DNA sequence of 200k bps as input and outputs a matrix $P \in \mathbb{R}^{896\times 638}$, representing a multi-cell type transcription profile. We sampled 300 promoter sequences from each source: *Natural DNA promoters*, *Hyena*, *DiscDiff*, and A&E. For each set, we calculated the average transcription profile across the sequences. The Sum of Squared Errors (SSE) between these average transcription profiles of the generated sequences and those of natural promoters are shown in Table 6. The results indicate that A&E produces the smallest SSE, suggesting it best captures the properties of natural DNA. This finding highlights the potential of generative algorithms to create promoter sequences that effectively regulate gene expression, with applications in bioproduct manufacturing and gene therapy.

5.3.1 Sensitivity Analysis of T_{Absorb}

We perform a *sensitivity analysis* of A&E algorithm over hyperparameter T_{absorb}. As shown in Figure 6, with a small T_{absorb}, the sequences generated by A&E are dominated by the diffusion model. As T_{absorb} increases, the AR helps to correct the errors made by the DM. Finally, when T_{absorb} is larger than 0.7, the correlation flattens and fluctuates. In conclusion, A&E is robust under different values of T_{absorb}, and it is best to use the validation dataset to choose the optimal value. However, a wide range of T_{absorb} can still be used with improved performance.

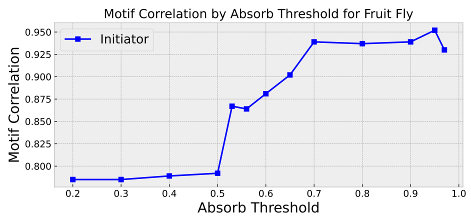


Figure 6: **Sensitivity of A&E under various T_{absorb}**. For each value of T_{absorb}, 3000 sequences are generated and compared with natural DNA. The correlation between the generated sequences and natural DNA increases initially as T_{absorb} increases, and then it flattens. An optimal T_{absorb} can be selected based on the validation set, or a default value of 0.85 can be used.

6 Conclusion

This paper demonstrates that (i) both the AutoRegressive (AR) model and Diffusion Models (DMs) fail to accurately model DNA sequences due to the heterogeneous nature of DNA sequences when used separately, and (ii) this limitation can be overcome by introducing A&E, a novel sampling algorithm that combines AR models and DMs. Additionally, we developed a fast implementation of the proposed algorithm, Fast A&E, which enables efficient generation of realistic DNA sequences without the repetitive function evaluations required by conventional sampling algorithms. Experimental results across 15 species show that Fast A&E consistently outperforms single models in generating DNA sequences with functional and structural similarities to natural DNA, as evidenced by metrics such as Motif Distribution, Sequence Diversity, and Genome Integration. Regarding the future work, the generated DNA sequences still require validation through wet-lab experiments before they can be directly used in clinical settings.

Acknowledgments and Disclosure of Funding

This research project has benefitted from the Microsoft Accelerate Foundation Models Research (AFMR) grant program. Zehui Li is grateful to his family—Jinghai Li, Huiqing Wang, and Yiqiu Sun for their support.

References

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3, 2024.

[2] Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Alex Xijie Lu, Nicolo Fusi, Ava Pardis Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *bioRxiv*, pages 2023–09, 2023.

[3] 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.

[4] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. *arXiv preprint arXiv:2305.10699*, 2023.

[5] 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.

[6] Gillian Bethune, Drew Bethune, Neale Ridgway, and Zhaolin Xu. Epidermal growth factor receptor (egfr) in lung cancer: an overview and update. *Journal of thoracic disease*, 2(1):48, 2010.

[7] Kingma Da. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, pages 2023–01, 2023.

[9] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pages 8489–8510. PMLR, 2023.

[10] Yilun Du and Leslie Kaelbling. Compositional generative modeling: A single model is not all you need. *arXiv preprint arXiv:2402.01103*, 2024.

[11] 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.

[12] Alan E Gelfand. Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304, 2000.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[14] Anvita Gupta and James Zou. Feedback gan for dna optimizes protein functions. *Nature Machine Intelligence*, 1(2):105–111, 2019.

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

[17] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

[18] W James Kent. Blat—the blast-like alignment tool. *Genome research*, 12(4):656–664, 2002.

[19] 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.

[20] Avantika Lal, David Garfield, Tommaso Biancalani, and Gokcen Eraslan. reglm: Designing realistic regulatory dna with autoregressive language models. *bioRxiv*, pages 2024–02, 2024.

[21] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing*

Systems, 35:4328–4343, 2022.

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] Patrick Meylan, René Dreos, Giovanna Ambrosini, Romain Groux, and Philipp Bucher. Epd in 2020: enhanced data visualization and extension to ncrna promoters. *Nucleic Acids Research*, 48(D1):D65–D69, 2020.

[24] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36, 2024.

[25] Magali Olivier, Monica Hollstein, and Pierre Hainaut. Tp53 mutations in human cancers: origins, consequences, and clinical use. *Cold Spring Harbor perspectives in biology*, 2(1):a001008, 2010.

[26] Luca Pinello. Dna-diffusion: Leveraging generative models for controlling chromatin accessibility and gene expression via synthetic regulatory elements. In *ICLR 2024 Workshop on Machine Learning for Genomics Explorations*.

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[29] Marion Rudolph, Tobias Anzeneder, Anke Schulz, Georg Beckmann, Annette T Byrne, Michael Jeffers, Carol Pena, Oliver Politz, Karl Köchert, Richardus Vonk, et al. Akt1 e17k mutation profiling in breast cancer: prevalence, concurrent oncogenic alterations, and blood-based detection. *BMC cancer*, 16:1–12, 2016.

[30] 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.

[31] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[32] 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.

[33] 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*, pages 1–3, 2023.

[34] 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.

