RNAGenScape: Property-Guided, Optimized Generation of mRNA Sequences with Manifold Langevin Dynamics

Danqi Liao[*,1] **Chen Liu**[*,1] **Xingzhi Sun**[*,1] **Dié Tang**[2] **Haochen Wang**[2]
Scott Youlten[2] **Srikar Krishna Gopinath**[2] **Haejeong Lee**[2] **Ethan C. Strayer**[2]
Antonio J. Giraldez[2] ✉ **Smita Krishnaswamy**[1,2,3] ✉

[1]Department of Computer Science, Yale University [2]Department of Genetics, Yale University
[3]Center for Neurocomputation and Machine Intelligence, Wu Tsai Institute [*]Equal contribution
KrishnaswamyLab/RNAGenScape ✉{antonio.giraldez,smita.krishnaswamy}@yale.edu

Abstract

Generating property-optimized mRNA sequences is central to applications such as vaccine design and protein replacement therapy, but remains challenging due to limited data, complex sequence-function relationships, and the narrow space of biologically viable sequences. Generative methods that drift away from the data manifold can yield sequences that fail to fold, translate poorly, or are otherwise nonfunctional. We present RNAGenScape, a property-guided manifold Langevin dynamics framework for mRNA sequence generation that operates on a learned manifold of real data. By performing iterative local optimization constrained to this manifold, RNAGenScape preserves favorable computational viability proxies, accesses reliable guidance, and discourages excursions into regions poorly supported by the training data. The framework integrates three components: ❶ an autoencoder jointly trained with a property predictor to learn a property-organized latent manifold, ❷ a denoising autoencoder that projects updates back onto the manifold, and ❸ a property-guided Langevin dynamics procedure that performs optimization along the manifold. Across three real-world mRNA datasets spanning two orders of magnitude in size, RNAGenScape increases median oracle-predicted property gain by up to 267% and success rate by up to 31%, while maintaining good biological viability proxy measures and achieving competitive inference efficiency compared to existing generative approaches.

1 Introduction

Messenger ribonucleic acids (mRNAs) have emerged as a core modality for modern therapeutics, enabling applications ranging from mRNA vaccine design and protein replacement therapy to gene regulation and synthetic biology [1–4]. In these settings, the nucleotide sequence of an mRNA directly determines properties such as stability, translational efficiency, and ultimately protein yield. Even small sequence edits, particularly in non-coding regions such as the 5' untranslated region (UTR), can lead to substantial changes in degradation rates and ribosome loading [5–7]. As a result, systematic optimization of the mRNA sequence has the potential to improve therapeutic efficacy, reduce dosage requirements, and expand the design space of viable mRNA-based interventions.

Despite its importance, mRNA sequence generation and optimization remains underexplored from a machine learning perspective [8, 9]. A key challenge is that viable mRNA sequences occupy a narrow and structured subset of the astronomically large ambient sequence space [10, 11]. Random or unconstrained edits can lead to nonfunctional transcripts, while experimentally measured datasets are

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

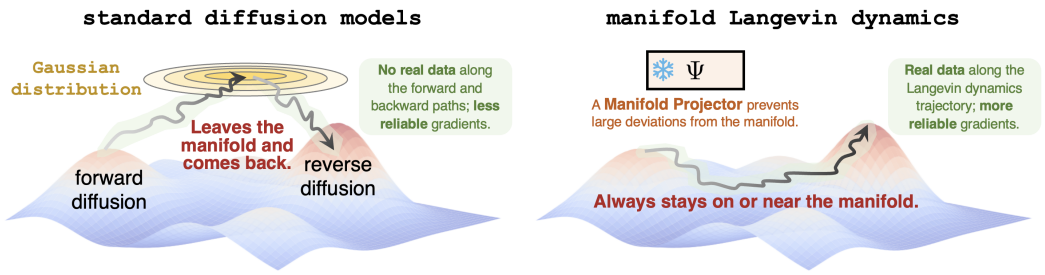


Figure 1: Illustration of manifold Langevin dynamics for optimization on the manifold of real data.

typically small, imbalanced, and sparsely cover the space of interest [12, 13]. Moreover, sequence-function relationships in mRNA are highly nonlinear and context-dependent, making the optimization of mRNA sequences a challenging problem [14, 15].

To address the challenges, we introduce `RNAGenScape`, a property-guided manifold Langevin dynamics framework for mRNA sequence generation built around three key components. ❶ An autoencoder and a property predictor, together termed an organized autoencoder (OAE), that jointly learn a latent manifold of mRNA sequences while organizing this manifold according to the target property. This latent manifold represents the subspace of viable mRNA sequences rather than the much larger Euclidean ambient space. ❷ A manifold projector, implemented as a denoising autoencoder, that maps data onto the learned manifold and prevents drift toward implausible regions. To ensure robustness under sparse and imbalanced data regimes common in mRNA studies, we use SUGAR [16] to fill holes in the data manifold and stabilize manifold projection. ❸ A property-guided manifold Langevin dynamics that iteratively refines sequences by combining gradient guidance with stochastic exploration, while ensuring that sequences generated at every optimization step remain near the learned viable mRNA manifold.

Our choice to perform iterative optimization directly on the learned data manifold, rather than adopting diffusion-based generation from Gaussian noise, is motivated by several practical considerations. **First**, mRNA sequence optimization is inherently local, where improvements are typically made relative to an existing sequence. **Second**, property-guided Langevin dynamics rely on gradients from a predictor trained on real data, and therefore staying close to the data manifold helps obtain stable and meaningful guidance. In contrast, standard diffusion and flow-matching models traverse off-manifold regions during generation, which increases the risk of unreliable guidance from predictors. **Third**, constraining the optimization trajectory to the manifold encourages the generated sequences to remain near regions supported by observed viable sequences, with the aim to promote biological plausibility. **Finally**, refining an existing latent representation avoids a full noise-to-data generation process and supports efficient inference. Figure 1 provides a conceptual illustration.

We evaluate `RNAGenScape` on three real-world mRNA datasets under diverse experimental settings. Across these datasets, `RNAGenScape` improves median property gain by up to 267% over state-of-the-art methods and success rates by up to 31%, while preserving biological viability proxies. In addition, `RNAGenScape` also achieves a competitive inference throughput among the evaluated methods.

2 Preliminaries

2.1 Manifold assumption and manifold learning

The manifold assumption [17–19] is a fundamental principle in machine learning which hypothesizes that high-dimensional natural data lie near a low-dimensional manifold embedded in the ambient space [20]. Formally, each observation $x_i \in \mathbb{R}^n$ arises from a smooth nonlinear map $\mathbf{f} : \mathcal{M}^d \to \mathbb{R}^n$ applied to a latent variable $z_i \in \mathcal{M}^d$, where $d \ll n$.

Manifold learning methods seek to recover this latent structure by constructing representations that preserve intrinsic geometry [21–30]. A point is considered *on-manifold* if it lies within the range of the nonlinear map $\mathbf{f}$, while *off-manifold* points deviate from this structure and may correspond to invalid or adversarial samples [31, 32]. Thus, projecting updated points back to the manifold is critical for robustness and geometry-aware optimization [33].

SUGAR [16] is a diffusion geometry-based method that learns the intrinsic geometry of the data manifold and samples uniformly along the learned geometry. When observed data are limited, augmenting training sets with SUGAR enriches the learned manifold with geometry-preserving samples, particularly in sparse regions.

Stochastic gradient descent (SGD) on Riemannian manifolds [34] extends classical SGD by computing updates in the tangent space and mapping them back to the manifold via exponential maps or retractions. However, such methods assume an analytic form of the manifold. In contrast, the manifold underlying biological sequence space is not known in a closed form. `RNAGenScape` addresses this by directly *learning* the projection operator, enabling optimization on data manifolds without requiring analytic solutions.

2.2 Langevin-dynamics and neural SDEs

Langevin dynamics is a stochastic process that describes motion driven by deterministic forces and thermal fluctuations. Its overdamped form with unit mobility is $\mathrm{d}X_t = -\nabla U(X_t)\,\mathrm{d}t + \sqrt{2\tau}\,\mathrm{d}W_t$, where U is the potential, τ is the temperature, and W_t is Brownian motion. Under mild conditions, the process converges to the stationary distribution with $p \propto e^{-U/\tau}$, and its Euler discretization is often used for sampling from p [35]. In generative modeling with score matching, a neural network $\boldsymbol{s_\theta}$ estimates log-density gradients of Gaussian-perturbed data for annealed Langevin sampling [36].

Neural stochastic differential equations (neural SDEs) [37] are SDEs whose drift and/or diffusion coefficients are parameterized by neural networks. They satisfy $\mathrm{d}X_t = f_\theta(t, X_t)\,\mathrm{d}t + g_\theta(t, X_t)\,\mathrm{d}W_t$, where f_θ is the drift, g_θ is the diffusion coefficient, and W_t is Brownian motion. A neural SDE has a direct connection with Langevin dynamics, as it recovers the Langevin equation above when its drift is $f_\theta = -\nabla U_\theta$ for a neural potential U_θ and its diffusion coefficient is $g_\theta = \sqrt{2\tau}I$.

3 `RNAGenScape`

The key components of our framework are ❶ **OAE**: an autoencoder jointly trained with a prediction model to learn a latent manifold of mRNA sequences organized by the target property (Section 3.1 and Figure 2b), ❷ **Manifold Projector**: a denoising autoencoder that brings the updated latent embeddings back to the learned data manifold during each step of optimization (Section 3.2 and Figure 2c, 2e), and ❸ **Property-guided manifold Langevin dynamics**, a procedure that integrates the two aforementioned modules to enable property-guided generation (Section 3.3 and Figure 2a). Once trained, these components allow `RNAGenScape` to optimize the target property, such as translation efficiency or stability, of a given mRNA sequence.

3.1 Learning a latent manifold organized by property

We begin by training an **organized autoencoder (OAE)**, where the latent manifold is implicitly structured via supervision from a property prediction task (Figure 2b).

Similar to a vanilla autoencoder [38], the encoder $\mathcal{E}$ maps the input mRNA sequence x to a latent representation z, which is decoded by $\mathcal{D}$ back to the sequence space. In addition to this standard architecture, a predictor $\mathcal{P}$ infers properties $\hat{y}$ from the embedding z. Formally, $z = \mathcal{E}(x),\ \hat{x} = \mathcal{D}(z),\ \hat{y} = \mathcal{P}(z)$. The latent manifold $\mathcal{Z}$ is thus shaped by jointly optimizing the reconstruction loss and the prediction loss (Eqn (1)), encouraging it to learn sequence-relevant information while being organized by the target properties. λ_{Pred} and λ_{Recon} are hyperparameters that balance between organizing the property landscape and capturing sequence information. They are empirically set as described in Supplementary Section B. Here, $x_i \in \mathbb{R}^V$ is the ground truth one-hot encoding of the nucleotide at position i in sequence x with an mRNA vocabulary of size V.

$$\mathcal{L}_{\text{OAE}} = \lambda_{\text{Pred}}\mathcal{L}_{\text{Pred}} + \lambda_{\text{Recon}}\mathcal{L}_{\text{Recon}} = \overbrace{\lambda_{\text{Pred}}\ \mathbb{E}_{(x,y)\sim p_{\text{data}}}\|\hat{y} - y\|_2^2}^{\text{optimize } \mathcal{P} \text{ and } \mathcal{E} \text{ with MSE}} \overbrace{- \lambda_{\text{Recon}}\ \mathbb{E}_{x\sim p_{\text{data}}} \frac{1}{L}\sum_{i=1}^{L}\sum_{v=1}^{V} x_{i,v}\log\frac{\exp(\hat{x}_{i,v})}{\sum_{u=1}^{V}\exp(\hat{x}_{i,u})}}^{\text{optimize } \mathcal{E} \text{ and } \mathcal{D} \text{ with CrossEntropy}} \tag{1}$$

3.2 Learning a module to project updates toward the latent manifold

Augmenting the data manifold mRNA datasets are typically scarce and unevenly sampled, which can hinder learning a faithful latent manifold directly from observed data. To mitigate this issue, we

