RNA DESIGN FOR TARGET STRUCTURE BY LANGUAGE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

RNA design, the task of finding a sequence that folds into a target secondary structure, has broad biological and biomedical impact but remains computationally challenging due to the exponentially large sequence space and exponentially many competing folds. Traditional approaches treat it as an optimization problem, relying on per-instance heuristics or local search. We instead reframe RNA design as conditional sequence generation and introduce a reusable neural approximator, instantiated as an autoregressive language model (LM), that maps target structures directly to sequences. We first train our model in a supervised setting on random-induced structure-sequence pairs, and then use reinforcement learning (RL) to optimize end-to-end metrics. We also propose methods to select a small subset for RL that greatly improves RL efficiency and quality. Across four datasets, our approach outperforms state-of-the-art systems on key metrics such as Boltzmann probability while being $1.7\times$ faster, establishing conditional LM generation as a scalable, reusable alternative to per-instance optimization for RNA design. Our code and data are available at **https://anonymous.4open.science/r/Submission-31895**.

1 INTRODUCTION

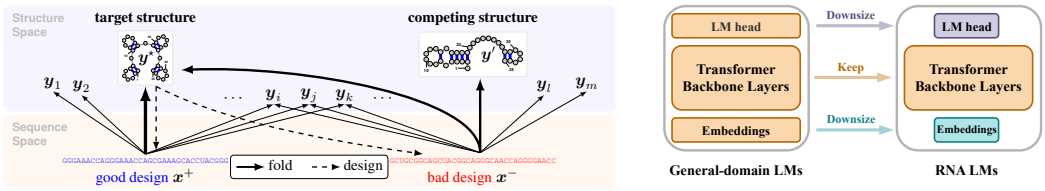
RNA plays a central role in cellular processes such as transcription, translation, catalysis, and gene regulation (Eddy, 2001; Doudna & Cech, 2002; Bachellerie et al., 2002). Its biological and biomedical importance is underscored by RNA viruses like SARS-CoV-2 and the Nobel Prizes in 2023 (mRNA vaccine) and 2024 (microRNAs). Since structure determines function, designing artificial RNA molecules with specific secondary structures enables many applications, including artificial ribozymes (Dotu et al., 2014; Yamagami et al., 2018), microRNAs (Schwab et al., 2006), aptamers (Hamada, 2018), and riboswitches (Bauer & Suess, 2006; Findeiß et al., 2017).

The problem of *RNA design*, also known as *RNA inverse folding*, aims to find an RNA sequence that folds into a given target structure (Hofacker et al., 1994; Andronescu et al., 2004; Garcia-Martin et al., 2013; Zadeh et al., 2011a; Zhou et al., 2023). Computationally, this is exceptionally challenging due to two levels of combinatorial explosion, in the *sequence (design) space* and the *structure (folding) space*: each target structure admits exponentially many candidate designs, and each design folds into exponentially many alternative structures that compete with the target. Indeed, the problem is NP-hard under simplified energy models (Bonnet et al., 2020).

Existing approaches to RNA design can be broadly categorized into three camps. The most common one is heuristic local search that iteratively mutates candidate sequences, such as RNAinverse (Hofacker et al., 1994), NEMO (Portela, 2018) and SAMFEO (Zhou et al., 2023). Operating on one or a few sequences at a time, they cannot keep up with the exponential growth of the design space and struggle with long and hard-to-design structures. To address this fundamental problem, the second camp recursively decomposes large structures into smaller ones: RNA-SSD (Andronescu et al., 2004), RNAiFold (Garcia-Martin et al., 2013), and NUPACK (Zadeh et al., 2011a); these are computationally expensive and generally underperform the first camp. A third group casts RNA design as a sequential decision-making problem for reinforcement learning (RL) (Eastman et al., 2018; Runge et al., 2019; Obonyo et al., 2022). However, these vanilla RL systems trained from scratch lag behind strong heuristic designers.

In this work, we propose a fundamentally different perspective: RNA design as *conditional sequence generation*. Instead of solving a combinatorial optimization problem from scratch for each structure, we learn a reusable (shared) *neural approximator solver*. We make the following contributions:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) (b)

Figure 1: **(a)** RNA design (inverse folding) is challenging due to *two levels of combinatorial explosion* in the design space and structure space. For a given target structure $\boldsymbol{y}^\star$, there are exponentially many candidate designs (e.g., $\boldsymbol{x}^+$, $\boldsymbol{x}^-$; dashed arrows), and each design folds into exponentially many structures ($\boldsymbol{y}_i$, $\boldsymbol{y}_j$, etc.; solid arrows) that compete with $\boldsymbol{y}^\star$. A good design ($\boldsymbol{x}^+$) folds predominantly into $\boldsymbol{y}^\star$, while a bad design ($\boldsymbol{x}^-$) folds into competing structures ($\boldsymbol{y}'$) with similar probabilities. **(b)** We convert a general-domain LLM into an RNA designer by keeping the pretrained transformer backbone and shrinking the input and output layers. The original embedding and LM head are downsized and reinitialized to support RNA tokens.

1. We formulate RNA design as conditional sequence generation by autoregressive language models.
2. We devise a constrained decoding algorithm to ensure all designs are valid for the input structure.
3. We develop a procedure combining pretraining, supervised learning (SL) on structure-sequence pairs, and reinforcement learning (RL) on selected structures to directly optimize folding success.
4. We show that, surprisingly, SL on random-induced structure-sequence pairs unrelated to the test data yield competitive results on the latter.
5. We develop methods to select a small subset of structures for efficient RL using sample diversity.
6. We utilize fast parallel sampling of large amounts of designs on GPUs, and our best model (SL+RL) outperforms state-of-the-art systems in key metrics on four test sets, and is 1.7× faster.

2 Preliminaries: RNA Folding and Design

An RNA sequence $\boldsymbol{x}$ of length n is a string of nucleotides $x_1 x_2 \dots x_n$, where $x_i \in \mathcal{N}$ and $\mathcal{N} \triangleq \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. A pseudoknot-free secondary structure $\boldsymbol{y}$ is represented as a dot-bracket string of the same length. Each character $y_i \in \{\mathtt{.}, \mathtt{(}, \mathtt{)}\}$ indicates whether nucleotide x_i is unpaired, paired downstream, or paired upstream, respectively. Base pairs are restricted to Watson-Crick-Franklin or wobble pairs: $\mathcal{P} = \{\mathtt{CG}, \mathtt{GC}, \mathtt{AU}, \mathtt{UA}, \mathtt{GU}, \mathtt{UG}\}$. For a given sequence $\boldsymbol{x}$, its *ensemble* $\mathcal{Y}(\boldsymbol{x})$ consists of all possible secondary structures. In thermodynamic RNA folding models, the *Gibbs free energy change* $\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})$ characterizes the stability of secondary structure $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$: the lower the free energy, the more stable $\boldsymbol{y}$ is for $\boldsymbol{x}$. The most stable structure in the ensemble is the one with *minimum free energy*, i.e., the *MFE structure*, $\mathrm{MFE}(\boldsymbol{x}) \triangleq \operatorname{argmin}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})$.