[35] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, pages 1–3, 2023.

[36] Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua Susskind, and Navdeep Jaitly. Planner: Generating diversified paragraph via latent language diffusion model. *Advances in Neural Information Processing Systems*, 36, 2024.

[37] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015.

[38] 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.

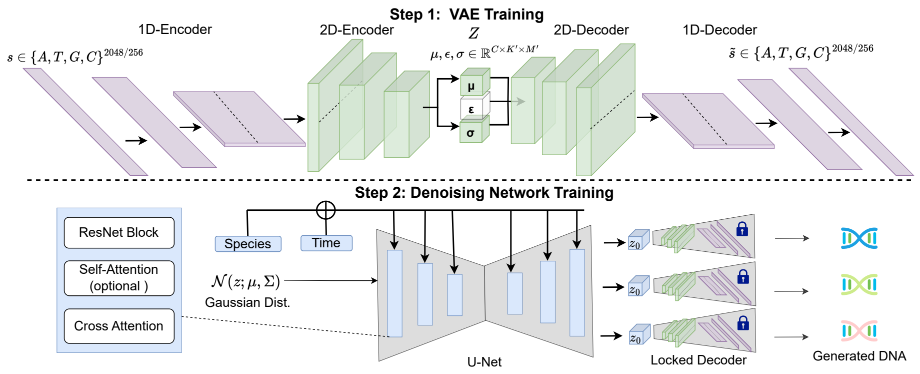

Step 1: VAE Training
1D-Encoder
2D-Encoder
2D-Decoder
1D-Decoder
$s \in \{A,T,G,C\}^{2048/256}$
Z
$\mu, \epsilon, \sigma \in \mathbb{R}^{C\times K'\times M'}$
$\tilde{s} \in \{A,T,G,C\}^{2048/256}$
Step 2: Denoising Network Training
ResNet Block
Self-Attention (optional)
Cross Attention
Species
Time
$\mathcal{N}(z;\mu,\Sigma)$
Gaussian Dist.
U-Net
Locked Decoder
Generated DNA

Figure 7: DiscDiff Model: A two-step process for DNA sequence generation. **Step 1: VAE Training**: A sequence $s \in \{A, T, G, C\}^{2048/256}$ is encoded via a 1D-Encoder to a 2D-Encoder. The latent space representation Z with parameters μ, ϵ, σ is then decoded back to $\tilde{s}$ through a 2D-Decoder and 1D-Decoder. **Step 2: Denoising Network Training**: The latent representation Z is processed through a denoising network comprising a ResNet Block, optional Self-Attention, and Cross Attention, with species and time information. The network outputs a Gaussian distribution $N(z; \mu, \Sigma)$. A U-Net architecture takes this distribution to produce various z_0 representations, which a Locked Decoder (fronzen parameters) used to generate the final DNA sequences.

A A simple baseline latent diffusion model for discrete data: DiscDiff

A.1 The DiscDiff Model

We design DiscDiff, a Latent Discrete Diffusion model for DNA generation tasks. As shown in Figure 7, this model is structured into two main components: a Variational-Auto-Encoder (VAE) and a denoising model. The VAE consists of an encoder $\mathbf{E} : \underline{s} \mapsto \underline{z}$, which maps discrete input sequence $\underline{s}$ to a continuous latent variable $\underline{z}$, and a decoder $\mathbf{D} : \underline{z} \mapsto \underline{\tilde{s}}$, which reverts $\underline{z}$ back to $\tilde{s}$ in the discrete space. The denoising model $\varepsilon_\theta(\mathbf{z}_t, t)$ is trained to predict the added noise ε in the latent space.

A.1.1 VAE Architecture

The choice of VAE architecture in LDMs is critical and often domain-specific. We find that mapping the input data to a higher dimension space can help to learn a better denoising network, generating more realistic DNA sequences. We hereby propose to use a two-stage VAE architecture as shown in Figure 7.

The first stage encoder $\mathbf{E}_{\phi_1} : \mathbb{N}_4^L \to \mathbb{R}^{K\times M}$ maps $\underline{s} \in \mathbb{N}_4^L$ to a 2D latent space $\underline{z}_1 \in \mathbb{R}^{K\times M}$, where K is the number of channels and M is the length of the latent representation. The second stage encoder $\mathbf{E}_{\phi_2} : \mathbb{R}^{1\times K\times M} \to \mathbb{R}^{C\times K'\times M'}$ first adds a dummy dimension to $\underline{z}_1$ such that $\mathbf{z}_1 \in \mathbb{R}^{1\times K\times M}$ and then maps it to 3d latent space $\mathbf{z} \in \mathbb{R}^{C\times K'\times M'}$, where C is the number of channels, K' and M' are the reduced dimensions of K and M respectively. The decoder in the first and second stage are $\mathbf{D}_{\theta 1}$ and $\mathbf{D}_{\theta 2}$ respectively. Which are symmetric to the encoders. Overall, we have $\mathbf{z} = \mathbf{E}_\phi(\mathbf{s}) = \mathbf{E}_{\phi_2}(\mathbf{E}_{\phi_1}(\mathbf{s}))$, and the reconstruction is $\tilde{\mathbf{s}} = \mathbf{D}_\theta(\mathbf{z}) = \mathbf{D}_{\theta 1}(\mathbf{D}_{\theta 2}(\mathbf{z}))$.