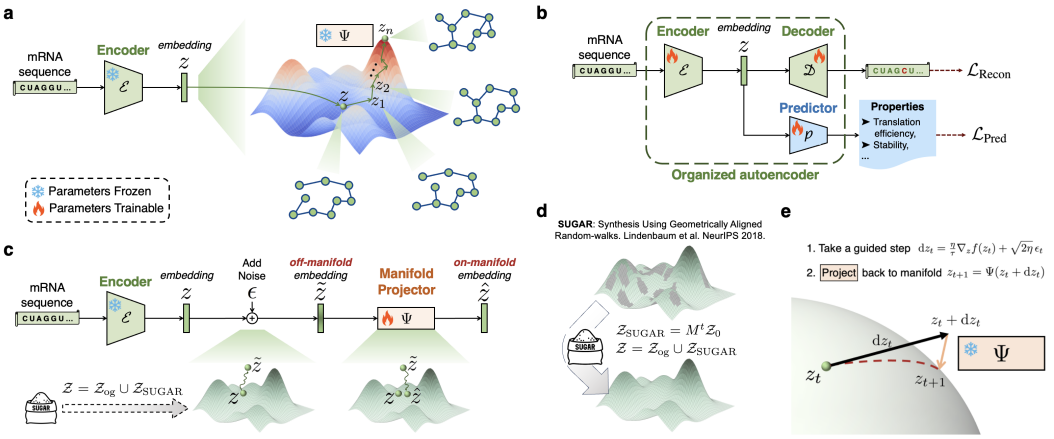


Figure 2: Schematic of `RNAGenScape`. **a,** We encode a given input mRNA sequence into a latent manifold organized by property, and iteratively optimize the sequence along the property landscape. The manifold projector Ψ encourages the optimization to stay on or near the manifold. Notably, we can decode the intermediate optimization result after every step. **b,** The organized latent manifold is created by jointly optimizing reconstruction and property prediction objectives. **c,** The manifold projector is trained to project perturbed data points back to the manifold. **d,** For undersampled mRNA manifolds, we use SUGAR [16] to learn key dimensions in the manifold and fill undersampled regions. **e,** During optimization, the manifold projector brings off-manifold points back to the manifold.

augment the latent embeddings using SUGAR [16], a diffusion geometry-based method introduced in Section 2. This augmentation enriches the learned manifold with geometry-preserving samples, particularly in sparse regions, improving robustness of subsequent manifold projection.

Specifically, this preprocessing step yields an expanded latent set: $\mathcal{Z} = \mathcal{Z}_{\text{og}} \cup \mathcal{Z}_{\text{SUGAR}}$, $\mathcal{Z}_{\text{SUGAR}} = M^t \mathcal{Z}_0$, where $\mathcal{Z}_{\text{og}}$ are the original latent embeddings, $\mathcal{Z}_0$ are locally-sampled neighbors, and M^t is the sparsity-corrected Markov diffusion transition matrix applied for t steps. The SUGAR augmentation is performed in the latent space: we encode training sequences into latent representations, apply SUGAR to enrich this space with geometry-consistent neighbors, and then train the manifold projector.

Training the manifold projector To access reliable property guidance during generation and keep the generated trajectories aligned with the latent data manifold, we introduce a manifold projector Ψ. As illustrated in Figure 2c and magnified in Figure 2e, Ψ takes in a noisy optimized point $\tilde{z}$ and projects it back onto or near the manifold. To train Ψ, we construct a short Gaussian corruption chain from a clean embedding z sampled from the SUGAR-augmented latent set: $\tilde{z}^{(0)} = z$ and $\tilde{z}^{(k)} = \tilde{z}^{(k-1)} + \sigma_k \epsilon_k$, where $\epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, I)$ and σ_k is the noise scale of the k-th corruption step. The projector learns to reverse each local corruption step, yielding the objective in Eqn (2).

$$\mathcal{L}_\Psi = \mathbb{E}_{z \sim p_{\mathcal{Z}},\, \epsilon_{1:K}} \Big[\sum_{k=1}^{K} \big\| \Psi(\tilde{z}^{(k)}) - \tilde{z}^{(k-1)} \big\|_2^2 \Big] \tag{2}$$

Here, $p_{\mathcal{Z}}$ is the empirical distribution of the augmented latent set. When $K = 1$, this objective reduces to the standard denoising autoencoder loss [39], as used in our single-step ablation. We keep K small ($K = 2$) to capture *local denoising transitions near the data manifold*. Pseudocode and implementation details are provided in Supplementary Sections A and B, respectively. At inference, the trained projector is applied once after each property-guided Langevin update.

3.3 Property-guided manifold Langevin dynamics

Next, we combine property-guided Langevin updates with the learned manifold projector to refine mRNA sequences. Given a trained encoder $\mathcal{E}$, a property predictor $\mathcal{P}$, and a manifold projector Ψ, our Langevin-dynamics framework optimizes sequences for a target property. Starting from the latent embedding $z = \mathcal{E}(x)$ of a sequence x, we iteratively update it using a gradient-based drift term $\nabla_z f(z)$, inject Gaussian noise ϵ, and apply a manifold projector $\Psi(\cdot)$ to encourage proximity to the learned data manifold. We define the update rule as described in Eqn (3).

$$z_{t+1} = \Psi(z_t + \mathrm{d}z_t) \qquad \mathrm{d}z_t = \tfrac{\eta}{\tau} \nabla_z f(z_t) + \sqrt{2\eta}\, \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I) \tag{3}$$

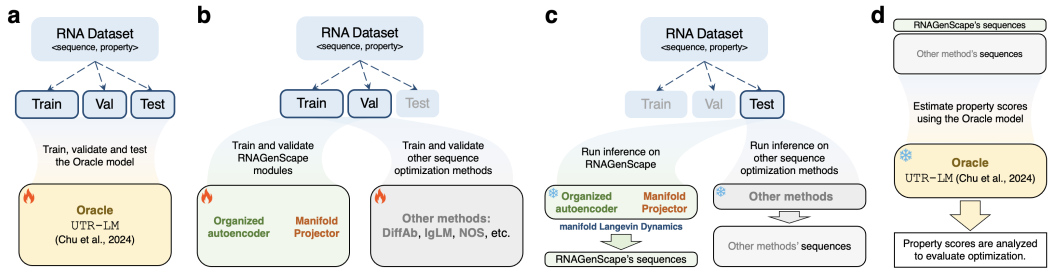


Figure 3: Overview of the experimental design. The same procedure is repeated for every mRNA dataset. **a,** We first fine-tune and evaluate a pre-trained oracle model that predicts target properties from mRNA sequences. **b,** We then train each sequence optimization method using the training and validation sets. **c,** The trained methods are applied to optimize mRNA sequences starting from unseen test samples. **d,** Finally, the oracle model is used to evaluate property improvement and success rate.

Here, η is the step size, τ is the temperature hyperparameter, and $\nabla_z f(z)$ denotes the property gradient given by the predictor $\mathcal{P}$. A smaller τ emphasizes focused updates along the gradient, while a larger τ encourages more diverse exploration. When $\tau \to \infty$, the property guidance vanishes and the update rule is dominated by the stochastic term, which becomes similar to generative modeling with the walkback algorithm [40]. The manifold projector Ψ, analogous to the retraction in Riemannian SGD [34], is applied after each update to ensure that each step remains near the learned latent manifold, enabling interpretable and controllable generation trajectories.

With the trained components $\mathcal{E}$, $\mathcal{P}$ and Ψ, we can optimize the target property of any given sequence. Notably, optimization entails both maximization and minimization: users can choose to increase or decrease the target property, depending on the application.

4 Empirical Results

We evaluate `RNAGenScape` on property-optimized mRNA sequence generation across multiple biologically relevant settings. The goal is to improve target properties such as translation efficiency and stability while preserving the biological viability of the generated sequences. To this end, we conducted comprehensive experiments to analyze the property gains, computational viability proxies, efficiency, and optimization trajectories.

4.1 Experimental settings

Datasets and the properties to optimize We evaluated `RNAGenScape` on three mRNA datasets that capture diverse contexts and experimental designs. The optimization objectives are italicized.

1. `OpenVaccine` contains 2,400 mRNA sequences devised for COVID-19 mRNA vaccines, each with 107 nucleotides [41]. The target property is mRNA *stability*.
2. `Zebrafish` consists of 5 subsets of zebrafish 5' UTR mRNAs each with 124 nucleotides, comprising more than 55,000 sequences in total [42]. The target property is *translation efficiency*, which measures how effectively an mRNA sequence is translated into protein.
3. `Ribosome` is a large-scale dataset of approximately 260,000 5' UTR mRNA sequences, each with 50 nucleotides [43]. The target property is the *mean ribosome load*, defined as the average number of ribosomes bound to the mRNA during translation.

Our main experiments follow the design in Figure 3. We train an oracle sequence-to-property model (panel a), train each optimization method using only the train and validation data (panel b), apply each method to improve sequences initialized from the unseen test set (panel c), and report gains and success rates using the oracle so all methods being compared are scored fairly (panel d). Throughout all evaluations, "property gain/improvement" refer to the oracle's predicted properties.

Baselines For direct comparison on the datasets above, we benchmark `RNAGenScape` against nine baselines for mRNA sequence optimization. DiffAb [44] performs sequence optimization using multinomial diffusion [45]. NOS-C [46] and NOS-D [46] perform property-guided sequence

Table 1: Our proposed RNAGenScape generates mRNA sequences that achieve strong property improvement while maintaining favorable computational viability proxies. $\mathcal{M}$ denotes whether the method takes the manifold structure into consideration. In property optimization results, Δ denotes the median change in property and % denotes success rate (percentage of mRNAs improved). In the biological viability checks, uORF OOF AUG denote out-of-frame start codon in the upstream open reading frame, Kozak similarity is the similarity to the consensus Kozak sequence, and mean pLDDT is a structural-confidence score averaged over nucleotides.