Note that most secondary structure prediction methods break argmin ties arbitrarily when multiple structures share the lowest free energy. To be precise, we define $\mathrm{MFEs}(\boldsymbol{x}) \triangleq \{\boldsymbol{y} \mid \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}')\}$ to be the *set of MFE structures* for $\boldsymbol{x}$. When $|\mathrm{MFEs}(\boldsymbol{x})| = 1$, we say $\boldsymbol{x}$ has a *unique MFE* (uMFE) structure.

The *partition function* $Q(\boldsymbol{x}) \triangleq \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} e^{-\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})/RT}$ sums the contribution of all structures in the ensemble, where R is the molar gas constant and T is the temperature. Accordingly, the Boltzmann probability of a sequence $\boldsymbol{x}$ folding into a structure $\boldsymbol{y}$ is defined as $p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{e^{-\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})/RT}}{Q(\boldsymbol{x})}$.

Given a target structure $\boldsymbol{y}^\star$, RNA design (Fig. 1(a)) aims to find a suitable RNA sequence $\boldsymbol{x}$ that can naturally and easily fold into $\boldsymbol{y}^\star$, within the design space $\mathcal{X}(\boldsymbol{y}^\star)$ of all valid sequences for $\boldsymbol{y}^\star$:

$$\mathcal{X}(\boldsymbol{y}^\star) \triangleq \{\boldsymbol{x} \in \mathcal{N}^{|\boldsymbol{y}^\star|} \mid \forall (i,j) \in \mathit{pairs}(\boldsymbol{y}^\star), x_i x_j \in \mathcal{P}\} \tag{1}$$

We formulate the RNA design problem as optimizing a score $f(\boldsymbol{x}, \boldsymbol{y}^\star)$ over the design space $\mathcal{X}(\boldsymbol{y}^\star)$:

$$\boldsymbol{x}^\star = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{y}^\star)} f(\boldsymbol{x}, \boldsymbol{y}^\star) \tag{2}$$

where the scoring function quantifies how "easily" $\boldsymbol{x}$ folds into $\boldsymbol{y}^\star$. In practice, design quality is evaluated with several standard metrics (Anderson-Lee et al., 2016; Zhou et al., 2023) that we adopt

below. The first two (ensemble-based) are more important than the last two (MFE-based), since even an MFE solution $\boldsymbol{x}$ can admit many structures of similar energy, leaving $\boldsymbol{y}^\star$ with low ensemble probability (see App. S1 and Fig. S1 for a detailed comparison).

- **Boltzmann probability** $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ is how likely $\boldsymbol{x}$ folds into $\boldsymbol{y}^\star$ among all possible structures.
- **Normalized Ensemble Defect** $\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$ quantifies how well the ensemble of structures formed by $\boldsymbol{x}$ aligns with the target, penalizing competing folds (Zadeh et al., 2011b): $\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star) = \frac{1}{|\boldsymbol{x}|} \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} p(\boldsymbol{y} \mid \boldsymbol{x})\, d_{\text{struct}}(\boldsymbol{y}^\star, \boldsymbol{y})$, where $d_{\text{struct}}(\boldsymbol{y}, \boldsymbol{y}') \triangleq |\boldsymbol{y}| - 2 \cdot |pairs(\boldsymbol{y}) \cap pairs(\boldsymbol{y}')| - |unpaired(\boldsymbol{y}) \cap unpaired(\boldsymbol{y}')|$ is the structural distance.
- **MFE (or uMFE) solution**: $f(\boldsymbol{x}, \boldsymbol{y}^\star) = 1$ iff. $\boldsymbol{y}^\star \in \mathrm{MFEs}(\boldsymbol{x})$ (or iff. $\mathrm{MFEs}(\boldsymbol{x}) = \{\boldsymbol{y}^\star\}$).

3 Models and Constrained Decoding

3.1 RNA Design as Conditional Sequence Generation

We frame RNA design as a conditional sequence generation task. Given a target secondary structure $\boldsymbol{y}^\star$, we train an autoregressive language model $p_\theta(\cdot \mid \boldsymbol{y}^\star)$ that defines the conditional distribution

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{y}^\star) = \prod_{t=1}^{n} p_\theta(x_t \mid \boldsymbol{x}_{<t}, \boldsymbol{y}^\star),$$

where $x_t \in \mathcal{N}$ is the nucleotide at position t and $\boldsymbol{x}_{<t}$ is the sequence prefix.

Vocabulary and input. We employ a domain-specific vocabulary for structure-conditioned RNA generation: the four nucleotides (`A`, `C`, `G`, `U`), the three dot-bracket structure symbols (`(`, `)`, `.`), and a few control tokens. The tokens `<struct>` and `</struct>` delimit the target structure, while `<bos>` and `<eos>` mark sequence boundaries. During inference, the structure is provided as a prefix prompt, after which the model emits nucleotides until `<eos>`:

<struct> (. (. . .)) </struct> <bos> G A C U U A G C <eos>

prompt

3.2 Constrained Decoding

By default, autoregressive generation does not respect the biochemical pairing rules required for a sequence to be compatible with the target structure, resulting in invalid sequences outside the valid design space $\mathcal{X}(\boldsymbol{y}^\star)$. For example, if $\boldsymbol{y}^\star =$ `((...))` and $x_1 =$ `A`, then the corresponding x_7 can only be `U`. So we need to employ a *constrained decoding* mechanism that enforces base-pairing rules during generation. In formal language terms, our valid design space $\mathcal{X}(\boldsymbol{y}^\star)$ is a **context-free language** needing a runtime stack for valid decoding; our method below instead uses a (stack-based) precomputation that avoids the runtime stack, reducing the overhead.

First, we precompute a mapping $m(t)$ that maps the index of each "`)`" to the index of its matching "`(`". Then at decoding step t, the set of admissible nucleotides is defined as:

$$\mathcal{A}_t = \begin{cases} \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}, & y_t^\star \in \{\mathtt{.}, \mathtt{(}\}, \\ \mathrm{Comp}(x_{m(t)}), & y_t^\star = \mathtt{)} \end{cases}$$

where $\mathrm{Comp}(\cdot)$ lists the nucleotides compatible with pairing constraints, i.e., $\mathrm{Comp}(\mathtt{A}) = \{\mathtt{U}\}$, $\mathrm{Comp}(\mathtt{C}) = \{\mathtt{G}\}$, $\mathrm{Comp}(\mathtt{G}) = \{\mathtt{C}, \mathtt{U}\}$, and $\mathrm{Comp}(\mathtt{U}) = \{\mathtt{A}, \mathtt{G}\}$.

At each decoding step, we mask the logits of all invalid nucleotides according to $\mathcal{A}_t$ and renormalize: $\tilde{p}_\theta(x_t \mid \boldsymbol{x}_{<t}, \boldsymbol{y}^\star) \propto \mathbf{1}[x_t \in \mathcal{A}_t]\, p_\theta(x_t \mid \boldsymbol{x}_{<t}, \boldsymbol{y}^\star)$. A lightweight callback computes $\mathcal{A}_t$ from $y_t^\star$ and the nucleotide previously generated at its paired site. This strategy guarantees structural validity by construction and reduces the effective search space from 4^n (regular language $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}^n$) to $6^m 4^u$ (context-free language $\mathcal{X}(\boldsymbol{y}^\star)$) for m base pairs and u unpaired positions. Its computational overhead is about 30% (Fig. 4(b)), but it substantially improves sample efficiency in RL.