A.1.2 VAE Loss

When training the VAE, we propose to use Cross Entropy (CE) as reconstruction loss. The loss function is given by:

$$\mathbf{L}_{\theta,\phi} = \underbrace{\mathbb{E}_{p(\mathbf{s})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{s})}\left[-\sum_{l=1}^{L}\sum_{i=1}^{4}\delta_{is_l}\log p_\theta(\mathbf{s}_l|\mathbf{z})\right]\right]}_{\text{Reconstruction Loss}} + \beta\cdot\underbrace{\mathbb{E}_{p(\mathbf{s})}\left[\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{s})\,||\,\mathcal{N}(\mathbf{z};\mu,\mathbf{\Sigma}))\right]}_{\text{KL Divergence}}$$

where δ_{ij} is the Kronecker delta, $p_\theta(\mathbf{s}|\mathbf{z})$ is the probabilistic decoder output from $\mathbf{D}_\theta$; $q_\phi(\mathbf{z}|\mathbf{s})$ is the probabilistic output from encoder $\mathbf{E}_\phi$ that represents the approximate posterior of the latent variable $\mathbf{z}$ given the input s; $\mathcal{N}(\mathbf{z};\mu,\mathbf{\Sigma})$ is the prior on $\mathbf{z}$. Here we use a simple isotropic. β is a mixing hyperparameter. β is set to 10^{-5} in the experiments used in this paper.

A.1.3 Denoising Network Training

Once $\mathbf{D}_\theta$ and $\mathbf{E}_\phi$ are trained in the first step, we train a noise prediction ε_θ in the latent space $\mathbf{z} = \mathbf{E}_\phi(\mathbf{s})$ with Equation (6).

$$\mathbb{E}_{\mathbf{z},t\sim U[1,T],\varepsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{z}_t,t)\|_2^2\right] \tag{6}$$

B Toy Experiment Training Details

We train both HyenaDNA and DiscDiff on an NVIDIA A100-PCIE-40GB using the Adam optimizer. For HyenaDNA, the learning rate is set to 0.0001, and we use the model `heyenadna-large-1m-seqlen` for this task. The maximum number of epochs is set to 100, with early stopping enabled to facilitate early convergence.

For DiscDiff, the VAE is trained with a learning rate of 0.0001, while the UNet is trained with a learning rate of 0.00005. DiscDiff is trained for 600 epoches; during the inference time, we use DDPM [16] sampler with 1000 denoising steps.

C Convergence Proof of the Absorb & Escape Algorithm

In this section, we provide a proof for the convergence of the Absorb & Escape (A&E) algorithm under certain assumptions. The convergence proof will demonstrate that the sequence generated by the A&E algorithm converges to a sample from the target distribution $p^C_{\theta,\beta}(\mathbf{x})$.

C.1 Assumptions

We make the following assumptions for the convergence proof:

1. The autoregressive model $p^{AR}_\theta(\mathbf{x})$ and the diffusion model $p^{DM}_\beta(\mathbf{x})$ are both properly trained and provide accurate approximations of the underlying data distribution.
2. The initial sample $\mathbf{x}^0 \sim p^{DM}_\beta(\mathbf{x})$ is a valid sample from the diffusion model.
3. The segments $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$ of the sequence $\mathbf{x}$ are chosen such that each segment is homogeneous.
4. The subset of segments $\mathcal{S}$ is chosen randomly and includes all segments over multiple iterations.
5. The conditional distribution $p(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1}, \mathbf{x}_{j+1:L})$ approximated by $p^{AR}_\theta(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1})$ is accurate.

C.2 Proof

We aim to show that the A&E algorithm produces samples from the target distribution $p^C_{\theta,\beta}(\mathbf{x})$. We do this by showing that the Markov chain defined by the A&E algorithm has $p^C_{\theta,\beta}(\mathbf{x})$ as its stationary distribution.

Step 1: Initialization The initial sample $\mathbf{x}^0 \sim p_\beta^{DM}(\mathbf{x})$ is drawn from the diffusion model. This ensures that $\mathbf{x}^0$ is a valid sample from $p_\beta^{DM}(\mathbf{x})$.

Step 2: Absorb Step For each segment $\mathbf{s}_k$ in the subset $\mathcal{S}$, the Absorb step samples $\tilde{\mathbf{x}}'_{i:j}$ from the conditional distribution $p(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1}, \mathbf{x}_{j+1:L})$. Under the assumption that $p_\theta^{AR}(\mathbf{x}_{i:j}|\mathbf{x}_{0:i-1})$ accurately approximates this conditional distribution, the refined segment $\tilde{\mathbf{x}}'_{i:j}$ is a valid sample from the target conditional distribution.

Step 3: Escape Step The Escape step updates the segment $\tilde{\mathbf{x}}^t_{i:j}$ to $\tilde{\mathbf{x}}'_{i:j}$. This ensures that the updated sequence $\tilde{\mathbf{x}}^t$ incorporates the refinement from the autoregressive model.

Step 4: Stationary Distribution To show that the Markov chain defined by the A&E algorithm converges to $p_{\theta,\beta}^C(\mathbf{x})$, we need to show that $p_{\theta,\beta}^C(\mathbf{x})$ is the stationary distribution of this Markov chain.

The transition probability for the A&E algorithm is given by the product of the probabilities of the Absorb and Escape steps:

$$P(\mathbf{x} \to \mathbf{x}') = P_{\text{Absorb}}(\mathbf{x} \to \mathbf{x}')P_{\text{Escape}}(\mathbf{x}' \to \mathbf{x}).$$

Given that $p_\beta^{DM}(\mathbf{x})$ captures the overall structure and $p_\theta^{AR}(\mathbf{x})$ refines the segments, the composed distribution $p_{\theta,\beta}^C(\mathbf{x})$ is achieved by iteratively applying the Absorb and Escape steps. We need to show that the stationary distribution satisfies:

$$\int p_{\theta,\beta}^C(\mathbf{x})P(\mathbf{x} \to \mathbf{x}')d\mathbf{x} = p_{\theta,\beta}^C(\mathbf{x}').$$

Using the detailed balance condition for the Markov chain:

$$p_{\theta,\beta}^C(\mathbf{x})P(\mathbf{x} \to \mathbf{x}') = p_{\theta,\beta}^C(\mathbf{x}')P(\mathbf{x}' \to \mathbf{x}).$$

Given that the Markov chain is ergodic, the detailed balance condition implies that $p_{\theta,\beta}^C(\mathbf{x})$ is the stationary distribution.