Method	$\mathcal{M}$	Property Optimization				Biological Viability Checks			
		+Property		−Property		Initiation Interference	Initiation Context	Structural Stability	Structural Confidence
		$\Delta\uparrow$	%↑	$\Delta\downarrow$	%↑	uORF OOF AUG %↓	Kozak Similarity %↑	Minimum Free Energy↓	Mean pLDDT↑
OpenVaccine (n≈2k), property to optimize: stability									
Data in the test set	—	—	—	—	—	1.75 ± 0.12	28.70 ± 0.62	−29.82 ± 0.66	42.18 ± 0.24
DiffAb [44]	✗	+0.49 ± 0.09	72.5 ± 3.4	−0.20 ± 0.16	60.1 ± 7.3	1.52 ± 0.55	22.13 ± 5.68	−14.41 ± 3.89	35.57 ± 3.00
IgLM [47]	✗	+0.59 ± 0.02	80.6 ± 0.9	+0.43 ± 0.06	21.4 ± 3.0	2.01 ± 0.16	28.04 ± 2.70	−15.68 ± 0.51	38.00 ± 0.77
NOS-C [46]	✗	+0.10 ± 0.06	61.4 ± 6.7	+0.03 ± 0.00	45.3 ± 3.3	1.77 ± 0.22	29.79 ± 1.54	−24.55 ± 0.37	39.63 ± 0.35
NOS-D [46]	✗	+0.16 ± 0.02	72.2 ± 1.5	−0.06 ± 0.02	60.0 ± 3.0	1.73 ± 0.23	29.41 ± 1.31	−24.11 ± 0.46	39.94 ± 0.70
PCD [50]	✗	+0.16 ± 0.04	56.1 ± 0.4	+0.16 ± 0.05	44.0 ± 0.5	2.67 ± 0.13	37.07 ± 3.49	−15.89 ± 0.17	32.12 ± 0.84
EM [51]	✓	+0.02 ± 0.02	54.2 ± 5.0	−0.03 ± 0.02	55.1 ± 5.0	1.69 ± 0.25	26.49 ± 2.13	−24.36 ± 0.85	38.15 ± 0.47
MFM [48]	✓	+0.31 ± 0.02	69.7 ± 1.7	−0.21 ± 0.07	64.4 ± 4.4	2.08 ± 0.29	33.21 ± 1.97	−17.42 ± 1.06	34.10 ± 0.29
gg-dWJS [49]	✓	+0.52 ± 0.07	70.4 ± 5.2	+0.34 ± 0.02	38.2 ± 3.1	**1.42** ± 0.45	27.01 ± 4.97	− 9.11 ± 4.94	39.73 ± 4.85
MPGD [33]	✓	+0.20 ± 0.05	61.0 ± 4.4	−0.07 ± 0.06	53.9 ± 3.9	2.20 ± 0.51	31.22 ± 4.01	−19.08 ± 1.08	34.71 ± 0.57
RNAGenScape **(ours)**	✓	**+0.92** ± 0.22	**83.6** ± 4.8	**−0.77** ± 0.18	**84.2** ± 2.2	1.54 ± 1.17	**45.60** ± 9.21	**−31.41** ± 5.17	**43.97** ± 3.06
Gain	—	+56%	+4%	+267%	+31%	—	—	—	—
Zebrafish (n≈55k), property to optimize: translation efficiency									
Data in the test set	—	—	—	—	—	2.38 ± 0.06	32.67 ± 0.33	−27.62 ± 0.09	30.42 ± 0.30
DiffAb [44]	✗	+0.38 ± 0.08	61.4 ± 3.0	−0.79 ± 0.08	72.2 ± 2.0	3.57 ± 0.17	37.46 ± 0.42	−20.74 ± 1.19	31.73 ± 0.45
IgLM [47]	✗	−0.01 ± 0.00	49.8 ± 0.1	−0.26 ± 0.04	61.1 ± 0.9	2.72 ± 0.09	36.93 ± 0.50	−26.44 ± 0.08	30.19 ± 0.41
NOS-C [46]	✗	+0.31 ± 0.09	63.5 ± 7.2	−0.14 ± 0.03	65.4 ± 2.5	2.24 ± 0.09	31.21 ± 0.67	−25.53 ± 0.67	31.26 ± 1.18
NOS-D [46]	✗	+0.20 ± 0.04	65.9 ± 4.5	−0.23 ± 0.04	76.2 ± 4.2	2.28 ± 0.04	31.70 ± 0.27	−25.48 ± 0.18	30.98 ± 0.38
PCD [50]	✗	−0.17 ± 0.01	46.0 ± 0.7	−0.18 ± 0.01	54.0 ± 0.7	2.68 ± 0.04	34.44 ± 0.32	−16.60 ± 0.15	32.07 ± 0.51
EM [51]	✓	+0.47 ± 0.05	69.1 ± 3.2	−0.73 ± 0.13	77.0 ± 2.3	1.89 ± 0.04	16.41 ± 1.29	−25.53 ± 1.25	30.94 ± 0.15
MFM [48]	✓	+0.13 ± 0.25	55.0 ± 8.1	−0.36 ± 0.42	66.7 ± 9.4	2.30 ± 0.16	32.71 ± 1.82	−25.20 ± 3.61	31.64 ± 0.10
gg-dWJS [49]	✓	−0.73 ± 0.20	32.1 ± 5.1	−1.14 ± 0.26	77.3 ± 5.7	3.60 ± 0.37	**40.68** ± 1.18	−26.36 ± 3.58	30.81 ± 0.88
MPGD [33]	✓	−0.08 ± 0.22	47.5 ± 7.2	−0.90 ± 0.20	69.5 ± 6.3	2.15 ± 0.41	32.62 ± 2.43	−29.34 ± 2.76	32.54 ± 1.92
RNAGenScape **(ours)**	✓	**+0.79** ± 0.17	**70.9** ± 5.0	**−1.25** ± 0.53	**77.3** ± 9.0	**1.33** ± 0.15	29.68 ± 6.22	**−29.55** ± 1.27	**32.82** ± 1.82
Gain	—	+68%	+3%	+10%	+0%	—	—	—	—
Ribosome (n≈260k), property to optimize: mean ribosome load									
Data in the test set	—	—	—	—	—	3.44 ± 0.00	19.31 ± 0.00	− 8.56 ± 0.00	41.99 ± 0.00
DiffAb [44]	✗	+0.64 ± 0.02	86.2 ± 1.5	−1.10 ± 0.05	77.7 ± 1.6	**1.61** ± 0.21	8.24 ± 0.99	− 6.54 ± 0.24	40.85 ± 0.19
IgLM [47]	✗	+0.26 ± 0.04	66.1 ± 1.1	−0.42 ± 0.02	66.0 ± 0.4	2.37 ± 0.04	25.73 ± 0.48	− 8.20 ± 0.09	41.87 ± 1.26
NOS-C [46]	✗	+0.08 ± 0.01	70.1 ± 2.2	−0.10 ± 0.02	60.5 ± 3.0	3.15 ± 0.07	18.47 ± 0.37	− 7.62 ± 0.03	41.03 ± 0.96
NOS-D [46]	✗	+0.09 ± 0.00	74.8 ± 1.0	−0.12 ± 0.01	73.7 ± 1.7	3.14 ± 0.08	18.77 ± 0.47	− 7.13 ± 0.05	40.45 ± 0.85
PCD [50]	✗	+0.03 ± 0.02	51.5 ± 1.1	+0.04 ± 0.02	48.5 ± 0.9	3.06 ± 0.14	18.43 ± 1.46	− 7.87 ± 0.13	42.70 ± 0.94
EM [51]	✓	+0.08 ± 0.02	68.6 ± 4.9	−0.01 ± 0.02	49.8 ± 6.9	2.64 ± 0.15	15.69 ± 0.49	− 8.24 ± 0.21	42.26 ± 0.87
MFM [48]	✓	+0.01 ± 0.02	46.5 ± 8.4	−0.01 ± 0.02	49.2 ± 9.2	3.35 ± 0.14	19.98 ± 0.27	− 8.48 ± 0.11	41.81 ± 0.57
gg-dWJS [49]	✓	+0.20 ± 0.05	59.0 ± 1.6	−0.07 ± 0.05	52.6 ± 1.8	2.87 ± 0.10	20.80 ± 0.74	− 7.38 ± 0.44	41.44 ± 0.71
MPGD [33]	✓	+0.60 ± 0.03	84.1 ± 0.8	−0.78 ± 0.11	90.7 ± 5.1	1.79 ± 0.10	19.75 ± 0.83	− 9.27 ± 1.83	41.45 ± 0.31
RNAGenScape **(ours)**	✓	**+0.93** ± 0.07	**88.8** ± 3.0	**−1.93** ± 0.57	**92.8** ± 5.7	3.35 ± 0.78	**36.46** ± 9.70	**−10.17** ± 1.12	**44.88** ± 5.59
Gain	—	+45%	+3%	+75%	+2%	—	—	—	—

optimization using continuous and discrete transition distributions, respectively. IgLM [47] refines candidates using an autoregressive language model. MFM [48] generalizes conditional flow matching by replacing Euclidean straight-line interpolants with approximate geodesic interpolants induced by a data-dependent Riemannian metric. gg-dWJS [49] follows gradient guidance from a discriminative property model, using discrete Markov chain to sample on a smoothed data manifold before jumping back to the discrete sequence space through one-step denoising. Persistent Contrastive Divergence (PCD) [50] trains energy-based models by approximating the likelihood gradient using persistent Markov chains. Energy Matching (EM) [51] learns a time-independent scalar potential governing transport toward the data manifold and Langevin sampling. Manifold Preserving Guided Diffusion (MPGD) [33] guides reverse-diffusion sampling through updates constrained to manifold tangent spaces. All these models are trained from scratch on each of the three mRNA datasets with the correct interfaces with RNA A/U/G/C nucleotides (Figure 3b), and then evaluated for mRNA sequence optimization (Figure 3c-d).

All experiments were repeated using five random seeds. Unless otherwise specified, results are reported as the mean ± standard deviation across the five runs. Experimental details and evaluation metrics are provided in Supplementary Sections B and C.

4.2 RNAGenScape generates property-optimized and computationally viable mRNA sequences

Across all three datasets, we evaluate RNAGenScape along two complementary axes: improvement of the target property and preservation of biological viability proxies. Both aspects are jointly reported in Table 1. Property optimization is assessed using UTR-LM [52], a widely adopted mRNA language model, as the external oracle. UTR-LM is first pre-trained on over 480,000 5' UTR sequences from diverse sources and subsequently fine-tuned for property prediction on each dataset (Figure 3a). After fine-tuning, the model is frozen and *never accessed by any method* during their own training and inference (Figure 3b-c), ensuring an unbiased evaluation of the optimized sequences (Figure 3d).

In addition to analyzing the core property optimization performance, we apply a set of complementary checks grounded in established principles of mRNA translation to computationally assess the biological viability of optimized sequences. Specifically, we consider three classes of viability metrics. First, upstream out-of-frame start codons (uORF OOF AUG) are known to suppress downstream protein expression and are therefore undesirable [53]. Second, Kozak similarity [54] measures how closely the initiation context matches the consensus Kozak sequence, with higher values indicating more favorable translation initiation. Third, we characterize RNA structure using minimum free energy [55] as a proxy for structural stability and mean predicted local distance difference test (pLDDT) [56] computed with RhoFold [57] as a proxy for structural confidence. These checks provide supporting evidence of biological plausibility, but a more reliable assessment requires wet-lab validation.

Across all three datasets and both optimization directions, RNAGenScape achieves the largest median property change and the highest success rate (Table 1). Compared to the evaluated baselines, RNAGenScape attains up to a 267% increase in median property improvement and up to a 31% increase in success rate. In addition to better optimization performance, RNAGenScape generally produces sequences with relatively few aberrant upstream start codons, higher Kozak similarity, and more favorable minimum free energy profiles. Moreover, the viability metrics of RNAGenScape-generated sequences closely match those of real sequences in the unseen test set, indicating that substantial property gains can be achieved without compromising biological viability as evaluated using the computational viability proxies.

Table 2: Reproducing results in Table 1 using third-party checkpoints which fine-tuned UTR-LM on external data as the oracles for translation efficiency (UTR-LM-TE) and ribosome loading (UTR-LM-MRL). These oracles are not fine-tuned on our datasets. Despite potential distribution shifts that limit direct comparison of predicted property values with those in Table 1, consistent gains suggest that our improvements are not solely due to shared predictor-oracle biases.

Method	$\mathcal{M}$	Property Optimization			
		+Property		−Property	
		Δ ↑	% ↑	Δ ↓	% ↑
Zebrafish (n≈55k), property to optimize: translation efficiency					
DiffAb [44]	✗	+0.25 ± 0.04	82.7 ± 3.0	+0.00 ± 0.03	49.6 ± 4.4
IgLM [47]	✗	+0.03 ± 0.00	56.5 ± 0.6	−0.01 ± 0.00	52.5 ± 1.0
NOS-C [46]	✗	+0.07 ± 0.02	77.6 ± 8.4	−0.04 ± 0.04	67.2 ± 14.8
NOS-D [46]	✗	+0.06 ± 0.00	80.4 ± 1.2	−0.12 ± 0.02	72.5 ± 4.4
PCD [50]	✗	−0.15 ± 0.02	29.4 ± 1.3	−0.15 ± 0.02	70.6 ± 1.3
EM [51]	✓	+0.06 ± 0.05	65.5 ± 12.0	−0.03 ± 0.05	56.4 ± 12.9
MFM [48]	✓	+0.09 ± 0.14	62.6 ± 24.4	−0.04 ± 0.11	55.4 ± 25.9
gg-dWJS[49]	✓	+0.06 ± 0.15	55.7 ± 15.8	−0.10 ± 0.17	58.8 ± 17.7
MPGD [33]	✓	−0.08 ± 0.10	38.8 ± 13.8	−0.20 ± 0.11	75.8 ± 12.5
RNAGenScape **(ours)**	✓	**+0.40** ± 0.04	**84.5** ± 3.7	**−0.25** ± 0.04	**76.1** ± 3.2
Ribosome (n≈260k), property to optimize: mean ribosome load					
DiffAb [44]	✗	+0.50 ± 0.05	75.0 ± 2.1	−0.19 ± 0.05	59.5 ± 1.7
IgLM [47]	✗	+0.10 ± 0.01	55.5 ± 0.3	−0.08 ± 0.01	54.2 ± 0.6
NOS-C [46]	✗	+0.20 ± 0.02	76.1 ± 1.7	+0.03 ± 0.04	45.9 ± 4.6
NOS-D [46]	✗	+0.26 ± 0.01	88.5 ± 0.9	−0.27 ± 0.01	75.4 ± 2.3
PCD [50]	✗	+0.01 ± 0.04	50.4 ± 2.0	−0.00 ± 0.04	49.9 ± 1.9
EM [51]	✓	+0.06 ± 0.04	61.1 ± 7.6	+0.01 ± 0.04	45.0 ± 9.4
MFM [48]	✓	−0.00 ± 0.01	41.0 ± 4.7	−0.03 ± 0.04	52.1 ± 9.2
gg-dWJS[49]	✓	+0.29 ± 0.13	63.1 ± 5.8	+0.02 ± 0.10	49.0 ± 4.4
MPGD [33]	✓	+0.33 ± 0.13	63.9 ± 6.2	−0.63 ± 0.27	76.8 ± 7.9
RNAGenScape **(ours)**	✓	**+2.26** ± 0.47	**96.8** ± 3.8	**−0.65** ± 0.16	**77.9** ± 4.3

An additional evaluation with third-party oracles fine-tuned on disjoint external data also shows that RNAGenScape achieves the largest median property gains and success rate on Zebrafish and Ribosome in both optimization directions (Table 2), demonstrating robustness across evaluation oracles. See Supplementary Section C for details on these third-party oracles.