4 Language Model Training

4.1 Pretrained Model

Large language models trained on massive, diverse text corpora acquire broad inductive biases that transfer surprisingly well beyond natural language (Touvron et al., 2023; Yang et al., 2024),

and even relatively small decoder-only LMs make downstream fine-tuning significantly more data-efficient than training a transformer from scratch. We therefore build on the pretrained Qwen2.5-0.5B (Yang et al., 2024) (GPT-style decoder-only Transformer) as the backbone of our RNA design system. Although its original training domain is textual, our experiments show that the pretrained backbone provides substantially better optimization stability and sample efficiency than a randomly initialized transformer of identical architecture.

Surgery for RNA adaptation. We turn the general-domain LLM into an RNA-capable generator with minimal architectural modification. As illustrated in Fig. 1(b), we keep all transformer backbone layers and modify only the input and output layers: the embedding matrix and LM head are replaced by downsized, RNA-specific versions for the vocabulary above. This preserves the pretrained backbone while letting the model process RNA tokens and structural prompts.

Why not use existing RNA LMs? Several RNA-focused language models have been proposed, but they are not well suited to our setting (see also Sec. 7):

1. **Lack of structural input.** Existing RNA LMs typically operate only on nucleotide sequences and do not accept dot-bracket structures as input, preventing structure-conditioned generation.
2. **Architectural mismatch.** Most publicly available RNA LMs are BERT-style encoders, not generative decoders. A GPT-style RNA LM was reported but with no usable checkpoint (Xiao et al., 2024).
3. **Tokenization mismatch.** Existing RNA LMs adopt subword or BPE-style tokenization (e.g., merging adjacent nucleotides into one token (Zhao et al., 2024)), breaking the one-nucleotide-per-token setting that our constrained decoding relies on, so base-pairing constraints can no longer be enforced easily.
4. **Software compatibility.** Many RNA LMs are implemented in standalone codebases or lack Hugging-Face compatibility, making integration with our decoding and RL pipeline difficult.

Why not encoder-decoder models? RNA design can be formulated as encoder-decoder translation ($\boldsymbol{y}^\star \rightarrow \boldsymbol{x}$), but we adopt a decoder-only LM for high-throughput sampling and pretraining compatibility. Encoder-decoder models compute cross-attention at every decoding step (Vaswani et al., 2017), which often dominates incremental decoding cost and limits sampling throughput when generating many candidates per target (Lu et al., 2024) (we use 10^4 samples). Prefix-prompt conditioning also keeps supervised training and future RNA pretraining straightforward and naturally supports mixing RNA sequences with sequence-structure pairs, whereas encoder-decoder pretraining on sequence-only data requires denoising objectives or back-translation (Raffel et al., 2020; Lewis et al., 2020; Sennrich et al., 2016).

4.2 Supervised Learning (SL)

To train the model by supervision, we construct a corpus of structure-sequence pairs using a strong optimization-based solver, SAMFEO (Zhou et al., 2023), on diverse target structures (see Sec. 5.2). The resulting dataset $\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{X}}_{\text{train}}$ contains $\{(\boldsymbol{y}^\star, \boldsymbol{x})\}$ pairs where each $\boldsymbol{x}$ is a valid design for $\boldsymbol{y}^\star$, and the model is trained by maximum likelihood estimation (MLE):

$$\mathcal{L}_{\text{SL}}(\theta) = - \sum_{(\boldsymbol{y}^\star, \boldsymbol{x}) \in \boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{X}}_{\text{train}}} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{y}^\star).$$

After SL, the model can rapidly generate many valid designs, enabling efficient exploration in RL.

4.3 Reinforcement Learning (RL)

Supervised learning aligns the model with solver-generated sequences but does not directly optimize our thermodynamic objective. We therefore fine-tune p_θ with a simple group-based REINFORCE-style objective (Williams, 1992), often referred to as *Group Relative Policy Optimization* (GRPO) (Shao et al., 2024).

Given a target structure $\boldsymbol{y}^\star$, we sample K sequences $\{\boldsymbol{x}_k\}_{k=1}^K \sim p_\theta(\cdot \mid \boldsymbol{y}^\star)$ (by constrained decoding) and compute rewards $R_k = R(\boldsymbol{x}_k, \boldsymbol{y}^\star)$. Since the Boltzmann probability is the most important metric in RNA design (Ward et al., 2023) (App. S1), we use $R(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y} \mid \boldsymbol{x})$. We form a group baseline $b = \frac{1}{K}\sum_{k=1}^K R_k$ with standard deviation $\sigma = \text{Std}_k(R_k)$, and define a normalized advantage $\hat{A}_k = \frac{R_k - b}{\sigma + \varepsilon}$. The GRPO loss $\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_k\left[\hat{A}_k \log p_\theta(\boldsymbol{x}_k \mid \boldsymbol{y}^\star)\right]$ increases probabilities of better-than-average samples within each group and decreases those worse-than-average ones, on top of SL model.

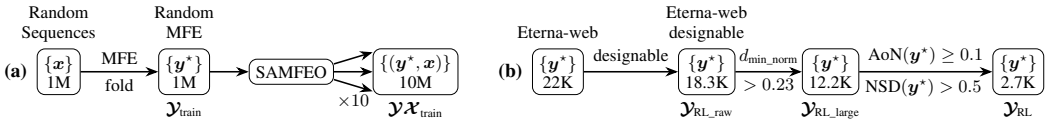


Figure 2: Construction of (a) SL and (b) RL training datasets; $d_{\text{min_norm}}$ denotes $d_{\text{min_norm}}(\boldsymbol{y}^\star, \mathcal{Y}_{\text{test}})$.

5 Datasets Construction

5.1 Test Dataset $\mathcal{Y}_{\text{TEST}}$

For RNA design, the test set $\mathcal{Y}_{\text{test}}$ is a collection of target structures $\boldsymbol{y}^\star$. We create a comprehensive test set consisting of four datasets, covering both manually-designed puzzles and native structures:

1. **Eterna100** (Anderson-Lee et al., 2016) is the most widely used benchmark for evaluating RNA design programs. It contains 100 secondary structures (i.e., human-designed "puzzles") of up to 400 *nt*, varying in design difficulty from simple hairpins to intricate multiloop structures.
2. **Eterna100-v2** (Koodli et al., 2021) modifies 19 puzzles from Eterna100 so that all puzzles are MFE-solvable under Turner 2004 energy model (Mittal et al., 2024) via ViennaRNA v2.
3. **Rfam-Taneda-27** (Taneda, 2011) includes consensus secondary structures from seed alignments in Rfam 9.0, with each entry corresponding to an RNA family.
4. **RNAsolo100**, extracted from Adamczyk et al. (2022), is a collection of secondary structures curated from a database of experimentally determined RNA 3D structures. The "100" refers to 100 cleaned, non-redundant structures we selected for design benchmarking.