Step 5: Ergodicity Ergodicity ensures that the Markov chain will visit all possible states given sufficient iterations. The random selection of segments $\mathcal{S}$ and the iterative updates in the A&E algorithm guarantee that all parts of the sequence are refined over time.

Conclusion By satisfying the detailed balance condition and ergodicity, we have shown that the Markov chain defined by the A&E algorithm converges to the target distribution $p_{\theta,\beta}^C(\mathbf{x})$. Therefore, the A&E algorithm produces samples from the composed distribution $p_{\theta,\beta}^C(\mathbf{x})$ as the number of iterations t approaches infinity.

D Experiment Details

Baselines The details about the **architecture and implementation** of the baseline models are as below:

- DNADiffusion [26]: We enhance the current DNADiffusion implementation for DNA synthesis, originally from the DNA-Diffusion project[2], by expanding the models to encompass 380 million parameters. This network is composed of Convolutional Neural Networks (CNNs), interspersed with layers of cross-attention and self-attention. The learning rate is set to 0.0001.
- DDSM [4]: We scale up the original implementation of the denoising network used for promoter design in DDSM[3] to what is the corresponding size of the network given 470 million parameters. It is a convolution-based architecture with dilated convolution layers. The learning rate is set to 0.00001.

[2]https://github.com/pinellolab/DNA-Diffusion

[3]https://github.com/jzhoulab/ddsm

- D3PM [3]: We take the implementation of D3PM for biological sequence generation from EvoDiff [2][4], adopting the algorithm for DNA generation. We use the original implementation of the denoising network, which has two versions: with sizes of 38M and 640M. We hereby have D3PM (small) and D3PM (big), respectively. The learning rate for both D3PM (small) and D3PM (large) are set to 0.0001.
- Hyena [24]: We modify the RegLM [20][5], a existing work uses hyena for DNA generation. Four pretrained Hyena models of different sizes (hyenadna-large-1m-seqlen, hyenadna-medium-160k0seqlen, heynadna-small-32k-seqlen, and hyenaana-tiny-16k-seqlen-d128) are downloaded from HuggingFace[6] and used for full-size fine-tuning, we apply the fine-tuned models for generations on EPD-GenDNA. The learning rate for fine-tuning is set to 0.0001.
- DiscDiff: A 2D-UNet of 500 Million parameters are used as the denoising network. See Appendix A for the implementation details. The learning rate is set to 0.00005 for UNet training, 0.0001 for VAE training.

For the Fast A&B algorithm, we set the T_{absorb} to 0.80.

E Content List of Supplementary Code and Data

Our code folder includes the following sub-folders, within each folder there is a readme file, detailing the steps to run the code. The below is the list of sub-folders.

E.1 Toy Experiment

Include the source code to reproduce Section 3. No external package is required to run the code except for python

E.2 DiscDiff

A implementation of the discdiff baseline in Appendix A. Please follow the readme for running the code.

E.3 AbsorbEscape

This includes an implementation of the proposed algorithm in Section 4.2; however, it depends on the external AR models, please adopt it accordingly to AR models which you want to try.

E.4 EPD Data

We include the training dataset used for producing the main results, which includes 160K DNA sequences from EPD, each sequence has a length of 256 bp.

F Motif Distributions for 15 species

We plot TATA-box, GC content, Initiator, and CCAAT-box for 15 speces as below.