4.3 Held-out validation on unseen near-optimal sequences

Table 3: Mean normalized Hamming distance between generated sequences and their nearest favorable-property test-set references. Distances lie in $[0, 1]$, with lower values being better.

	DiffAb [44]	IgLM [47]	NOS-C [46]	NOS-D [46]	PCD [50]	EM [51]	MFM [48]	gg-dWJS [49]	MPGD [33]	RNAGenScape **(ours)**
OpenVaccine (n≈2k)	0.424 ± 0.015	0.583 ± 0.005	0.421 ± 0.001	0.417 ± 0.001	0.546 ± 0.001	0.416 ± 0.002	0.433 ± 0.009	0.479 ± 0.048	0.412 ± 0.002	**0.411** ± 0.023
Zebrafish (n≈55k)	0.643 ± 0.002	0.642 ± 0.001	0.637 ± 0.004	0.635 ± 0.004	0.687 ± 0.001	0.653 ± 0.004	0.640 ± 0.003	0.638 ± 0.003	0.642 ± 0.000	**0.632** ± 0.001
Ribosome (n≈260k)	0.501 ± 0.001	0.509 ± 0.000	0.502 ± 0.001	0.497 ± 0.001	0.524 ± 0.000	0.510 ± 0.002	0.509 ± 0.003	0.504 ± 0.002	0.513 ± 0.001	**0.491** ± 0.005

As a supplementary consistency check, we assess whether generated candidates resemble held-out sequences with favorable oracle-predicted properties. We retain the standard train/validation/test splits and optimize sequences sampled from the test set. During evaluation, we select a subset from the test set whose properties are more than one standard deviation better than the population mean. This provides a curated set of favorable-property references unseen during training without changing the main experimental setup. Then, we report the mean normalized Hamming distance from generated candidates to their nearest sample in the favorable-property reference set.

Table 4: Reconstruction and prediction objectives in organized autoencoder are constructive and complementary.

	$\mathcal{L}_{\text{Recon}}$	$\mathcal{L}_{\text{Pred}}$	Reconstruction	Property Prediction	
			Nucleotide Acc. ↑	Pearson Corr. ↑	Spearman Corr. ↑
OpenVaccine (n≈2k)	✓	✗	**0.92** ± 0.00	0.02 ± 0.08	0.01 ± 0.08
	✗	✓	0.14 ± 0.03	0.65 ± 0.09	0.72 ± 0.08
	✓	✓	**0.92** ± 0.01	**0.68** ± 0.06	**0.75** ± 0.04
Zebrafish (n≈55k)	✓	✗	**0.93** ± 0.01	0.04 ± 0.01	0.04 ± 0.02
	✗	✓	0.13 ± 0.02	**0.76** ± 0.00	**0.75** ± 0.01
	✓	✓	0.89 ± 0.00	0.73 ± 0.01	0.73 ± 0.01
Ribosome (n≈260k)	✓	✗	**1.00** ± 0.00	0.02 ± 0.01	0.02 ± 0.01
	✗	✓	0.15 ± 0.02	**0.92** ± 0.00	**0.90** ± 0.00
	✓	✓	**1.00** ± 0.00	**0.92** ± 0.00	**0.90** ± 0.00

Table 5: The manifold projector Ψ substantially improves property optimization by enforcing optimization along the data manifold.

	Ψ	+property		−property	
		Δ ↑	% ↑	Δ ↓	% ↑
OpenVaccine (n≈2k)	✗	0.26 ± 0.09	63.5 ± 1.2	−0.69 ± 0.17	79.6 ± 2.0
	✓	**0.92** ± 0.22	**83.6** ± 4.8	**−0.77** ± 0.18	**84.2** ± 2.2
Zebrafish (n≈55k)	✗	0.77 ± 0.25	68.3 ± 8.9	−1.05 ± 0.48	75.0 ± 9.7
	✓	**0.79** ± 0.17	**70.9** ± 5.0	**−1.25** ± 0.53	**77.3** ± 9.0
Ribosome (n≈260k)	✗	0.58 ± 0.12	81.1 ± 6.9	−0.58 ± 0.24	67.1 ± 6.9
	✓	**0.93** ± 0.07	**88.8** ± 3.0	**−1.93** ± 0.57	**92.8** ± 5.7

Table 6: Ablation study on the data augmentation ratio of SUGAR.

	OpenVaccine (n≈2k)				Zebrafish (n≈55k)				Ribosome-loading (n≈260k)			
	+property		−property		+property		−property		+property		−property	
SUGAR ratio	Δ ↑	% ↑	Δ ↓	% ↑	Δ ↑	% ↑	Δ ↓	% ↑	Δ ↑	% ↑	Δ ↓	% ↑
0	+0.44	73.2	−0.97	**86.2**	+0.14	54.7	−0.98	73.0	+0.83	83.7	−0.74	67.9
0.01	+0.08	56.9	−0.62	67.5	+0.69	70.7	−1.20	**80.0**	+0.53	78.3	−0.20	51.3
0.1	+0.20	58.1	−0.51	74.0	+0.65	70.8	−0.93	75.3	+0.49	69.6	−1.18	71.4
1.0	**+0.92**	**83.6**	−0.77	84.2	**+0.79**	**70.9**	**−1.25**	77.3	**+0.93**	**88.8**	**−1.93**	**92.8**
10.0	−0.13	57.1	**−1.18**	80.8	−0.07	47.9	−1.10	78.9	+0.29	63.1	−1.44	82.3

Table 3 reports these distances. Across all datasets, RNAGenScape consistently achieves the lowest Hamming distance compared to alternative biological sequence generation methods. This consistency check provides complementary and indirect evidence that generated candidates resemble real sequences with favorable properties.

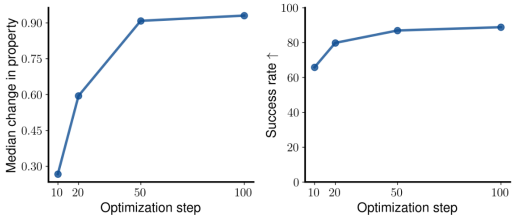


Figure 4: RNAGenScape optimization is step-efficient and remains stable over a range of optimization steps.

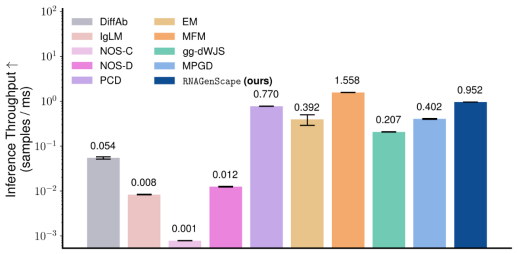


Figure 5: Inference throughput measured using CUDA-synchronized wall-clock time.

4.4 Ablation studies

Organized autoencoder Since the optimization process in RNAGenScape is performed in the learned latent manifold, we verify that the OAE provides both accurate reconstruction and property prediction (Table 4). Additionally, we demonstrate that the reconstruction and prediction objectives ($\mathcal{L}_{\text{Recon}}$ and $\mathcal{L}_{\text{Pred}}$) are generally constructive to each other, which justifies our OAE design.

Manifold projector Next, we show that the manifold projector Ψ is essential for the optimization performance of RNAGenScape (Table 5). Without Ψ, optimization becomes substantially less effective across datasets. This behavior is expected: updating latent representations solely by following the property gradient allows trajectories to drift away from the learned data manifold, entering regions that are weakly supported by real sequences. In these off-manifold regions, the predictor is less constrained by training data, which may contribute to diminished optimization gains. Further analysis of predictor behavior in different latent regions and ablations of projector design as well as reconstruction weight are provided in Supplementary Sections D and E, respectively.

SUGAR augmentation Table 6 shows that the default SUGAR ratio improves property optimization in most settings, while excessive augmentation can reduce performance.

Optimization steps We further analyze how sensitive RNAGenScape is to the number of optimization steps. The results in Figure 4 show that our method is able to converge quickly and remains stable over a range of optimization steps during Langevin dynamics updates.

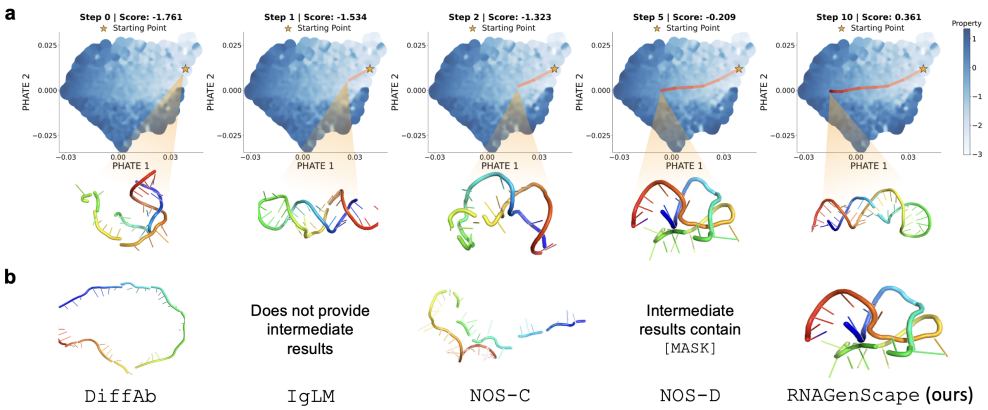


Figure 6: A sample latent space trajectory of RNAGenScape over 10 optimization steps. The trajectory follows a smooth path with steady improvement in the target property. **a,** The trajectory in the PHATE space and the corresponding 3D structures. **b,** Intermediate results of RNAGenScape and competing methods midway through property optimization (at the 5th step out of 10 steps).

***De novo* generative methods and classic optimization algorithms** The results in Supplementary Section F show that (1) *de novo* generative models exhibit limited property control and (2) classic optimization strategies in the OAE latent manifold also underperform RNAGenScape, indicating that our manifold Langevin dynamics contributed substantially to the favorable performance.

4.5 Efficiency and scalability

In addition to its other merits, RNAGenScape is also highly efficient at inference time. As reported in Figure 5, it achieves a throughput of 0.952 samples/ms, second only to MFM at 1.558 samples/ms. This efficiency makes RNAGenScape well suited for large-scale or iterative design workflows where high throughput is essential.

4.6 Case study: RNAGenScape produces structured, data-aligned optimization trajectories

To illustrate how RNAGenScape iteratively navigates along a learned latent manifold, we visualize a sample optimization run in Figure 6a. The trajectory exhibits monotonic increases in the target property, while remaining near regions populated by real sequences. Supplementary Sections H and G provide further trajectories and sequence interpolation results, respectively. These trajectories are direct consequences of the manifold-constrained dynamics, which guided each step toward high-property regions while staying on the manifold. As a qualitative illustration of biological plausibility, we show that the intermediate results of RNAGenScape optimization can be properly folded by RhoFold [57] into 3D structures (Figure 6b). In contrast, folding intermediate products of several other methods causes failure, as indicated by fragmented structures (Figure 6b).

5 Related Works

Sequence-to-function modeling Predicting quantitative properties such as expression and stability from biological sequences is central to sequence design [59]. Models trained on high-throughput experimental data capture sequence-function relationships in regulatory regions such as 5' UTRs and promoters [43, 60]. ConvNets [61] and Transformers [62] capture dependencies in mRNA sequences and provide property predictors for downstream optimization.

Generative models for design *De novo* generative models create novel biological sequences using VAEs [63, 64, 5], adversarial models [65–68], autoregressive language models [69], and diffusion models [70, 71]. ProteinMPNN [72] generates sequences conditioned on protein structures, while RFdiffusion [71] generates protein structures with or without design constraints. These approaches provide foundations for biological design; our setting emphasizes property-guided optimization of an existing mRNA sequence.