We call the set of unique structures from all four datasets our **"Union Test Set"**, notated $\mathcal{Y}_{\text{test}}$, which contains 246 unique structures, with length up to 400 *nt*.

5.2 Supervised Learning Dataset $\mathcal{YX}_{\text{TRAIN}}$

Our supervised training dataset $\mathcal{YX}_{\text{train}}$ is a set of $(\boldsymbol{y}^\star, \boldsymbol{x})$ pairs where each $\boldsymbol{y}^\star$ is a target structure (i.e., a puzzle) and each $\boldsymbol{x}$ is the corresponding RNA sequence (i.e., a design). We construct this set in two steps: (a) collect the set of target structures $\mathcal{Y}_{\text{train}}$, and then (b) use a state-of-the-art tool to generate candidate designs for each structure in $\mathcal{Y}_{\text{train}}$.

For step (a), while previous work used native or native-derived structures such as Rfam-Learn (Runge et al., 2019), those data suffers from three serious drawbacks:

1. **Testset contamination**: the test set contains native structures that are too similar to some native or native-derived ones in the training set;
2. **Limited size**: native structures are often limited in quantity (e.g., Rfam-Learn has only 65K structures), which may not provide enough data for the data-hungry Transformer models.
3. **Limited diversity**: many native sequences are homologous ones from the same families (e.g., Rfam-Learn is derived from Rfam, a collection of native RNA sequences by families), where sequences and structures in each family share high sequence identities and structural similarities.

To address these issues, we instead propose to use large amounts of randomly generated (and thus diverse) artificial structures unrelated to the test sets. However, directly generating random secondary structures is problematic, as most of them will be undesignable under the MFE criterion (Zhou et al., 2024). Instead, we first generated 1M random RNA *sequences* of lengths up to 500 *nt*, with uniform distribution of nucleotides $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$, and then used ViennaRNA v2 (Lorenz et al., 2011) to get their MFE structures, yielding 1M random-MFE structures in $\mathcal{Y}_{\text{train}}$ (Fig. 2(a)), all MFE-designable by construction (though some may not be uMFE-designable).

For each target structure in $\mathcal{Y}_{\text{train}}$, we use the state-of-the-art SAMFEO tool to generate 10 candidate designs, so finally we have 10M $(\boldsymbol{y}^\star, \boldsymbol{x})$ pairs in $\mathcal{YX}_{\text{train}}$. Surprisingly, this random-induced supervised dataset unrelated to the test set yields competitive results on the latter (Sec. 6.1).

5.3 Reinforcement Learning Dataset $\mathcal{Y}_{\text{RL}}$

It is a general consensus in language model training that the later the training stage, the less data is needed (with more compute per example), but with higher quality standards. Given that RL is the

last stage in training, we construct a small and relevant dataset $\mathcal{Y}_{\text{RL}}$ in three steps. It is important to note that, different from supervised learning, RL only needs structures $\boldsymbol{y}^\star$, but not solutions $\boldsymbol{x}$, since it will explore possible solutions via decoding.

Step 1. data source: Eterna-web. We first downloaded all 22K player-designed structures from the Eterna website (http://eternagame.org), excluding those from Eterna100. We then removed structures longer than 500 *nt* (to match test sets), and removed MFE-undesignable ones using the method of Zhou et al. (2024), which resulted in 18.3K MFE-designable structures ($\mathcal{Y}_{\text{RL_raw}}$ in Fig. 2(b)).

Step 2. Removing structures similar to test data. Unlike our randomly generated SL training set that are unrelated to our test data, our RL set is drawn from a distribution similar to one of our testsets (Eterna100), so potentially there is a risk of testset contamination. To ensure a rigorous evaluation, we removed structures $\boldsymbol{y}^\star \in \mathcal{Y}_{\text{RL_raw}}$ that were too close to any test structure $\boldsymbol{y}'$ in $\mathcal{Y}_{\text{test}}$. For each $\boldsymbol{y}^\star$, we computed a *normalized minimum edit distance* that accounts for length differences:

$$d_{\text{min_norm}}(\boldsymbol{y}^\star, \mathcal{Y}_{\text{test}}) = \min_{\boldsymbol{y}' \in \mathcal{Y}_{\text{test}}} \frac{d_{\text{edit}}(\boldsymbol{y}^\star, \boldsymbol{y}')}{\min(|\boldsymbol{y}^\star|, |\boldsymbol{y}'|)}.$$

Here, $d_{\text{edit}}(\cdot,\cdot)$ denotes the edit distance between two structures, and normalization by the length of the shorter structure $\min(|\boldsymbol{y}^\star|, |\boldsymbol{y}'|)$ ensures that our similarity measure is comparable across structures of different lengths. To calibrate the cutoff, we use RNAStrAlign (Tan et al., 2017), $\sim$40K aligned native sequence-structure pairs from eight RNA families. For each structure, we compute the minimum $d_{\text{min_norm}}$ to any structure from a different family; aggregating these cross-family minima characterizes how similar structurally unrelated native structures can appear under our metric (Fig. 3(a)). The empirical minimum is approximately 0.23, which we adopt as the threshold $\tau = 0.23$. Structures in $\mathcal{Y}_{\text{RL_raw}}$ with $d_{\text{min_norm}}(\boldsymbol{y}^\star, \mathcal{Y}_{\text{test}}) \leq \tau$ are excluded to prevent information leakage between RL and test sets (Fig. 3(b)), resulting in 12.2K structures ($\mathcal{Y}_{\text{RL_large}}$ in Fig. 2(b)).

Step 3. Further filtering for RL efficiency and quality. We could do RL on the above $\mathcal{Y}_{\text{RL_large}}$ set, but it is computationally expensive because, for each target structure the model must generate many samples ("exploration") and evaluate their folding performance. Our goal is to identify a small subset $\mathcal{Y}_{\text{RL}} \subseteq \mathcal{Y}_{\text{RL_large}}$ for which RL is likely to yield meaningful improvement.

To do so, we first decode structures in $\mathcal{Y}_{\text{RL_large}}$ with the SL model trained on $\mathcal{YX}_{\text{train}}$ by generating K candidate designs $\{\boldsymbol{x}_k\}_{k=1}^K$ and compute their Boltzmann probabilities $p(\boldsymbol{y}^\star \mid \boldsymbol{x}_k)$. We then summarize the distribution of these probabilities with two statistics:

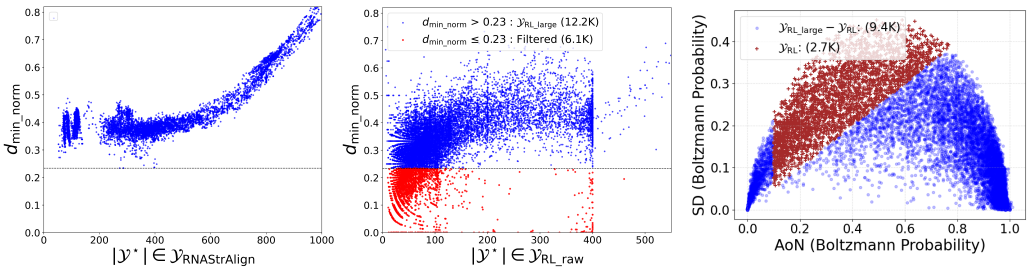
- **Average probability (AoN):** $\text{AoN}(\boldsymbol{y}^\star) = \frac{1}{K}\sum_{k=1}^K p(\boldsymbol{y}^\star \mid \boldsymbol{x}_k)$, which reflects the performance of the SL model for structure in terms of the ensemble metric $p(\boldsymbol{y}^\star \mid \boldsymbol{x}_k)$.
- **Standard deviation of probabilities:** $\text{SD}(\boldsymbol{y}^\star) = \text{Std}_k(p(\boldsymbol{y}^\star \mid \boldsymbol{x}_k))$ capturing the variability, and hence the potential for improvement across sampled designs.