[4]https://github.com/microsoft/evodiff

[5]https://github.com/Genentech/regLM

[6]https://huggingface.co/LongSafari

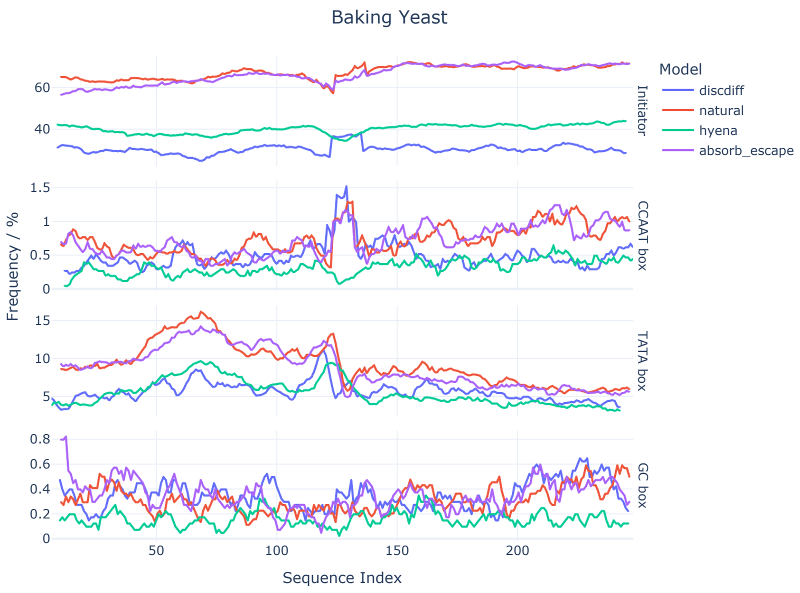


Figure 8: Baking Yeast

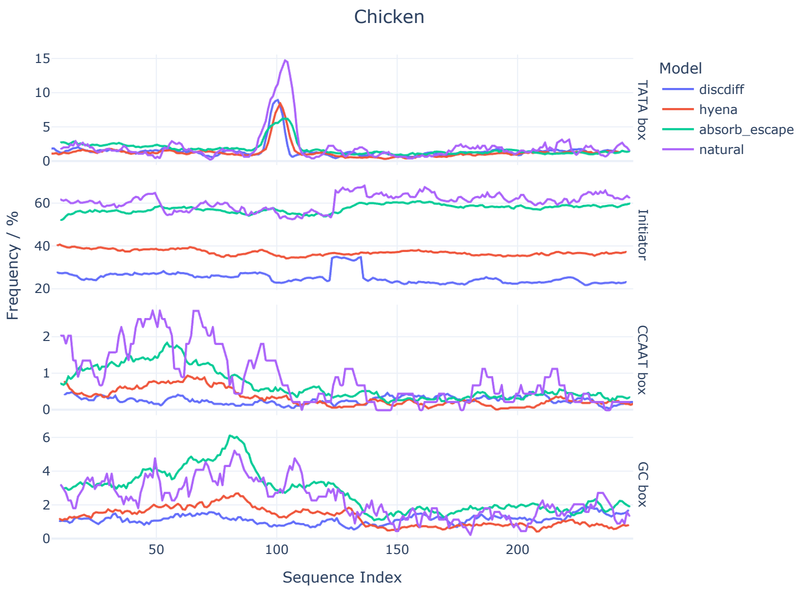


Figure 9: Chicken

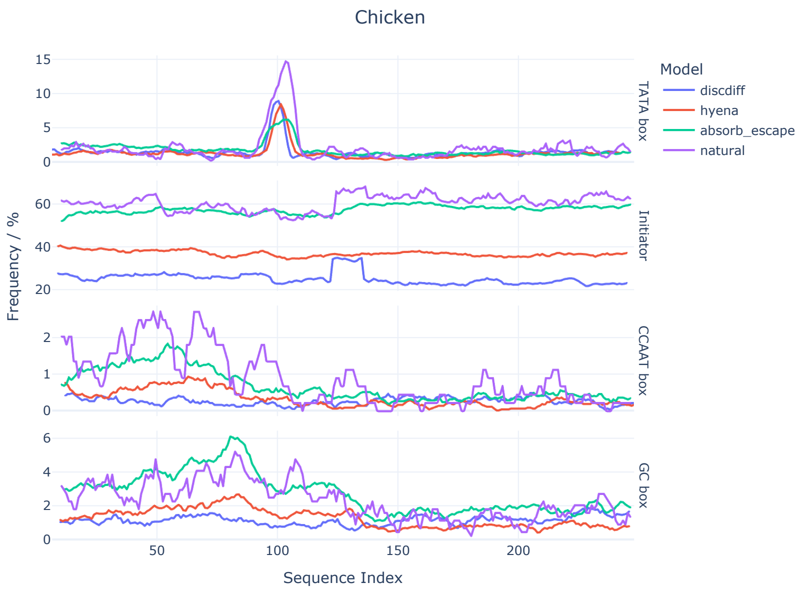


Figure 10: Chicken

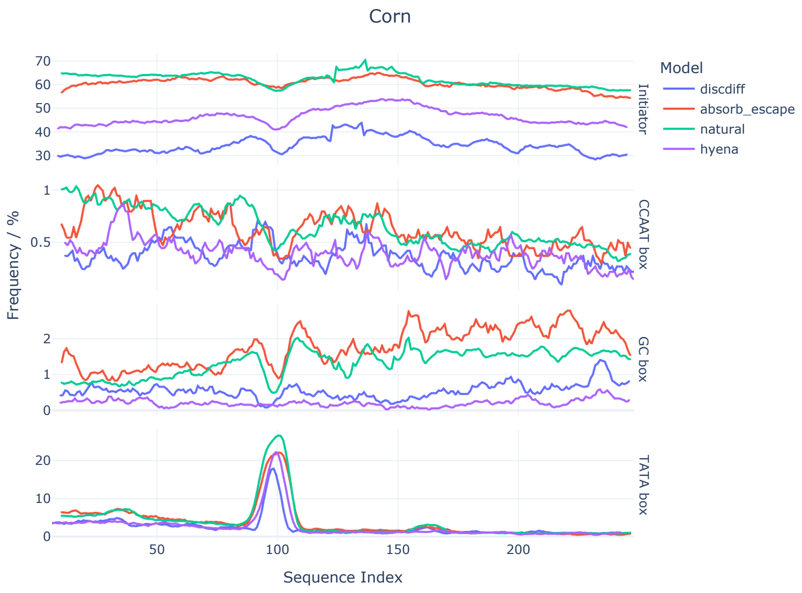


Figure 11: Corn

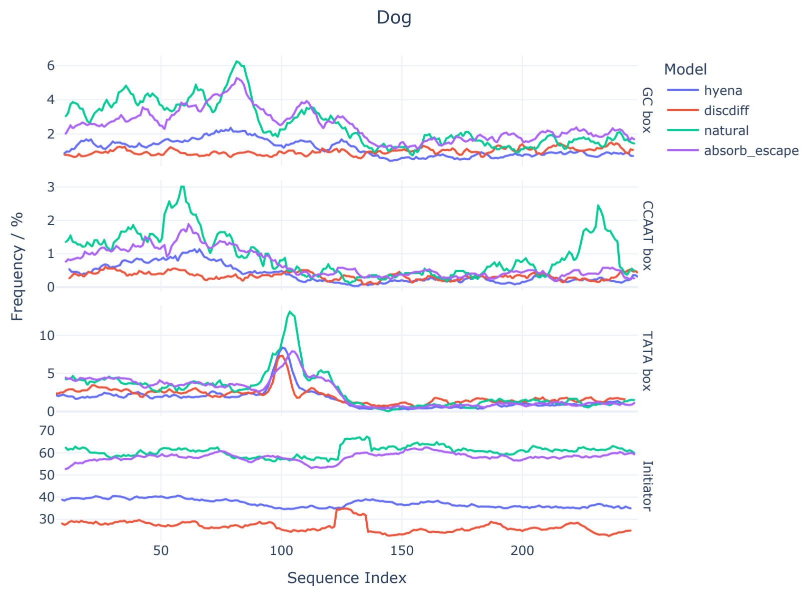


Figure 12: dog

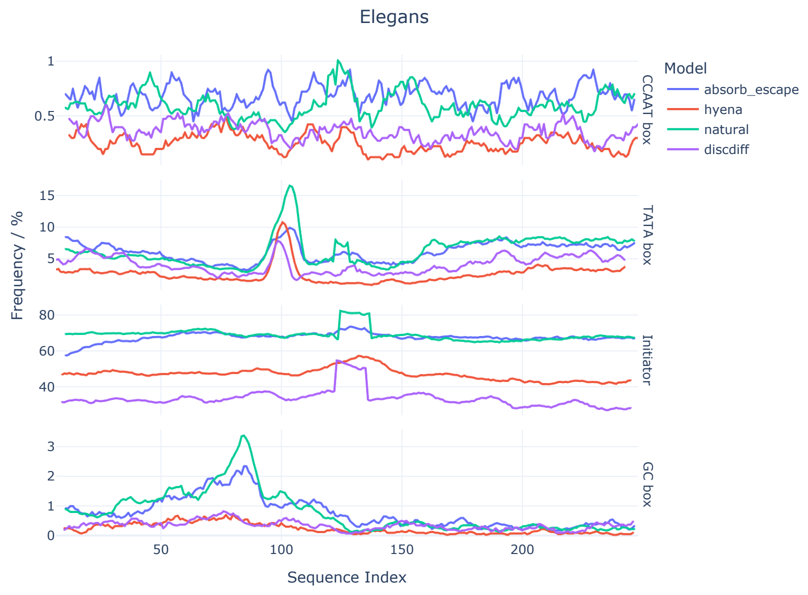


Figure 13: elegans

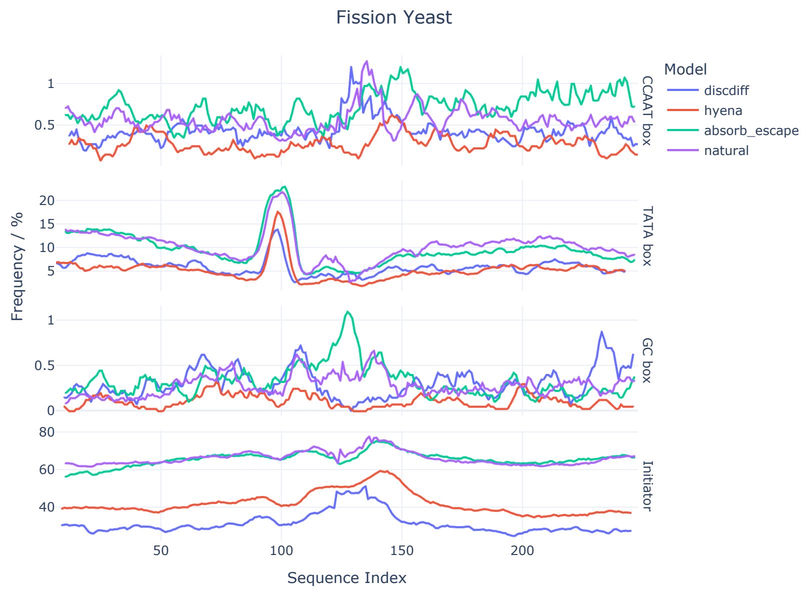


Figure 14: fission yeast

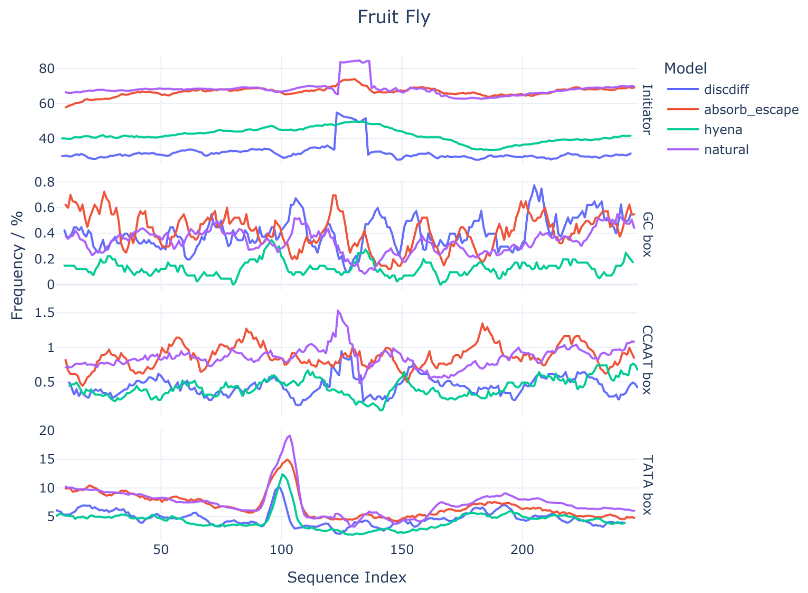


Figure 15: fruit fly

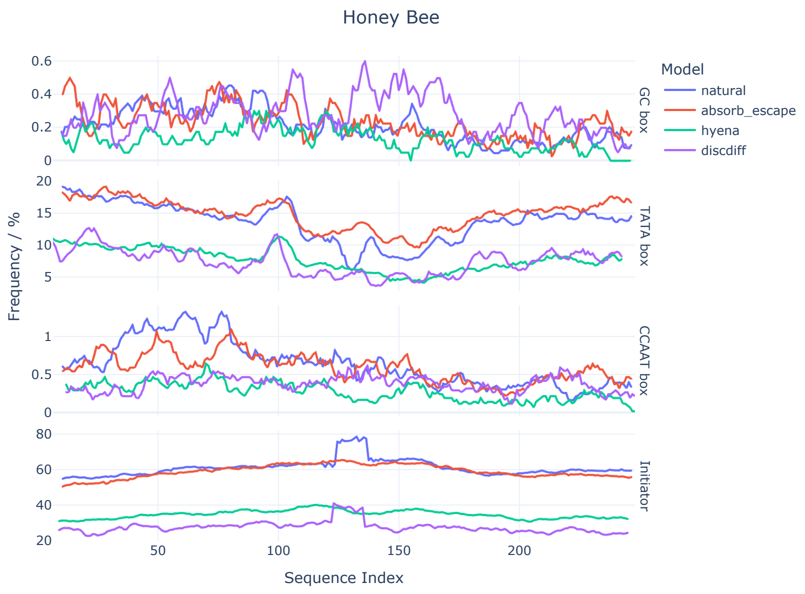


