# DiffusionSELEX

## 1 Introduction

Systematic Evolution of Ligands by Exponential Enrichment (SELEX) is a powerful method for discovering novel ligands with high affinity and specificity for target molecules. However, the experimental process is time-consuming, resource-intensive, and involves numerous parameters that can significantly impact the outcome. This thesis aims to develop an in-silico SELEX simulator using diffusion models, which are a class of generative models that learn to denoise data by iteratively refining a signal through a series of noise-removal steps. By leveraging the power of diffusion models, we can streamline ligand discovery and optimize experimental conditions.

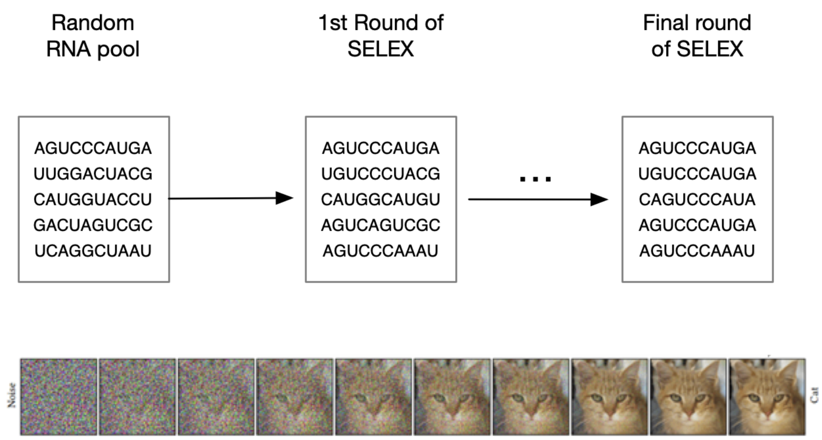


Figure 1: Comparison of a SELEX diffusion approach (top) and a standard image diffusion process (bottom). RNA pools evolve from random (left) to specific (right), with enrichment of specific target-affine sequences. Image diffusion starts with Gaussian noise (left) and progressively denoises to generate a realisitic image.

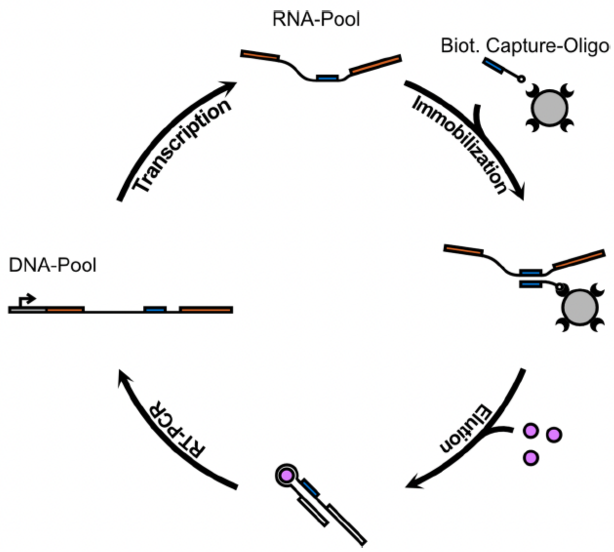


Figure 2: Schematic representation of the SELEX process for aptamer selection. Starting from a random RNA library, iterative rounds of selection, binding, and amplification are performed to enrich for high-affinity aptamers against a specific target molecule. Image from [1].

## 2 Approach

When transferring the process of the SELEX to a diffusion process, the idea is to consider each sequence in the pool of the final round as a sample, analogously to the image of the cat in fig. 1. This sequence should be diffused, i.e. it's entries should randomly change from one round to the next. However, one sequence in one round of the SELEX will also appear unchanged in the previous round. In fact, the additional sequences that exist in previous rounds compared to later rounds, increase the randomness, while the entries of the sequences do not change. This is a problem and we will address this in the following.
To consider the SELEX as a diffusion process, we need to connect a chosen sample from the last round to one sequence of the previous rounds, respectively. The most straightforward way would be to chose the sequence of the previous round that is closest to the current sample. However, this would usually result in identical sequences from the final round to the random pool. In order to increase the randomness, the idea for the construction of the forward pass is the following: Choose a sample $s_0^i$ from the final round, compare it with sequence $s_1^j$, $j \in \{1, ..., N\}$ from the previous pool and make a similarity score for each pair. This similarity score should be transferred to probabilities for the transition of the current sample to the corresponding sequence in the previous pool $p(x_1 = s_1^j | x_0 = s_0^i)$. Then sample $x_1$ from this distribution and do the same for the

next round until the SELEX has reached its first round. With this approach, we have a well-defined forward process for the diffusion model which can be inserted into the diffusion model framework.

# 3 Use sampling without replacement to match the pools

The following procedure is done by image diffusion:

**Algorithm 1** High-level Diffusion Model Training

**Require:** $\mathcal{D}$: dataset of clean images, $T$: number of diffusion steps, $f_\theta$: neural network with parameters $\theta$

**Ensure:** $f_\theta$: neural network

**for** $i = 1, \ldots, \text{num\_epochs}$ **do**
  **for** $x_0 \in \mathcal{D}$ **do** ▷ Iterate over clean images
    $t \sim \text{Uniform}(\{1, \ldots, T\})$ ▷ Sample a random time step
    $x_t \leftarrow \text{ForwardDiffusion}(x_0, t)$ ▷ Apply forward diffusion to obtain a noisy image
    $\epsilon_\theta \leftarrow f_\theta(x_t, t)$ ▷ Predict noise using the neural network
    $\mathcal{L} \leftarrow \text{ComputeLoss}(\epsilon_\theta, \epsilon)$ ▷ Compute the loss between predicted and actual noise
    $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}$ ▷ Update neural network parameters using gradient descent
  **end for**
**end for**
**return** $f_\theta$

Noise needs to be added in several steps to each pixel to obtain nosed images which can be used for training to reconstruct original image estimation from noised images. After training, one can sample pure noise and apply the neural network to create a random image. Experiments show that this is very effective to create new realistic images.