Using $\text{SD}(\boldsymbol{y}^\star)$ alone can be misleading for structures whose probabilities remain uniformly low. Thus, we propose the *normalized standard deviation* (NSD): $\text{NSD}(\boldsymbol{y}^\star) = \frac{\text{SD}(\boldsymbol{y}^\star)}{\text{AoN}(\boldsymbol{y}^\star)}$, which expresses variability relative to the mean probability and highlights structures where RL can exploit useful diversity in folding outcomes.

Selection criteria. We select a structure $\boldsymbol{y}^\star$ from $\mathcal{Y}_{\text{RL_large}}$ for reinforcement learning if

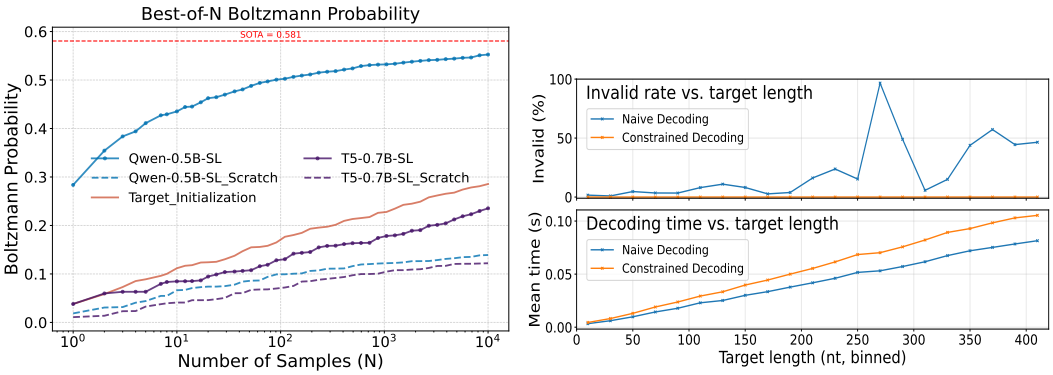
$$\text{AoN}(\boldsymbol{y}^\star) \geq 0.1 \quad \text{and} \quad \text{NSD}(\boldsymbol{y}^\star) > 0.5. \tag{3}$$

The AoN threshold removes structures on which the SL model performs extremely poorly, while the NSD threshold ensures that there is sufficient variance among sampled sequences for RL to meaningfully refine the model. Applying both criteria yields a filtered subset $\mathcal{Y}_{\text{RL}}$ of approximately 2.7K structures (Fig. 3(c)) on which RL is both computationally feasible and empirically effective. Moreover, Fig. S6 compares AoN scores before and after RL when training on either the full set $\mathcal{Y}_{\text{RL_large}}$ or the filtered subset $\mathcal{Y}_{\text{RL}}$. When RL is applied to the entire $\mathcal{Y}_{\text{RL_large}}$, many structures in $\mathcal{Y}_{\text{RL_large}}$ actually exhibit degraded performance after RL. In contrast, when RL is restricted to $\mathcal{Y}_{\text{RL}}$, the model not only improves performance on most structures in $\mathcal{Y}_{\text{RL}}$ but also generalizes positively, yielding improvements for many structures in $\mathcal{Y}_{\text{RL_large}}$ as well. These results confirm that our filtering yields a small subset that improves both the efficiency and quality of RL training. Sec. 6.4 further showed a $\sim 2.9\times$ speedup by RL on $\mathcal{Y}_{\text{RL}}$ (16 hours vs. 47 hours of RL on $\mathcal{Y}_{\text{RL_large}}$).



(a) Identifying $d_{\text{min_norm}}$ using RNAStrAlign (b) distance filtering on $\mathcal{Y}_{\text{RL_raw}}$ (c) decoding $\mathcal{Y}_{\text{RL_large}}$ with SL model

Figure 3: (a) Calibrating $d_{\text{min_norm}}$ with RNAStrAlign (Tan et al., 2017). (b) Removing structures in $\mathcal{Y}_{\text{RL_raw}}$ that are too close to the Eterna100 test set. (c) Selecting the small subset $\mathcal{Y}_{\text{RL}}$ from $\mathcal{Y}_{\text{RL_large}}$.



(a) (b)

Figure 4: **(a)** Supervised learning results of different models trained on $\mathcal{YX}_{\text{train}}$ in terms of best-of-N Boltzmann probability on Eterna100. **(b)** Naive vs. constrained decoding on $\mathcal{Y}_{\text{test}}$ (10^3 samples per structure; SL model). Naive decoding becomes increasingly invalid for longer targets, while constrained decoding guarantees validity at a $\sim 30\%$ slowdown.

6 Evaluation Results

6.1 Supervised Learning Results

Figure 4(a) reports best-of-N Boltzmann probability for each supervised-learning (SL) model trained on our 10M $\mathcal{YX}_{\text{train}}$ set, as a function of the number of decoded samples N per target structure. The pretrained Qwen-0.5B-SL model is strongest at every sampling budget, reaching ≈ 0.55 at $N = 10^4$, while both models trained from scratch, Qwen-0.5B-SL_Scratch (decoder-only) and T5-0.7B-SL_Scratch (encoder-decoder), are substantially worse. Training Transformer LMs for RNA design entirely from scratch is thus ineffective regardless of architecture.

`Target_Initialization` is a simple baseline where for each unpaired position in $\boldsymbol{y}^\star$, we use a distribution of 70% `A` (10% each for `C`, `G`, and `U`) and for each basepair in $\boldsymbol{y}^\star$, we use a distribution of 73% `GC`/`CG` pairs, 18% `AU`/`UA` pairs, and 9% wobble pairs. This baseline performs surprisingly well, even surpassing both T5 models at moderate sampling budgets; although far below the best model, its competitiveness reveals the limitations of uninitialized models in capturing structural constraints. It is also encouraging that supervised training on random-induced structure-sequence pairs completely unrelated to the test set performs reasonably well on the latter.