Figure 16: honey bee

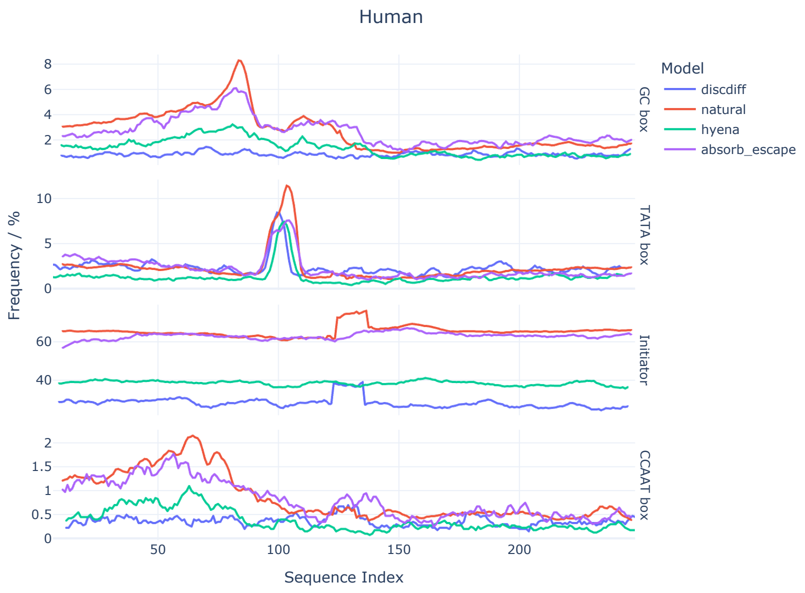


Figure 17: human

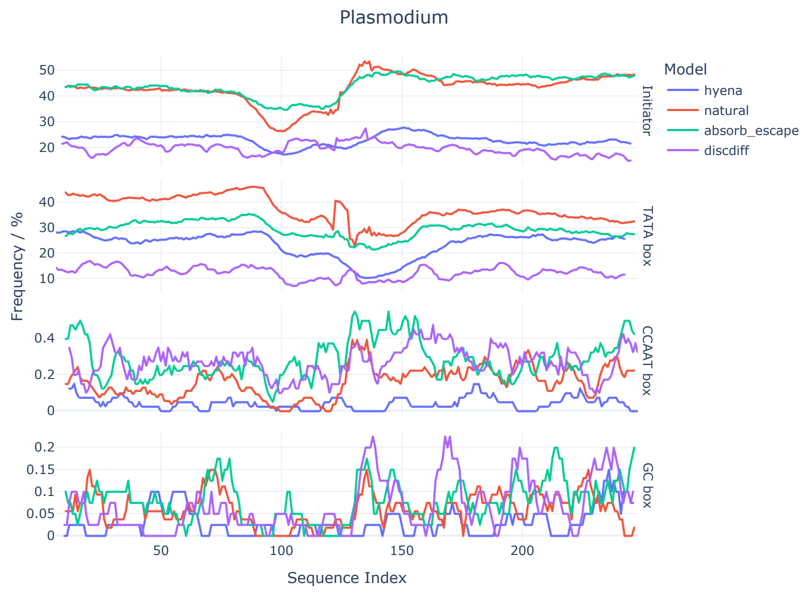


Figure 18: plasmodium

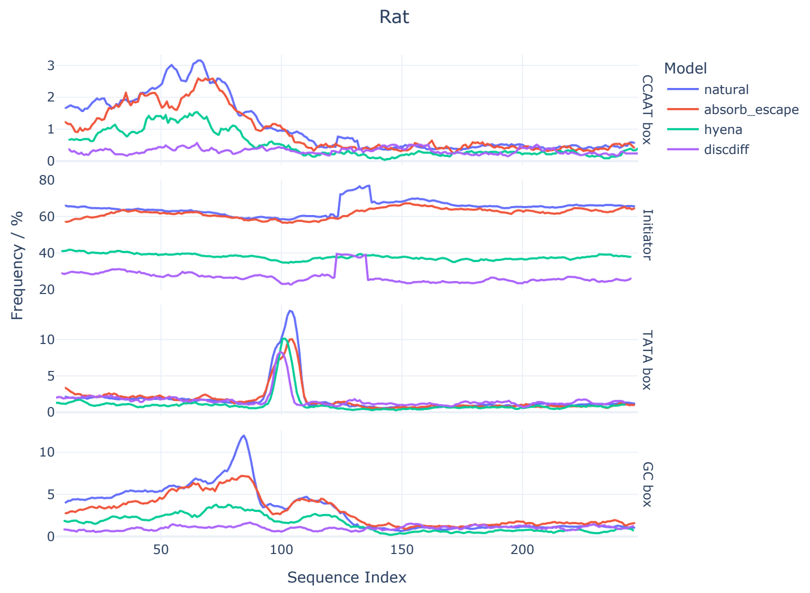


Figure 19: rat

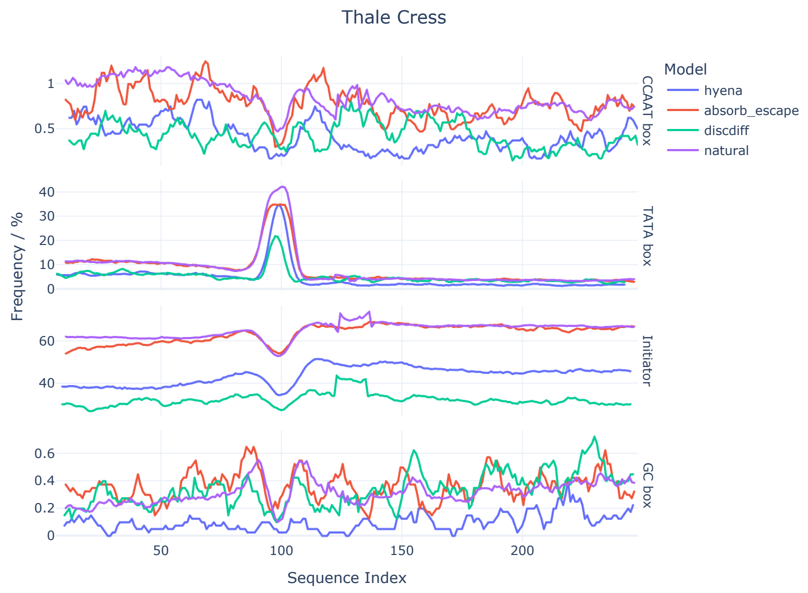


Figure 20: thale cress

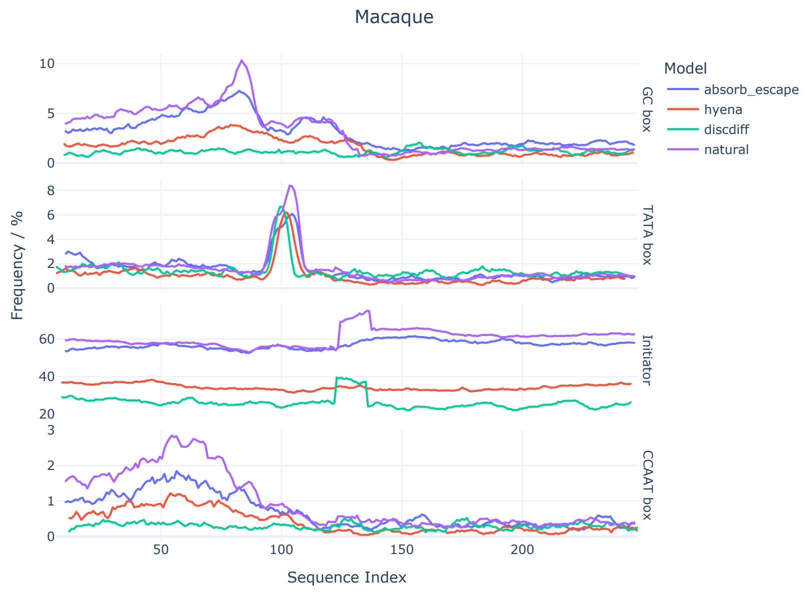


Figure 21: macaque

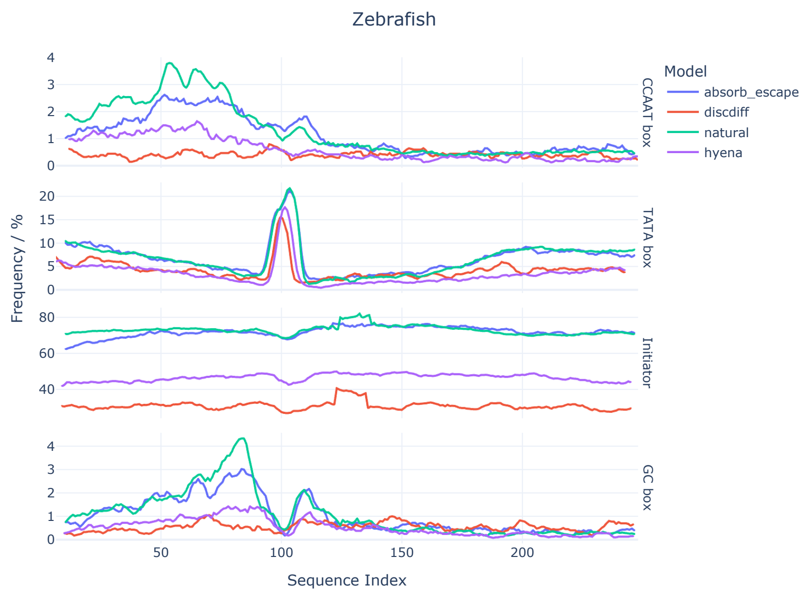


Figure 22: zebrafish

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The sections related to the contributions are listed at the end of the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We talked about the assumption required by the algorithm in the Appendix C, we also talked about other limitations through the paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Can be found in Appendix C

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: For the Toy Example experiment details see Appendix B, for the baseline and main result set up see Appendix D

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See Appendix E for the instructions to access the code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 5.3 and Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the main result which compares the motif distributions across 15 species, we compute the error bar. For some of the other tasks requiring heavy compute, we didn't repeat the experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix D and Section 5.3

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Mainly in the introduction part

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the code repos - The MIT License (MIT) eukaryotic promoter database - CC-BY 4.0

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: See Appendix E

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No Crowdsourcing Involved

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No Crowdsourcing Involved

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.