Optimization of biological sequences Classic gradient-based and sampling-based optimization strategies [73–77] can improve known sequences, but do not inherently constrain intermediate

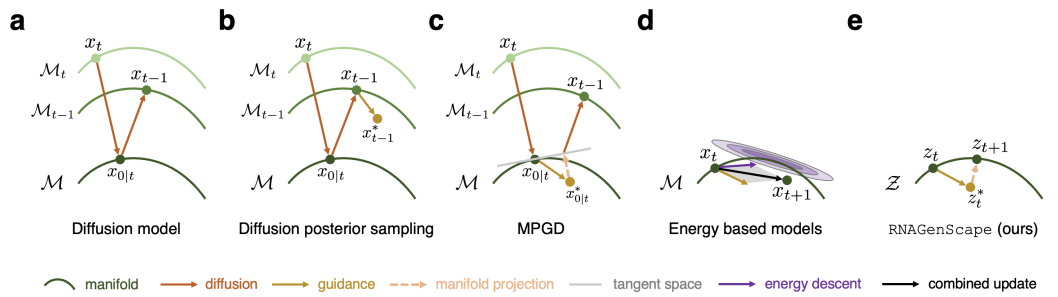


Figure 7: Conceptual comparison of sampling and guidance mechanisms relevant to our property-guided optimization problem. **a,** Unguided reverse diffusion. **b,** Diffusion posterior sampling [58] adds guidance to the reverse diffusion. **c,** Manifold Preserving Guided Diffusion (MPGD) [33] constrains guidance through a local manifold projection before reverse diffusion. **d,** Guided Langevin sampling in energy-based models [51] combine guidance with energy descent; contours depict the learned energy landscape. **e,** `RNAGenScape` alternates property-guided updates and a SUGAR-assisted denoising projection in the OAE latent space. The distinct projection mechanisms used by MPGD and `RNAGenScape` are represented in the same color since they share the same high-level concept, whereas implementation details and stochastic perturbations are omitted from the illustration.

variants to the biological data distribution. Differentiable sequence optimization uses continuous relaxations [78, 79] or learned latent representations, as in ReLSO [80]. Other approaches use reinforcement learning [81], Monte Carlo search [82], or multi-objective Bayesian optimization [83]. Deep generative methods such as DiffAb [44], IgLM [47], and NOS [46] further connect sequence generation with design objectives. Our framework combines property-guided optimization with a learned manifold correction at each optimization step.

Energy-based and manifold-guided methods Energy-based and manifold-aware methods provide precedents for balancing property improvement with consistency with a learned data distribution. Models trained with Persistent Contrastive Divergence (PCD) [50] and Energy Matching (EM) [51] provide energy priors that can be combined with property objectives. Metric Flow Matching (MFM) [48] incorporates geometry into transport paths, while gradient-guided discrete walk-jump sampling (gg-dWJS) [49] couples property guidance with sampling and denoising in a smoothed sequence space. Manifold Preserving Guided Diffusion (MPGD) [33] addresses guidance-induced departures from the manifold within reverse diffusion. `RNAGenScape` shares these motivations but combines property-organized mRNA embeddings with a denoising map trained on SUGAR-augmented embeddings, applied after each property-guided Langevin update without diffusion. Our contribution is an mRNA optimization framework with manifold learning and projection mechanisms distinct from those in prior works, evaluated computationally on real-world datasets of varying sizes. Figure 7 illustrates the mechanistic distinctions.

Integration of structural context Structural context provides additional information for biological sequence modeling. ProteinMPNN [72] and diffusion-based inverse folding [84] condition sequence generation on 3D structures. ImmunoStruct [85] jointly models protein sequence, structure, and biochemical properties to predict immunogenicity. CellSpliceNet [86] integrates sequence, secondary structure, and gene expression to predict alternative splicing. EternaFold [87] also models RNA secondary structure. Incorporating structural context is a potential extension to `RNAGenScape`.

6 Conclusion

We introduced `RNAGenScape`, a manifold Langevin dynamics framework for property-optimized mRNA sequence generation. Rather than optimizing in the ambient sequence space, `RNAGenScape` operates directly on a learned manifold of viable mRNA sequences. In this setting, property gradients are more reliable, biological viability is better preserved, and optimization can be performed with lower computational cost. Comprehensive experiments on three real-world mRNA datasets show that our method leads to greater property gains and higher success rates under realistic data regimes. Limitations, future work, and potential impacts are discussed in Supplementary Sections I and J.

Acknowledgements

S.K. is funded by the National Institutes of Health (NIGMSR01GM135929, R01GM130847), the National Science Foundation (NSF CAREER award IIS-2047856, NSF IIS-2403317, NSF DMS-2327211, NSF CISE-2403317) and the National Cancer Institute (1OT2CA319802-01). S.K is also funded by the Sloan Fellowship FG-2021-15883 and the Novo Nordisk grant GR112933. A.J.G. is funded by the National Institutes of Health (R01HD100035, R35GM122580).

References

[1] Norbert Pardi, Michael J Hogan, Frederick W Porter, and Drew Weissman. mrna vaccines—a new era in vaccinology. *Nature reviews Drug discovery*, 17(4):261–279, 2018.

[2] Namit Chaudhary, Drew Weissman, and Kathryn A Whitehead. mrna vaccines for infectious diseases: principles, delivery and clinical translation. *Nature reviews Drug discovery*, 20(11):817–838, 2021.

[3] Shugang Qin, Xiaoshan Tang, Yuting Chen, Kepan Chen, Na Fan, Wen Xiao, Qian Zheng, Guohong Li, Yuqing Teng, Min Wu, et al. mrna-based therapeutics: powerful and versatile tools to combat diseases. *Signal transduction and targeted therapy*, 7(1):166, 2022.

[4] Theofanis Vavilis, Eleni Stamoula, Alexandra Ainatzoglou, Athanasios Sachinidis, Malamatenia Lamprinou, Ioannis Dardalas, and Ioannis S Vizirianakis. mrna in the context of protein replacement therapy. *Pharmaceutics*, 15(1):166, 2023.

[5] Sebastian Castillo-Hair, Stephen Fedak, Ban Wang, Johannes Linder, Kyle Havens, Michael Certo, and Georg Seelig. Optimizing 5'utrs for mrna-delivered gene editing using deep learning. *Nature Communications*, 15(1):5284, 2024.

[6] Qi Ma, Xiaoguang Zhang, Jing Yang, Hongxia Li, Yanzhe Hao, and Xia Feng. Optimization of the 5' untranslated region of mrna vaccines. *Scientific Reports*, 14(1):19845, 2024.

[7] Ting Li, Gangfeng Liu, Guolong Bu, Yien Xu, Caiyun He, and Gexin Zhao. Optimizing mrna translation efficiency through rational 5' utr and 3' utr combinatorial design. *Gene*, 942:149254, 2025.

[8] Sebastian M Castillo-Hair and Georg Seelig. Machine learning for designing next-generation mrna therapeutics. *Accounts of chemical research*, 55(1):24–34, 2021.

[9] Niels Schlusser, Asier González, Muskan Pandey, and Mihaela Zavolan. Current limitations in predicting mrna translation with deep learning models. *Genome Biology*, 25(1):227, 2024.

[10] Francesco Calvanese, Camille N Lambert, Philippe Nghe, Francesco Zamponi, and Martin Weigt. Towards parsimonious generative modeling of rna families. *Nucleic Acids Research*, 52(10):5465–5477, 2024.

[11] He Zhang, Liang Zhang, Ang Lin, Congcong Xu, Ziyu Li, Kaibo Liu, Boxiang Liu, Xiaopin Ma, Fanfan Zhao, Huiling Jiang, et al. Algorithm for optimized mrna design improves stability and immunogenicity. *Nature*, 621(7978):396–403, 2023.

[12] Oskar Taubert, Fabrice von der Lehr, Alina Bazarova, Christian Faber, Philipp Knechtges, Marie Weiel, Charlotte Debus, Daniel Coquelin, Achim Basermann, Achim Streit, et al. Rna contact prediction by data efficient deep learning. *Communications biology*, 6(1):913, 2023.

[13] Muhammad Nabeel Asim, Muhammad Ali Ibrahim, Tayyaba Asif, and Andreas Dengel. Rna sequence analysis landscape: A comprehensive review of task types, databases, datasets, word embedding methods, and language models. *Heliyon*, 11(2), 2025.

[14] Donny D Licatalosi and Robert B Darnell. Resolving rna complexity to decipher regulatory rules governing biological networks. *Nature Reviews. Genetics*, 11(1):75, 2010.

[15] Caleb Weinreb, Adam J Riesselman, John B Ingraham, Torsten Gross, Chris Sander, and Debora S Marks. 3d rna and functional interactions from evolutionary couplings. *Cell*, 165(4):963–975, 2016.

[16] Ofir Lindenbaum, Jay S. Stanley III, Guy Wolf, and Smita Krishnaswamy. Geometry-based data generation, 2018.

[17] Lawrence Cayton et al. *Algorithms for manifold learning*. eScholarship, University of California, 2008.

[18] Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. *Advances in neural information processing systems*, 23, 2010.

[19] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[20] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[21] David Van Dijk, Roshan Sharma, Juozas Nainys, Kristina Yim, Pooja Kathail, Ambrose J Carr, Cassandra Burdziak, Kevin R Moon, Christine L Chaffer, Diwakar Pattabiraman, et al. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 174(3):716–729, 2018.

[22] Kevin R Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B Burkhardt, William S Chen, Kristina Yim, Antonia van den Elzen, Matthew J Hirn, Ronald R Coifman, et al. Visualizing structure and transitions in high-dimensional biological data. *Nature biotechnology*, 37(12):1482–1492, 2019.

[23] Daniel B Burkhardt, Jay S Stanley III, Alexander Tong, Ana Luisa Perdigoto, Scott A Gigante, Kevan C Herold, Guy Wolf, Antonio J Giraldez, David van Dijk, and Smita Krishnaswamy. Quantifying the effect of experimental perturbations at single-cell resolution. *Nature biotechnology*, 39(5):619–629, 2021.

[24] Chen Liu, Matthew Amodio, Liangbo L. Shen, Feng Gao, Arman Avesta, Sanjay Aneja, Jay C. Wang, Lucian V. Del Priore, and Smita Krishnaswamy. CUTS: A Deep Learning and Topological Framework for Multigranular Unsupervised Medical Image Segmentation . In *proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*, volume LNCS 15008. Springer Nature Switzerland, October 2024.

[25] Danqi Liao, Chen Liu, Benjamin W Christensen, Alexander Tong, Guillaume Huguet, Guy Wolf, Maximilian Nickel, Ian Adelstein, and Smita Krishnaswamy. Assessing neural network representations during training using noise-resilient diffusion spectral entropy. In *2024 58th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2024.

[26] Chen Liu, Danqi Liao, Alejandro Parada-Mayorga, Alejandro Ribeiro, Marcello DiStasio, and Smita Krishnaswamy. Diffkillr: Killing and recreating diffeomorphisms for cell annotation in dense microscopy images. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025.

[27] Chen Liu, Ke Xu, Liangbo L Shen, Guillaume Huguet, Zilong Wang, Alexander Tong, Danilo Bzdok, Jay Stewart, Jay C Wang, Lucian V Del Priore, and Smita Krishnaswamy. Imageflownet: Forecasting multiscale trajectories of disease progression with irregularly-sampled longitudinal medical images. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025.

[28] Xingzhi Sun, Danqi Liao, Kincaid MacDonald, Yanlei Zhang, Chen Liu, Guillaume Huguet, Guy Wolf, Ian Adelstein, Tim GJ Rudner, and Smita Krishnaswamy. Geometry-aware generative autoencoders for warped riemannian metric learning and generative modeling on data manifolds. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2025.

[29] Alexander Tong, Manik Kuchroo, Shabarni Gupta, Aarthi Venkat, Beatriz P San Juan, Laura Rangel, Brandon Zhu, John G Lock, Christine L Chaffer, and Smita Krishnaswamy. Learning transcriptional and regulatory dynamics driving cancer cell plasticity using neural ode-based optimal transport. *bioRxiv*, pages 2023–03, 2023.

[30] Yanlei Zhang, Guillaume Huguet, Edward De Brouwer, Danqi Liao, Oluwadamilola Fasina, Alexander Tong, Ricky TQ Chen, Guy Wolf, Maximilian Nickel, Ian Adelstein, et al. Neural fim: Bridging statistical manifolds and generative modeling through fisher geometry. *Authorea Preprints*, 2025.

[31] Salah Rifai, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. *Advances in neural information processing systems*, 24, 2011.