6.2 Constrained Decoding

We compare naive and constrained decoding on the union test set $\mathcal{Y}_{\text{test}}$ using the Qwen2.5-0.5B-SL model, drawing 10^3 samples per target with both methods and recording, for every decoded sequence, (i) structural invalidity (any base pair in $\boldsymbol{y}^\star$ violating Watson-Crick-Franklin or wobble rules) and (ii) wall-clock decoding time. As shown in Fig. 4(b), naive decoding yields a sharp rise in invalid sequences as target length increases, wasting a substantial fraction of samples, whereas con-

Method	Eterna100				RNAsolo100				Eterna100-v2				Rfam-Taneda-27			
	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE	uMFE	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE	uMFE	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE	uMFE	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE	uMFE
RNAinverse-pf	0.571	0.045	77	72	0.710	0.015	79	79	0.590	0.039	82	77	0.808	**0.004**	24	24
NEMO	0.271	0.098	**79**	**77**	0.506	0.038	**81**	**81**	0.270	0.098	**90**	**87**	0.411	0.025	**25**	**25**
SAMFEO	0.581	0.043	77	74	0.723	**0.013**	80	79	0.596	**0.036**	83	79	0.809	**0.004**	24	24
Ours: SL+RL	**0.588**	**0.042**	75	72	**0.728**	**0.013**	**81**	79	**0.607**	**0.036**	82	79	**0.814**	**0.004**	24	24

Table 1: Comparison with state-of-the-art systems on all four test sets. RNAinverse-pf was updated in ViennaRNA 2.6 (2023). Our results report the best of 10^4 samples.

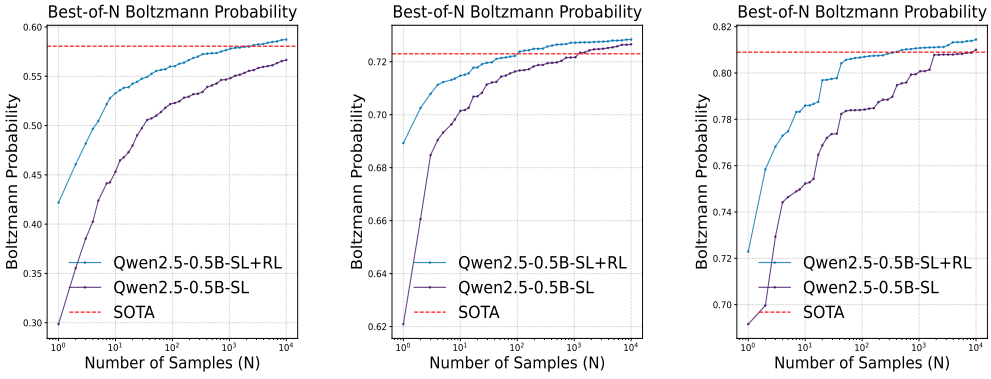
strained decoding eliminates invalid outputs by design at a modest $\sim 30\%$ overhead. The resulting gain in effective sample efficiency leads us to adopt constrained decoding in RL. Fig. S5 shows an attention map from constrained decoding, which correlates with base pairs in $\boldsymbol{y}^\star$.

6.3 Reinforcement Learning Results

Fig. 5 shows that reinforcement learning (RL) on top of the supervised-learning (SL) model substantially improves best-of-N Boltzmann probability on all three benchmarks, with particularly large gains on Eterna100 and Rfam-Taneda-27 at every sampling budget, strengthening the model on complex and human-designed structures. In Table 1, our reinforcement-fine-tuned model matches or surpasses other methods on ensemble-based metrics, achieving the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ across all datasets and the best or tied-best NED, while beating NEMO on both metrics everywhere. It is likewise sample-efficient for MFE solutions: Fig. S2 shows it needs only 20 samples on Rfam-Taneda-27 and 50 on RNAsolo100 to match the state-of-the-art method.

Fig. S7 compares Boltzmann probabilities of SAMFEO and our SL+RL model on Eterna100, one "+" per puzzle with marker size reflecting structure length. Our approach generally achieves higher probabilities, with the largest advantage on the longest, hardest-to-design puzzles. Since arithmetic means (Tab. 1) are dominated by puzzles where both methods reach $p(\boldsymbol{y}^\star \mid \boldsymbol{x}) \approx 1$, we also compare geometric means over the 81 MFE-solvable puzzles: ours is $1.77\times$ better than SAMFEO overall, and $10.7\times$ better on puzzles longer than 200 nt (App. S2).

Since RL optimizes $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ under ViennaRNA, its gains could be model-specific artifacts; two checks argue otherwise. First, under two predictors never used in training, LinearFold-C (machine-learned) and MXfold2 (hybrid thermodynamic/machine-learned), neither purely thermodynamic like ViennaRNA, our model outperforms SAMFEO in all 12 comparisons (both predictors, three benchmarks, NED and $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$), so its gains transfer across model types (App. S3). Second, RL does not produce degenerate sequences: on Eterna100, GC content and homopolymer runs barely change after RL, and 84% of samples per target remain distinct (App. S4).



(a) Eterna100 (b) RNAsolo100 (c) Rfam-Taneda-27

Figure 5: Best-of-N probability curves for the SL and SL+RL models on three test sets. In all cases, RL brought substantial improvements over SL, surpassing SAMFEO. See Fig. S3 for other metrics.

6.4 Efficiency

On a single NVIDIA H100 GPU, our Qwen2.5-0.5B model requires 50 hours of supervised learning (SL) on the 10M structure-sequence pairs of $\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{X}}_{\text{train}}$, and 16 and 47 hours of reinforcement learning (RL) on $\boldsymbol{\mathcal{Y}}_{\text{RL}}$ and $\boldsymbol{\mathcal{Y}}_{\text{RL_large}}$ (2.7K and 12.2K structures), respectively.

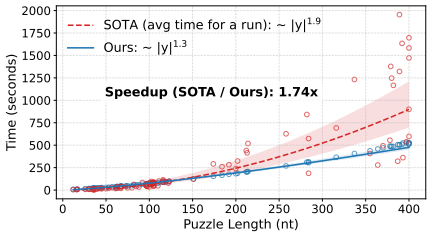


Figure 6: Decoding time: LM (10^4 samples) on GPU is $1.7\times$ faster than SAMFEO on 5 CPUs (each run explores $\sim 8\times 10^3$ designs).

While training is slow, decoding scales well to our best-of-10^4 budget. Fig. 6 reports Eterna100 wall-clock time vs. structure length for GPU autoregressive decoding ($\sim$98% GPU; $\sim$2% CPU for folding checks) and SAMFEO with five CPU workers. SAMFEO returns one sequence but by default evaluates about 8×10^3 intermediates, so this is not "10^4 draws vs. a single-shot optimizer." We observe gentler length scaling ($|\boldsymbol{y}^\star|^{1.3}$ vs. $|\boldsymbol{y}^\star|^{1.9}$) and $1.74\times$ faster end-to-end time.

7 Related Work

RNA LMs for RNA Analysis. Recent studies train BERT-style encoders on large RNA corpora for structure prediction and analysis, including RNABERT, RNA-FM, RNA-MSM, and RiNALMo (Akiyama & Sakakibara, 2022; Chen et al., 2022; Zhang et al., 2023; Penić et al., 2025). Benchmarks (Zablocki et al., 2025) show such RNA LMs improve analysis, but take no structure input and are encoder-only, hence incapable of *de novo* design. We instead train a structure-conditioned decoder-only LM end-to-end to *generate* sequences with constrained decoding and RL.