For images, there is a 1:1 relationship from data to noise. I.e. we exactly know to which original image pixel a noise-added pixel corresponds. In our case, having different pools this is not so easy. But an one-to-one relationship can be enforced by removing matched sequences from the pool. This is similar to flow-matching.

# 4 Simple Markov Model Estimation

**Algorithm 2** Forward Process

**Require:** $\mathcal{X} = \{\mathbb{X}_0, \ldots, \mathbb{X}_R\}$: dataset of pools of $R$ rounds, round 0 is random pool, round $R$ is specific pool. Pools containing sequences $\boldsymbol{s}$ together with frequencies $n_{\boldsymbol{s}}$ for each sequence. We denote $n_{\boldsymbol{s}} = n_{\boldsymbol{s},\mathbb{X}_i} \in \mathbb{R}$ the frequency of sequence $\boldsymbol{s}$ (in pool $\mathbb{X}_i$), and $\boldsymbol{n}_{\mathbb{X}_i} \in \mathbb{R}^{N_{\text{unique}}}$ the vector containing all frequencies for all unique sequences in pool $i$. $T$: number of diffusion steps, $f_\theta$: neural network with parameters $\theta$.

**Require:** All pools have the same number of sequences, i.e. $\mathbf{1}^T \boldsymbol{n}_{\mathbb{X}_i} =: N$ is the same for all $i$. Since this is in general not the case, sample $N$ sequences of every pool, weighting each probability of a sequence by its relative frequency in the pool.

**Ensure:** $f_\theta$: trained neural network

**for** $t = 1, \ldots, T$ **do**
    $i \leftarrow$ UNIFORM($\{1, \ldots, R\}$) Pick random round on which to add noise.
    **for** sequence $\boldsymbol{s}$, randomly selected, in pool $\mathbb{X}_i$ **do** ▷ While $\mathbf{1}^T \boldsymbol{n}_{i,\mathbb{X}_i} > 0$
        $\boldsymbol{d}_{\boldsymbol{s},\mathbb{X}_{i-1}} = (d(\boldsymbol{s}, \widetilde{\boldsymbol{s}}))_{\widetilde{\boldsymbol{s}} \in \mathbb{X}_{i-1}}$ ▷ calculate (Levenstein) distances from $\boldsymbol{s}$ to all sequences of pool $i-1$
        $\widetilde{\boldsymbol{p}}_{s,\mathbb{X}_{i-1}} =$ COMPUTESCORE($\boldsymbol{d}_{\boldsymbol{s},\mathbb{X}_{i-1}}$) $\odot\, \boldsymbol{n}_{\mathbb{X}_{i-1}}$ ▷ Calculate probablities proportional to the frequencies of Sequences in pool $i-1$
        $\boldsymbol{p}_{s,\mathbb{X}_{i-1}} = \frac{\widetilde{\boldsymbol{p}}_{s,\mathbb{X}_{i-1}}}{\mathbf{1}^T \widetilde{\boldsymbol{p}}_{s,\mathbb{X}_{i-1}}}$ ▷ Normalize to obtain probalbilities
        $\boldsymbol{s}_{i-1} \leftarrow$ SAMPLE($\mathbb{X}_{i-1}$, $\boldsymbol{p}_{s,\mathbb{X}_{i-1}}$) ▷ Apply forward diffusion by discrete sampling to obtain a noisy sequence from the previous pool
        $n_{\boldsymbol{s}_i,\mathbb{X}_i} \leftarrow n_{\boldsymbol{s}_i,\mathbb{X}_i} - 1$
        $n_{\boldsymbol{s}_{i-1},\mathbb{X}_{i-1}} \leftarrow n_{\boldsymbol{s}_{i-1},\mathbb{X}_{i-1}} - 1$ ▷ Remove matched sequences from the pools to match all sequences from pool $i$ with a sequence from pool $i-1$
        $\widehat{\boldsymbol{s}}_i \leftarrow f_\theta(\boldsymbol{s}_{i-1}, i)$ ▷ Predict original sequence
        $\mathcal{L} \leftarrow$ COMPUTELOSS($\widehat{\boldsymbol{s}}_i$, $\boldsymbol{s}_i$) ▷ Compute the loss between predicted and actual noise
        $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}$ ▷ Update neural network parameters using gradient descent
    **end for**
**end for**
**return** $f_\theta$

**Algorithm 3** Pseudocode for SELEX diffusion sampling

**Ensure:** $f_\theta$: trained neural network

$s_T \sim$ UNIFORM($\{A, C, T, G\}$) Sample random sequence from uniform distribution
**for** $t = T-1, \ldots, 1$ **do**
    $p_t \leftarrow f_\theta(s_{t+1})$ Compute probabilities for each time step given the state at time $t+1$
    $s_t \sim p_t$ Sample the next state for time $t$
**end forreturn** $s_0$