[32] Qian Li, Yuxiao Hu, Ye Liu, Dongxiao Zhang, Xin Jin, and Yuntian Chen. Discrete point-wise attack is not enough: Generalized manifold adversarial attack for face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20575–20584, 2023.

[33] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, WeiHsiang Liao, Yuki Mitsufuji, Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. In *International Conference on Learning Representations*, volume 2024, pages 44819–44850, 2024.

[34] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, September 2013.

[35] Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 1996.

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

[37] Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International conference on machine learning*, pages 5453–5463. PMLR, 2021.

[38] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[39] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[40] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models, 2013.

[41] Rhiju Das, H Wayment-Steele, Christian Choe Do Soon Kim, Bojan Tunguz, Walter Reade, and Maggie Demkin. Openvaccine: Covid-19 mrna vaccine degradation prediction. *URL https://kaggle.com/competitions/stanford-covid-vaccine*, 2020.

[42] Ethan C Strayer, Srikar Krishna, Haejeong Lee, Charles Vejnar, Nils Neuenkirchen, Amit Gupta, Jean-Denis Beaudoin, and Antonio J Giraldez. Nap-trap reveals the regulatory grammar in 5'utr-mediated translation regulation during zebrafish development. *Nature Communications*, 15(1):10898, 2024.

[43] Paul J Sample, Ban Wang, David W Reid, Vlad Presnyak, Iain J McFadyen, David R Morris, and Georg Seelig. Human 5' utr design and variant effect prediction from a massively parallel translation assay. *Nature biotechnology*, 37(7):803–809, 2019.

[44] Shitong Luo, Yufeng Su, Xingang Peng, Sheng Wang, Jian Peng, and Jianzhu Ma. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. *Advances in Neural Information Processing Systems*, 35:9754–9767, 2022.

[45] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

[46] Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in neural information processing systems*, 36:12489–12517, 2023.

[47] Richard W Shuai, Jeffrey A Ruffolo, and Jeffrey J Gray. Iglm: Infilling language modeling for antibody sequence design. *Cell Systems*, 14(11):979–989, 2023.

[48] Kacper Kapuśniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael Bronstein, Avishek J Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold. *Advances in Neural Information Processing Systems*, 37:135011–135042, 2024.

[49] Zarif Ikram, Dianbo Liu, and M Saifur Rahman. Gradient-guided discrete walk-jump sampling for biological sequence generation. *Transactions on Machine Learning Research*, 2024.

[50] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[51] Michal Balcerak, Tamaz Amiranashvili, Antonio Terpin, Suprosanna Shit, Lea Bogensperger, Sebastian Kaltenbach, Petros Koumoutsakos, and Bjoern Menze. Energy matching: Unifying flow matching and energy-based models for generative modeling. In *Advances in Neural Information Processing Systems*, 2025.

[52] Yanyi Chu, Dan Yu, Yupeng Li, Kaixuan Huang, Yue Shen, Le Cong, Jason Zhang, and Mengdi Wang. A 5útr language model for decoding untranslated regions of mrna and function predictions. *Nature Machine Intelligence*, 6(4):449–460, 2024.

[53] Thomas E Dever, Ivaylo P Ivanov, and Alan G Hinnebusch. Translational regulation by uorfs and start codon selection stringency. *Genes & development*, 37(11-12):474–489, 2023.

[54] Alec C Gleason, Ghanashyam Ghadge, Jin Chen, Yoshifumi Sonobe, and Raymond P Roos. Machine learning predicts translation initiation sites in neurologic diseases with nucleotide repeat expansions. *PLoS One*, 17(6):e0256411, 2022.

[55] Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for molecular biology*, 6(1):26, 2011.

[56] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[57] Tao Shen, Zhihang Hu, Siqi Sun, Di Liu, Felix Wong, Jiuming Wang, Jiayang Chen, Yixuan Wang, Liang Hong, Jin Xiao, et al. Accurate rna 3d structure prediction using a language model-based deep learning approach. *Nature Methods*, 21(12):2287–2298, 2024.

[58] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.

[59] Stephen G Oliver. From dna sequence to biological function. *Nature*, 379(6566):597–600, 1996.

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

[61] Yifan Chen, Zhenya Du, Xuanbai Ren, Chu Pan, Yangbin Zhu, Zhen Li, Tao Meng, and Xiaojun Yao. mrna-cla: An interpretable deep learning approach for predicting mrna subcellular localization. *Methods*, 227:17–26, 2024.

[62] Shujun He, Baizhen Gao, Rushant Sabnis, and Qing Sun. Rnadegformer: accurate prediction of mrna degradation at nucleotide resolution with deep learning. *Briefings in Bioinformatics*, 24(1):bbac581, 2023.

[63] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[64] Sam Sinai, Eric Kelsic, George M Church, and Martin A Nowak. Variational auto-encoding of protein sequences. *arXiv preprint arXiv:1712.03346*, 2017.

[65] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[66] Oleksii Prykhodko, Simon Viet Johansson, Panagiotis-Christos Kotsias, Josep Arús-Pous, Esben Jannik Bjerrum, Ola Engkvist, and Hongming Chen. A de novo molecular generation method using latent vector based generative adversarial network. *Journal of cheminformatics*, 11(1):74, 2019.

[67] Oscar Méndez-Lucio, Benoit Baillif, Djork-Arné Clevert, David Rouquié, and Joerg Wichard. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nature communications*, 11(1):10, 2020.

[68] Zachary Wu, Kadina E Johnston, Frances H Arnold, and Kevin K Yang. Protein sequence design with deep generative models. *Current opinion in chemical biology*, 65:18–27, 2021.

[69] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature biotechnology*, 41(8):1099–1106, 2023.

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

[71] 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*, 620(7976):1089–1100, 2023.

[72] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[73] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[74] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.

[75] Stephen Brooks. Markov chain monte carlo method and its application. *Journal of the royal statistical society: series D (the Statistician)*, 47(1):69–100, 1998.

[76] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50:5–43, 2003.

[77] Bart Selman and Carla P Gomes. Hill-climbing search. *Encyclopedia of cognitive science*, 81(333-335):10, 2006.

[78] Johannes Linder, Nicholas Bogard, Alexander B Rosenberg, and Georg Seelig. Deep exploration networks for rapid engineering of functional dna sequences. *BioRxiv*, page 864363, 2019.

[79] Johannes Linder and Georg Seelig. Fast activation maximization for molecular sequence design. *BMC bioinformatics*, 22:1–20, 2021.

[80] Egbert Castro, Abhinav Godavarthi, Julian Rubinfien, Kevin Givechian, Dhananjay Bhaskar, and Smita Krishnaswamy. Transformer-based protein generation with regularized latent space optimization. *Nature Machine Intelligence*, 4(10):840–851, 2022.

[81] Peter Eastman, Jade Shi, Bharath Ramsundar, and Vijay S Pande. Solving the rna design problem with reinforcement learning. *PLoS computational biology*, 14(6):e1006176, 2018.

[82] Tomasz Wirecki, Grzegorz Lach, Farhang Jaryani, Nagendar Goud Badepally, S Naeim Moafinejad, Gaja Klaudel, and Janusz M Bujnicki. Desirna: structure-based design of rna sequences with a monte carlo approach. *bioRxiv*, pages 2023–06, 2023.

[83] Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating bayesian optimization for biological sequence design with denoising autoencoders. In *International conference on machine learning*, pages 20459–20478. PMLR, 2022.

[84] Kai Yi, Bingxin Zhou, Yiqing Shen, Pietro Liò, and Yuguang Wang. Graph denoising diffusion for inverse protein folding. *Advances in Neural Information Processing Systems*, 36:10238–10257, 2023.

[85] Kevin Bijan Givechian, João Felipe Rocha, Chen Liu, Edward Yang, Sidharth Tyagi, Kerrie Greene, Rex Ying, Etienne Caron, Akiko Iwasaki, and Smita Krishnaswamy. Immunostruct enables multimodal deep learning for immunogenicity prediction. *Nature Machine Intelligence*, pages 1–14, 2025.

[86] Arman Afrasiyabi, Jake Kovalic, Chen Liu, Egbert Castro, Alexis Weinreb, Erdem Varol, David Miller, Marc Hammarlund, and Smita Krishnaswamy. Cellsplicenet: Interpretable multimodal modeling of alternative splicing across neurons in c. elegans. *bioRxiv*, pages 2025–06, 2025.

[87] Hannah K Wayment-Steele, Wipapat Kladwang, Alexandra I Strom, Jeehyung Lee, Adrien Treuille, Alex Becka, Eterna Participants, and Rhiju Das. Rna secondary structure packages evaluated and improved by high-throughput experiments. *Nature methods*, 19(10):1234–1242, 2022.

[88] 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.

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

Technical Appendices

Contents

A Pseudocode of the manifold projector 17

B Hyperparameters and Architecture 18

C Evaluation Metrics 19

D Property Gradient Analysis 20

E Additional Ablation Studies 21

F Additional comparison against de novo baselines 22

G Interpolation between sequences 23

H Additional optimization trajectories 24

I Limitations and Future Work 25

J Impact Statement 25

A Pseudocode of the manifold projector

The pseudocode of the manifold projector is shown in Algorithm 1.

Algorithm 1 Manifold Projector Ψ

Input: Dataset $\mathcal{Z}$, projector Ψ with parameters θ_Ψ, noise scales $\sigma_{1\ldots S}$, epochs E, learning rate η
for $s = 1$ to S **do**
  Initialize AdamW optimizer state q and validation early stopping for this stage.
  for epoch$= 1$ to E or validation early stopping **do**
    for each minibatch $\mathcal{B} \subset \mathcal{Z}$ **do**
      for each $z_i \in \mathcal{B}$ **do**
        $\epsilon_{i,1}, \epsilon_{i,2} \overset{\text{iid}}{\sim} \mathcal{N}(0, I)$
        $\tilde{z}_i^{(1)} \leftarrow z_i + \sigma_s \epsilon_{i,1}, \quad \tilde{z}_i^{(2)} \leftarrow \tilde{z}_i^{(1)} + \sigma_s \epsilon_{i,2}$
        $\mathcal{L}_i \leftarrow \|\Psi(\tilde{z}_i^{(1)}) - z_i\|_2^2 + \|\Psi(\tilde{z}_i^{(2)}) - \tilde{z}_i^{(1)}\|_2^2$
      end for
      $g \leftarrow \nabla_{\theta_\Psi} \left(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{L}_i \right)$
      $(\theta_\Psi, q) \leftarrow \text{AdamWUpdate}(\theta_\Psi, q, g; \eta)$
    end for
  end for
end for

B Hyperparameters and Architecture

Training the organized autoencoder (OAE) We trained the OAE using the AdamW optimizer with an initial learning rate of 10^{-3}, together with a linear warmup cosine annealing scheduler. The learning rate was linearly increased from 10^{-5} to the target value during the first 10% of training epochs (warmup), and then cosine-annealed to 10^{-4} over the remaining epochs. We used a batch size of 128, a maximum of 100 epochs, and early stopping with a patience of 20 epochs.

Training the manifold projector The manifold projector is an MLP with widths $128 \to 32 \to 16 \to 32 \to 128$, ReLU hidden activations, and a linear output. We trained it using AdamW with a learning rate of 10^{-3}. We used a batch size of 128, a maximum of 100 epochs per noise stage, and applied early stopping with a patience of 20 epochs based on the validation loss. The same projector was trained successively at noise scales 0.5, 0.2, and 0.1, with two independent Gaussian corruptions at the current scale for each clean embedding.

Learning the manifold with SUGAR Prior to training the manifold projector Ψ, we hole-filled the OAE-encoded training latent space with SUGAR. To estimate degrees (and thus sparsity), we used the k-NN mode with an α-decay kernel (α=2) and an adaptive bandwidth given by the distance to the 5 nearest neighbors. The diffusion time was set to t=1. This k-NN degree estimator is distinct from the k-NN manifold retractor considered in our projector ablation study.

Table S1: Hyperparameters used for different datasets. All runs use latent dimension 128, denoising autoencoder manifold projector with sugar_w=1, and 100 Langevin steps.

Dataset	λ_{Recon} / λ_{Pred}	Noise levels	Langevin (step size, temperature)
`OpenVaccine`	5.0 / 1.0	{0.5, 0.2, 0.1}	1×10^{-2}, 1×10^{-3}
`Zebrafish`	5.0 / 1.0	{0.5, 0.2, 0.1}	1×10^{-3}, 1×10^{-3}
`Ribosome`	5.0 / 1.0	{0.5, 0.2, 0.1}	1×10^{-3}, 1×10^{-3}

Organized Autoencoder (OAE) The OAE maps mRNA sequences $x \in \mathbb{R}^{L \times V}$ to a compact continuous latent $z \in \mathbb{R}^d$ with d=128 on all datasets.