Generative Models for RNA Generation. Deep generative models have recently been explored for RNA sequence design: GAN- and VAE-style approaches such as RNAGEN and RfamGen generate family- or target-specific RNAs (Ozden et al., 2023; Sumi et al., 2024; Zhao et al., 2024; Shukueian Tabrizi et al., 2025). These target different purposes than ours, e.g., protein binding rather than conditioning on a target structure.

Other Reinforcement Learning for RNA Design. In Eastman et al. (2018), each state is a full sequence and each action edits one or two nucleotides. Runge et al. (2019) introduced LEARNA and Meta-LEARNA, whose states are partial sequences and whose actions add one unpaired or two paired nucleotides. Obonyo et al. (2022) proposed RNASP, a self-play RL framework inspired by AlphaZero, and Partial RNA design (Runge et al., 2024) extends LEARNA to sequence constraints.

Our use of RL is complementary: rather than training a task-specific agent from scratch, we fine-tune a *single* structure-conditioned LM already trained by supervised learning on solver-generated examples. Our model thus acts as an amortized neural solver, rapidly generating high-quality designs for new targets without per-instance RL optimization while still benefiting from RL's direct optimization of folding-based rewards (see also Tab. S1).

8 Conclusions and Future Work

We have presented the first study of autoregressive language models for RNA design (inverse folding). Combining pretraining, constrained decoding, supervised learning, and reinforcement learning, our final model outperforms or ties the state-of-the-art in ensemble-based metrics on all four datasets while being $1.7\times$ faster. We also release carefully curated datasets for SL and RL training and testing, which will benefit future studies.

Our work can be improved in several ways:

- Decoding speed is a bottleneck, as both RL and testing need many sampled designs. We can use other decoding algorithms such as beam search (Huang et al., 2017), and reduce the $\sim$30% constraint overhead with advanced techniques for LM generation of context-free languages (Dong et al., 2024) (our RNA language is context-free).
- RL is still slow, limiting sample size; faster decoding would allow more data and exploration.
- Finally, Test-Time Reinforcement Learning (TTRL) (Zuo et al., 2025) could train a separate model per test structure, further exploiting test-time scaling.

AI USE STATEMENT

In this work, we used generative AI tools for minor writing assistance (editing and polishing text). We have not used generative AI tools for generating research ideas, experimental design, analysis of results, or creation of figures. We have reviewed all AI-assisted text and take responsibility for the final content of this work, including text, claims, and artifacts.

ETHICS STATEMENT

This work studies computational RNA sequence design and does not involve human subjects or private data. All training and evaluation data are derived from publicly available sources (e.g., Eterna and Rfam-related benchmarks) as described in the paper. We do not foresee direct harmful dual-use risks beyond those inherent to general nucleic-acid design research; designed sequences are evaluated computationally under standard folding models.

REPRODUCIBILITY STATEMENT

We provide anonymized code and data at the anonymous repository linked in the abstract. Training data construction, model architecture and constrained decoding, supervised and RL training details, and evaluation protocols are described in Sections 3 to 6 and the appendix. Settings not stated in those sections, including learning rates, batch sizes, the GRPO group size, and the sampling temperature, are in the released code.

REFERENCES

Bartosz Adamczyk, Maciej Antczak, and Marta Szachniuk. Rnasolo: a repository of cleaned PDB-derived RNA 3d structures. *Bioinformatics*, 38(14):3668–3670, 2022.

Manato Akiyama and Yasubumi Sakakibara. Informative RNA base embedding for RNA structural alignment and clustering by deep representation learning. *NAR Genom. Bioinf.*, 4(1), 02 2022. ISSN 2631-9268. doi: 10.1093/nargab/lqac012. URL https://doi.org/10.1093/nargab/lqac012.

Jeff Anderson-Lee, Eli Fisker, Vineet Kosaraju, Michelle Wu, Justin Kong, Jeehyung Lee, Minjae Lee, Mathew Zada, Adrien Treuille, and Rhiju Das. Principles for predicting RNA secondary structure design difficulty. *J. Mol. Biol.*, 428(5), 2016.

Mirela Andronescu, Anthony P Fejes, Frank Hutter, Holger H Hoos, and Anne Condon. A New Algorithm for RNA Secondary Structure Design. *J. Mol. Biol.*, 336(3), 2004.

Jean Pierre Bachellerie, Jérôme Cavaillé, and Alexander Hüttenhofer. The expanding snoRNA world. *Biochimie*, 84(8), 2002.

Gesine Bauer and Beatrix Suess. Engineered riboswitches as novel tools in molecular biology. *J. Biotech.*, 124(1), 2006.

Édouard Bonnet, Pawel Rzazewski, and Florian Sikora. Designing RNA secondary structures is hard. *J. Comp. Bio.*, 27, 2020.

Jiayang Chen, Zhihang Hu, et al. Interpretable RNA Foundation Model from Unannotated Data for Highly Accurate RNA Structure and Function Predictions. *bioRxiv*, 2022.

Chuong B. Do, Daniel A. Woods, and Serafim Batzoglou. Contrafold: RNA secondary structure prediction without physics-based models. In *Proceedings 14th International Conference on Intelligent Systems for Molecular Biology 2006, Fortaleza, Brazil, August 6-10, 2006*, pp. 90–98, 2006. doi: 10.1093/bioinformatics/btl246. URL https://doi.org/10.1093/bioinformatics/btl246.

Yixin Dong, Charlie Ruan, et al. Xgrammar: Flexible and efficient structured generation engine for large language models. *arXiv:2411.15100*, 2024.

Ivan Dotu, Juan Antonio Garcia-Martin, Betty L. Slinger, Vinodh Mechery, Michelle M. Meyer, and Peter Clote. Complete RNA inverse folding: computational design of functional hammerhead ribozymes. *Nucleic Acids Res.*, 42(18), 09 2014. ISSN 0305-1048. doi: 10.1093/nar/gku740. URL https://doi.org/10.1093/nar/gku740.

Jennifer A. Doudna and Thomas R. Cech. The chemical repertoire of natural ribozymes. *Nature*, 418(6894):222–228, 2002.

Peter K. Eastman, Jade Shi, Bharath Ramsundar, and Vijay S. Pande. Solving the RNA design problem with reinforcement learning. *PLoS Comput. Biol.*, 14(6), 2018. doi: 10.1371/JOURNAL.PCBI.1006176. URL https://doi.org/10.1371/journal.pcbi.1006176.

Sean R. Eddy. Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genetics*, 2(12), 2001.

Sven Findeiß, Maja Etzel, Sebastian Will, Mario Mörl, and Peter F Stadler. Design of artificial riboswitches as biosensors. *Sensors*, 17(9):1990, 2017.

Juan Garcia-Martin, Peter Clote, and Ivan Dotu. RNAiFOLD: a constraint programming algorithm for RNA inverse folding and molecular design. *J. Bioinform. Comput. Biol.*, 11(02), 2013.

Michiaki Hamada. In silico approaches to RNA aptamer design. *Biochimie*, 145:8–14, 2018.