The encoder uses a `Conv1d` stem followed by two downsampling stages of residual 1D blocks with `GroupNorm`, `SiLU`, and channel squeeze-excitation, doubling channels at each stride-2 stage. Features are pooled with `AdaptiveAvgPool1d` to 16 positional bins and linearly projected to z. The property head is a three-layer MLP with `SiLU` and dropout 0.1.

Decoding expands z to a seed feature map, then applies a stack of `upsample` blocks (linear $\times 2$ + residual `Conv1d`) until length $\geq L$, interpolates to exact L if needed, and refines with two residual `Conv1d` blocks to produce token logits. Weights use Kaiming/Xavier initialization. `GroupNorm` scales are set to 1 and biases to 0.

mRNA sequence vocabulary Although mRNA sequences naturally consist of the nucleotides `A`, `U`, `G`, and `C`, some experimental datasets represent `U` as `T` (borrowing the DNA alphabet). To handle this heterogeneity consistently, we define a unified vocabulary of size $V = 7$: `<pad>`, `A`, `G`, `C`, `T`, `U`, and `N`. Here, `N` denotes an unknown nucleotide during sequencing, and both `U` and `T` tokens are retained to ensure compatibility across datasets. Note that each dataset uniquely uses either `U` or `T`, and since we are training the model separately on each dataset, we do not risk confusing the model with interchangeability during decoding.

Hardware The evaluations were performed on a single NVIDIA A100 GPU. Generation timing covers optimization and decoding; model loading and oracle evaluation are excluded. For `RNAGenScape`, seed encoding is performed before timing.

Baseline adaptation MFM [48] was originally designed for smooth interpolation between populations on a manifold rather than direct property optimization. To adapt MFM to our optimization setting, we partition the training data into property tertiles, as required by the MFM formulation. We then train MFM to model flows from the low-property subpopulation to the high-property

subpopulation for property maximization, and in the reverse direction for property minimization. The trained MFM model is subsequently applied to the unseen test set for property optimization, following the same evaluation protocol used for the other methods.

For PCD [50], we use a multinomial restricted Boltzmann machine with categorical nucleotide variables and a property predictor trained on sequence labels. During optimization, property gradients with respect to one-hot nucleotide representations bias the visible-state logits in Gibbs sampling initialized from the input sequence. For EM [51], we train a sequence autoencoder and learn the energy potential on standardized latent embeddings. Optimization starts from the encoded input sequence and uses Langevin updates guided by a property-target loss evaluated through the decoder and a sequence property predictor. For MPGD [33], we use a sequence autoencoder and a latent diffusion model. We partially noise the encoded input sequence and guide the clean latent estimates during reverse diffusion using a property-target loss through the decoder. These adaptations retain the respective sampling mechanisms while enabling property-directed RNA sequence optimization.

DiffAb [44], IgLM [47], NOS-C [46], NOS-D [46], and gg-dWJS [49] are implemented with the RNA vocabulary and trained on each dataset as described in the main text.

C Evaluation Metrics

Dataset partitioning For the `Ribosome` dataset, the original paper provides predefined training and test splits. We use the provided test split as the test set and further partition the predefined training split into training and validation sets using an 85%:15% ratio. For the `OpenVaccine` and `Zebrafish` datasets, which do not come with predefined splits, we randomly partition the data into train, validation, and test sets using a 65%:15%:20% ratio.

Property Optimization To quantify the effectiveness of property optimization of different models, we measure both the median property gain and the fraction of sequences that are successfully optimized.

Specifically, given a test set of mRNA sequences X_{test} with predicted properties $\mathcal{P}_{\text{oracle}}(X_{\text{test}})$ and their optimized counterparts $\tilde{X}_{\text{test}}$ with properties $\mathcal{P}_{\text{oracle}}(\tilde{X}_{\text{test}})$, we compute:

$$\Delta_{\text{median}} = \text{median}\left(\mathcal{P}_{\text{oracle}}(\tilde{x}) - \mathcal{P}_{\text{oracle}}(x)\right), \quad x \in X_{\text{test}} \tag{4}$$

and

$$\%_{\text{success}} = \frac{1}{|X_{\text{test}}|} \sum_{x \in X_{\text{test}}} \mathbb{1}\left[\mathcal{P}_{\text{oracle}}(\tilde{x}) > \mathcal{P}_{\text{oracle}}(x)\right] \times 100\% \tag{5}$$

Here, Δ_{median} measures the gain in the oracle-predicted target property across the test set, while $\%_{\text{success}}$ reports the percentage of sequences that improve after optimization.

For models that cannot refine existing sequences (e.g., pure *de novo* generators), we assign a random pairing between initial and final sequences to enable a fair comparison.

Table S2: The UTR-LM oracle for property evaluation.

	Property Prediction	
	Pearson Corr. ↑	Spearman Corr. ↑
`OpenVaccine (n≈2k)`	0.62	0.73
`Zebrafish (n≈55k)`	0.82	0.81
`Ribosome (n≈260k)`	0.93	0.91

The external oracle In our evaluation for property optimization, we use an external oracle $\mathcal{P}_{\text{oracle}}(x)$ to predict the property of sequences generated by `RNAGenScape` as well as competing methods. Specifically, we adopt UTR-LM [52], a widely used mRNA language model, as the property prediction oracle. It is first pre-trained on more than 480,000 5' UTR mRNA sequences from diverse sources, and then fine-tuned for property prediction on each of our datasets. After fine-tuning, it is frozen and

never accessed during optimization, ensuring unbiased evaluation. Performance of the fine-tuned UTR-LM is shown in Table S2.

For the third-party checkpoints used to produce the results in Table 2, UTR-LM-TE is available at https://huggingface.co/multimolecule/utrlm-te_el and UTR-LM-MRL is available at https://huggingface.co/multimolecule/utrlm-mrl.

D Property Gradient Analysis

A central motivation of `RNAGenScape` is that property optimization along the data manifold should provide more reliable guidance than in the unconstrained ambient latent space. To examine predictor behavior in relation to training-data support, we compare gradient norms of the OAE property head and prediction variance among nearest training-set neighbors across three regimes in the OAE latent space:

- **Densely sampled on-manifold regions:** training-set latent representations whose local density falls within the top 30%, estimated using adaptive kNN graph degree.
- **Sparsely sampled on-manifold regions:** training-set latent representations whose local density falls within the bottom 30%.
- **Off-manifold regions:** we perturb training-set latent vectors with isotropic Gaussian noise of scale $5\times$ the median on-manifold nearest-neighbor distance, then retain only those candidates whose distance to the nearest training latent exceeds $2.5\times$ the 95th percentile of on-manifold nearest-neighbor distances (falling back to the farthest candidates if fewer than N points pass). This procedure constructs perturbed points far from the training embeddings.

Concretely, we encode the training set, subsample a reference dataset of up to $10{,}000$ latent vectors, and draw $N{=}500$ points per regime. For each point we report local density, distance to the nearest training latent, the raw ℓ_2 norm $\|\nabla_z f(z)\|_2$ of the property head, and the variance of f over the 10 nearest neighbors in the reference dataset.

Table S3 summarizes predictor behavior across the three regions. For each seed, we compute the mean and standard deviation of each metric across the 500 points in each region, then average these means and standard deviations separately over five seeds. Thus, the reported $\pm$ values primarily describe within-region variability rather than variability across seeds.

Off-manifold points have lower mean gradient norms and greater variation in predictions among their nearest training-set neighbors. These observations characterize predictor behavior and neighborhood composition away from training-data support.

Table S3: Property-gradient analysis in the OAE latent space on the Ribosome dataset. Reported values are the within-region mean and standard deviation averaged over five random seeds. Gradient norms are the raw ℓ_2 norms $\|\nabla_z f(z)\|_2$ of the OAE property head.

region	N	density	nearest-neighbor distance	grad norm	prediction variance
on-manifold, dense	500	39.1 ± 8.6	6.273 ± 0.428	0.398 ± 0.146	0.127 ± 0.103
on-manifold, sparse	500	9.1 ± 3.2	8.254 ± 0.669	0.345 ± 0.119	0.182 ± 0.142
off-manifold	500	0.0 ± 0.0	399.3 ± 25.7	0.192 ± 0.130	0.583 ± 0.430

E Additional Ablation Studies

Table S4: Ablation study on the design of the manifold projector Ψ. A simple kNN retractor is competitive in some settings but does not consistently support optimization in both directions. With a denoising autoencoder (DAE) as Ψ, multi-step denoising outperforms single-step denoising on `Zebrafish` and `Ribosome-loading`, and stays reasonably close on `OpenVaccine`.

	`OpenVaccine` (n≈2k)				`Zebrafish` (n≈55k)				`Ribosome-loading` (n≈260k)			
	+property		−property		+property		−property		+property		−property	
Ψ design	$\Delta\uparrow$	$\%\uparrow$	$\Delta\downarrow$	$\%\uparrow$	$\Delta\uparrow$	$\%\uparrow$	$\Delta\downarrow$	$\%\uparrow$	$\Delta\uparrow$	$\%\uparrow$	$\Delta\downarrow$	$\%\uparrow$
kNN (k=1)	+0.01	53.2	+0.00	50.3	−0.22	45.8	−0.42	61.5	+0.05	55.5	−0.09	56.7
kNN (k=5)	+0.06	56.2	+0.05	45.4	−0.22	46.0	−0.37	59.7	−1.05	9.7	−1.17	92.4
kNN (k=10)	+0.10	58.6	+0.07	43.1	−0.22	45.7	−0.29	58.2	−1.09	8.9	−1.21	92.5
DAE, single-step	**+0.93**	**88.3**	**−0.79**	**87.4**	+0.24	58.1	−0.83	72.1	+0.75	84.4	−0.99	74.4
DAE, multi-step	+0.92	83.6	−0.77	84.2	**+0.79**	**70.9**	**−1.25**	**77.3**	**+0.93**	**88.8**	**−1.93**	**92.8**

Table S5: Ablation study on the reconstruction weight λ_{Recon}. Although $\lambda_{\text{Recon}} = 1.0$ outperforms $\lambda_{\text{Recon}} = 5.0$ on `OpenVaccine` in both optimization directions, we decide to use $\lambda_{\text{Recon}} = 5.0$ across all datasets to maintain a consistent hyperparameter setting.

	`OpenVaccine` (n≈2k)				`Zebrafish` (n≈55k)				`Ribosome-loading` (n≈260k)			
	+property		−property		+property		−property		+property		−property	
λ_{Recon}	$\Delta\uparrow$	$\%\uparrow$	$\Delta\downarrow$	$\%\uparrow$	$\Delta\uparrow$	$\%\uparrow$	$\Delta\downarrow$	$\%\uparrow$	$\Delta\uparrow$	$\%\uparrow$	$\Delta\downarrow$	$\%\uparrow$
0.0	−1.29	16.2	−2.40	91.0	−0.79	33.9	−0.34	54.9	+0.05	54.0	−0.21	57.3
0.1	+1.00	66.8	−1.60	88.5	−0.11	51.3	−1.08	76.1	+0.39	77.5	−1.20	90.6
0.5	+0.93	87.2	**−2.92**	**99.4**	+0.51	63.7	−1.07	76.7	+0.85	88.7	−0.98	69.5
1.0	**+1.11**	**91.4**	−1.31	87.8	**+1.13**	**81.2**	−1.09	77.2	+0.71	85.4	−0.88	69.6
5.0	+0.92	83.6	−0.77	84.2	+0.79	70.9	**−1.25**	**77.3**	**+0.93**	**88.8**	**−1.93**	**92.8**
10.0	−0.49	49.2	−0.57	70.7	+0.58	66.4	−0.98	76.4	+0.68	82.0	−0.06	40.3

F Additional comparison against de novo baselines

In addition to state-of-the-art biological sequence optimization methods, we compare `RNAGenScape` with two different types of methods. The first type is *de novo* generative modeling approaches, namely variational autoencoder (VAE) [63], denoising diffusion probabilistic model (DDPM) [70], latent diffusion model (LDM) [88], and flow matching (FM) [89]. The second type comprises classic optimization algorithms that operate in our OAE latent manifold, including gradient ascent [73, 74], Markov chain Monte Carlo (MCMC) [75, 76], and hill climbing [77]. All classic optimization baselines are our GPU-compatible re-implementations of Castro et al. [80]. Hill climbing moves toward the best-scoring neighboring training embedding; its stochastic variant samples an improving neighbor uniformly, or any neighbor if none improves the score.

Table S6: `RNAGenScape` achieves better property optimization than *de novo* generative methods and classic optimization algorithms that operate on our learned data manifold.

Method	Property Optimization			
	+Property		−Property	
	$\Delta\uparrow$	$\%\uparrow$	$\Delta\downarrow$	$\%\uparrow$
`OpenVaccine (n≈2k)`				
VAE [63]	+0.23	56.5	+0.23	43.5
DDPM [70]	−0.33	35.1	−0.33	64.9
LDM [88]	−0.41	32.4	−0.41	67.6
FM [89]	−0.28	37.9	−0.28	62.1
OAE + gradient ascent	+0.00	38.9	+0.00	40.0
OAE + MCMC	+0.02	52.9	+0.00	48.9
OAE + hill climbing	+0.05	57.5	−0.10	65.6
OAE + stochastic hill climbing	+0.02	52.4	−0.03	55.3
`RNAGenScape` **(ours)**	**+0.92**	**83.6**	**−0.77**	**84.2**
`Zebrafish (n≈55k)`				
VAE [63]	−0.17	46.0	−0.17	54.0
DDPM [70]	+0.27	57.5	+0.27	42.5
LDM [88]	−0.67	36.0	−0.67	64.0
FM [89]	+0.22	56.7	+0.22	43.3
OAE + gradient ascent	−0.01	48.6	−0.02	52.5
OAE + MCMC	−0.18	43.9	−0.25	59.3
OAE + hill climbing	+0.17	56.9	−0.16	56.4
OAE + stochastic hill climbing	+0.09	54.5	−0.06	53.5
`RNAGenScape` **(ours)**	**+0.79**	**70.9**	**−1.25**	**77.3**
`Ribosome (n≈260k)`				
VAE [63]	−0.28	43.0	−0.28	57.0
DDPM [70]	−0.10	45.8	−0.10	54.2
LDM [88]	−0.18	44.6	−0.18	55.4
FM [89]	−0.11	45.6	−0.11	54.4
OAE + gradient ascent	+0.10	57.0	−0.04	52.1
OAE + MCMC	+0.01	50.6	−0.01	50.3
OAE + hill climbing	+0.07	53.9	−0.12	60.2
OAE + stochastic hill climbing	+0.05	53.7	−0.09	55.6
`RNAGenScape` **(ours)**	**+0.93**	**88.8**	**−1.93**	**92.8**

G Interpolation between sequences

In addition to optimizing mRNA sequences, we can interpolate between two existing sequences by guiding the latent embedding of one sequence toward that of another. Specifically, given a source sequence x_{source} and a target sequence x_{target}, we first obtain their latent embeddings via the encoder: $z_{\text{source}} = \mathcal{E}(x_{\text{source}})$ and $z_{\text{target}} = \mathcal{E}(x_{\text{target}})$.

We then perform property-guided Manifold Langevin dynamics starting from $z = z_{\text{source}}$, with the drift term being the force term that drives the embedding toward z_{target}:

$$f_{\text{interp}}(z, z_{\text{target}}) = -\frac{z - z_{\text{target}}}{\|z - z_{\text{target}}\|_2} \tag{6}$$

In this case, we slightly modify the update rule of the Langevin dynamics (Eqn (3)) as follows:

$$\mathrm{d}z_t = \eta \left(\frac{1}{\tau} \nabla_z f(z_t) + \lambda f_{\text{interp}}(z_t, z_{\text{target}}) \right) + \sqrt{2\eta} \cdot \epsilon_t. \tag{7}$$

By setting the interpolation weight $\lambda > 0$ in Eqn (7), we add a directional bias that drives the latent trajectory toward the target point, while following the property gradients.

Results on interpolation Guided by a directional force toward a specified target, `RNAGenScape` generates smooth and coherent trajectories on the learned manifold while preserving biological plausibility and continuity (Figure S1). These trajectories connect arbitrary input-target sequence pairs in a structured manner, reflecting semantically meaningful transitions. The distances from each intermediate point to the source and target quantitatively demonstrate the monotonicity and smoothness of the interpolation (Figure S2).

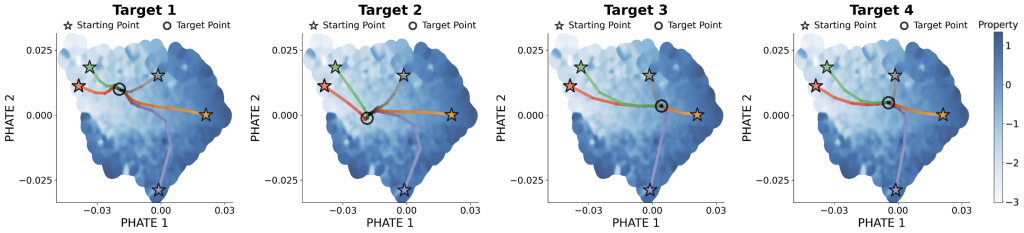


Figure S1: Latent space interpolation trajectories from 5 sources to 4 targets. Each trajectory is shown as a line fading from bright to dark in a consistent color. `RNAGenScape` produces smooth and coherent paths on the manifold between arbitrary input-target mRNA pairs.

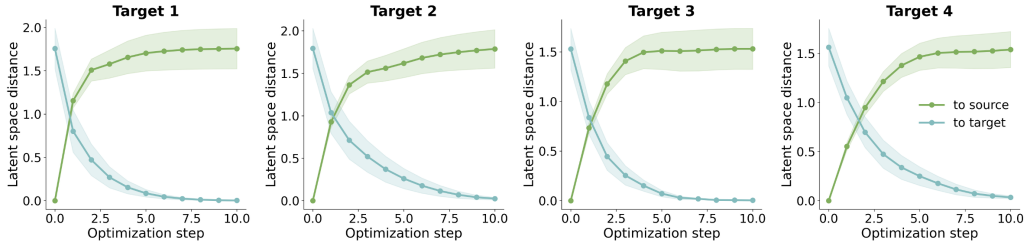


Figure S2: Latent space ℓ_2 distances during interpolation show smooth and monotonic transition from the source to the target. Results are averaged over all data samples.

H Additional optimization trajectories

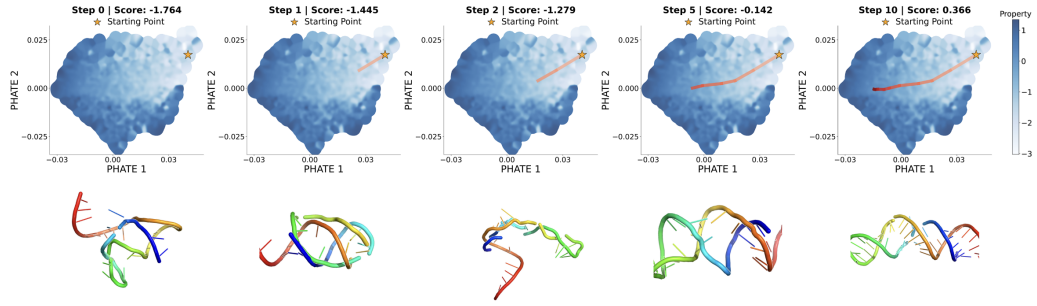


Figure S3: More examples of latent manifold trajectories. Trajectories in PHATE space and corresponding 3D structures are shown.

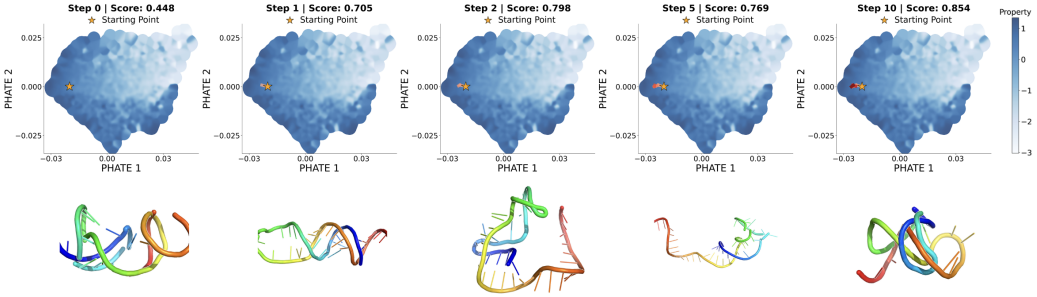


Figure S4: More examples of latent manifold trajectories. Trajectories in PHATE space and corresponding 3D structures are shown.

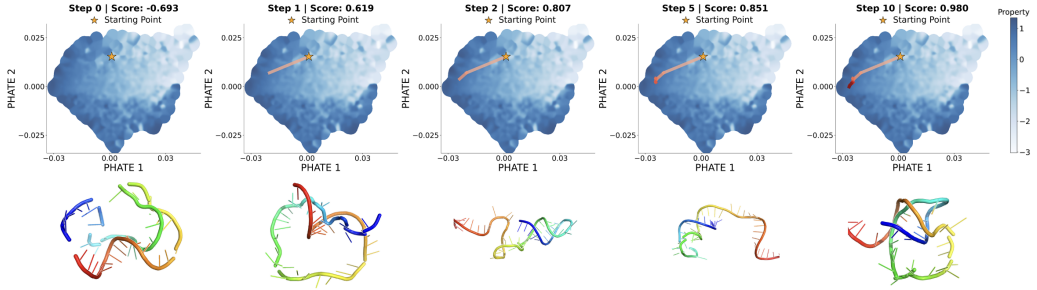


Figure S5: More examples of latent manifold trajectories. Trajectories in PHATE space and corresponding 3D structures are shown.

I Limitations and Future Work

One limitation of `RNAGenScape` is that it models and optimizes mRNA sequences only, without explicitly incorporating mRNA structure. While sequence-based representations capture many aspects of mRNA function, structures can provide auxiliary information in regulating translation, stability, and interactions.

In future work, we will study the possibility of performing sequence-structure joint modeling and optimization. Beyond mRNA, we plan to extend `RNAGenScape` to other modalities such as protein sequences and regulatory elements, and integrate active learning frameworks that guide wet lab experimentation. Such extensions would require a domain-appropriate encoder-decoder and training on domain-specific data, while retaining the overall formulation of property-organized latent learning, manifold projection, and property-guided Langevin optimization. By grounding sequence optimization on the manifold of real data, we aim to provide a versatile platform for interpretable high-throughput design in synthetic biology.

J Impact Statement

This work proposes a machine learning framework for optimizing mRNA sequences while preserving biological viability proxies by constraining optimization to a learned data manifold. Potential positive impacts include supporting computational design workflows in mRNA therapeutics and synthetic biology such as vaccine development by enabling more controlled and interpretable sequence optimization. As a general machine learning contribution, the framework may also inform broader research on manifold-constrained optimization for biological sequences.

The approach is intended as a computational tool to complement experimental and domain expertise rather than a substitute for empirical validation. As with other data-driven sequence models, limitations and biases in the available training data may influence optimization outcomes, and inappropriate use without biological oversight could lead to nonfunctional or misleading designs.

We do not anticipate direct negative societal consequences arising from this work beyond those already associated with machine learning-based biological modeling. We encourage future research to further study robustness, data bias, and safe integration of such methods into experimental pipelines.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: We have provided corresponding empirical results.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 have a limitations section.

Guidelines:

- The answer [N/A] 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: [N/A]

   Justification: No theoretical result.

   Guidelines:

   - The answer [N/A] 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: Provided details on reproducibility. Code is open-sourced.

   Guidelines:

   - The answer [N/A] 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: We have released the data and code.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified experimental details in the main paper and the Appendix.

   Guidelines:

   - The answer [N/A] 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: We have provided error bars in main experiments.

   Guidelines:

   - The answer [N/A] 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: We have provided information on compute resources.

   Guidelines:

   - The answer [N/A] 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: Yes, it conforms.

   Guidelines:

   - The answer [N/A] 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: We have an impact statement section.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: It seems not applicable to our case, as discussed in the impact statement.

    Guidelines:

- The answer [N/A] 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: Dataset owners were described.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: No new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

Justification: Not applicable.

Guidelines:

- The answer [N/A] 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: [N/A]

    Justification: Not applicable.

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is used only for paraphrasing and grammar checks.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.