Ivo Hofacker, Walter Fontana, et al. Fast folding and comparison of RNA secondary structures. *Monatshefte fur chemie*, 125, 1994.

Liang Huang, Kai Zhao, and Mingbo Ma. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proc. EMNLP*, 2017.

Liang Huang, He Zhang, Dezhong Deng, Kai Zhao, Kaibo Liu, David A. Hendrix, and David H. Mathews. LinearFold: linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search. *Bioinformatics*, 35(14):i295–i304, 2019. doi: 10.1093/bioinformatics/btz375. URL https://doi.org/10.1093/bioinformatics/btz375.

Rohan V. Koodli, Boris Rudolfs, Jonathan Romano, Hannah K. Wayment-Steele, William A. Dunlap, IV, Eterna Participants, and Rhiju Das. Redesigning the Eterna100 for the Vienna 2 folding engine. *bioRxiv*, 2021. doi: 10.1101/2021.08.26.457839.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proc. ACL*, 2020. doi: 10.18653/V1/2020.ACL-MAIN.703. URL https://doi.org/10.18653/v1/2020.acl-main.703.

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 Mol. Biol.*, 6:26, 2011. doi: 10.1186/1748-7188-6-26. URL https://doi.org/10.1186/1748-7188-6-26.

Bo-Ru Lu, Nikita Haduong, Chien-Yu Lin, Hao Cheng, Noah A Smith, and Mari Ostendorf. Efficient encoder-decoder transformer decoding for decomposable tasks. *arXiv preprint arXiv:2403.13112*, 2024.

Abhinav Mittal, Douglas H Turner, and David H Mathews. NNDB: an expanded database of nearest neighbor parameters for predicting stability of nucleic acid secondary structures. *J. Mol. Biol.*, 436(17), 2024.

Stephen Obonyo, Nicolas Jouandeau, and Dickson Owuor. Designing RNA sequences by self-play. In *Proc. NCTA*, 2022. doi: 10.5220/0011550300003332. URL https://doi.org/10.5220/0011550300003332.

Furkan Ozden, Sina Barazandeh, et al. RNAGEN: A generative adversarial network-based model to generate synthetic RNA sequences to target proteins. *bioRxiv*, July 2023.

Rafael Josip Penić, Tin Vlašić, Roland G Huber, Yue Wan, and Mile Šikić. RiNALMo: general-purpose RNA language models can generalize well on structure prediction tasks. *Nat. Commun.*, 16(1):5671, July 2025.

Fernando Portela. An unexpectedly effective Monte Carlo technique for the RNA inverse folding problem. *bioRxiv*, 2018.

Colin Raffel, Noam Shazeer, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Frederic Runge, Danny Stoll, Stefan Falkner, and Frank Hutter. Learning to design RNA. In *Proc. ICLR*, 2019.

Frederic Runge, Jörg Franke, Daniel Fertmann, Rolf Backofen, and Frank Hutter. Partial RNA design. *Bioinfo.*, 40, 06 2024. ISSN 1367-4811. doi: 10.1093/bioinformatics/btae222. URL https://doi.org/10.1093/bioinformatics/btae222.

Kengo Sato, Manato Akiyama, and Michiaki Hamada. RNA secondary structure prediction using deep learning with thermodynamic integration. *Nature Communications*, 12:941, 2021. doi: 10.1038/s41467-021-21194-4.

Rebecca Schwab, Stephan Ossowski, Markus Riester, Norman Warthmann, and Detlef Weigel. Highly specific gene silencing by artificial microRNAs in Arabidopsis. *Plant Cell*, 18, 2006.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proc. ACL*, 2016. doi: 10.18653/V1/P16-1009. URL https://doi.org/10.18653/v1/p16-1009.

Zhihong Shao, Peiyi Wang, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv 2402.03300*, 2024. URL https://doi.org/10.48550/arXiv.2402.03300.

Sobhan Shukueian Tabrizi, Sina Barazandeh, Helyasadat Hashemi Aghdam, and A. Ercument Cicek. RNAtranslator: Modeling protein-conditional RNA design as sequence-to-sequence natural language translation. *PLOS Comp. Bio.*, 21, 2025. doi: 10.1371/journal.pcbi.1013541. URL https://doi.org/10.1371/journal.pcbi.1013541.

Shunsuke Sumi, Michiaki Hamada, and Hirohide Saito. Deep generative design of RNA family sequences. *Nat. Methods*, 21(3):435–443, March 2024.

Zhen Tan, Yinghan Fu, Gaurav Sharma, and David H. Mathews. TurboFold II: RNA structural alignment and secondary structure prediction informed by multiple homologs. *Nucleic Acids Research*, 45(20):11570–11581, 2017. doi: 10.1093/nar/gkx815.

Akito Taneda. MODENA: a multi-objective RNA inverse folding. *Adv. Appl. Bioinform. Chem.*, 4: 1, 2011.

Hugo Touvron, Thibaut Lavril, et al. Llama: Open and efficient foundation language models. *arXiv 2302.13971*, 2023. URL https://doi.org/10.48550/arXiv.2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NIPS*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Max Ward, Eliot Courtney, and Elena Rivas. Fitness functions for rna structure design. *Nucleic acids research*, 51(7):e40–e40, 2023.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Yijia Xiao, Edward Sun, Yiqiao Jin, and Wei Wang. RNA-GPT: multimodal generative system for RNA sequence understanding. *arXiv 2411.08900*, 2024. URL https://doi.org/10.48550/arXiv.2411.08900.

Ryota Yamagami, Mohammad Kayedkhordeh, David H Mathews, and Philip C Bevilacqua. Design of highly active double-pseudoknotted ribozymes: a combined computational and experimental study. *Nucleic Acids Res.*, 47(1), 11 2018. ISSN 0305-1048. doi: 10.1093/nar/gky1118. URL https://doi.org/10.1093/nar/gky1118.

An Yang, Baosong Yang, et al. Qwen2.5 technical report. *arXiv:2412.15115*, 2024. URL https://doi.org/10.48550/arXiv.2412.15115.

L. Zablocki, Leandro Bugnon, et al. Comprehensive benchmarking of large language models for RNA secondary structure prediction. *Brief. Bioinfo.*, 26(2), 04 2025. ISSN 1477-4054. doi: 10.1093/bib/bbaf137. URL https://doi.org/10.1093/bib/bbaf137.

Joseph N Zadeh, Conrad D Steenberg, Justin S Bois, Brian R Wolfe, Marshall B Pierce, Asif R Khan, Robert M Dirks, and Niles A Pierce. NUPACK: Analysis and design of nucleic acid systems. *J. Comp. Chem.*, 32(1), 2011a.

Joseph N Zadeh, Brian R Wolfe, and Niles A Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *J. Comp. Chem.*, 32(3), 2011b.

He Zhang, Liang Zhang, David H. Mathews, and Liang Huang. LinearPartition: linear-time approximation of RNA folding partition function and base-pairing probabilities. *Bioinformatics*, 36 (Supplement_1):i258–i267, 2020. doi: 10.1093/bioinformatics/btaa460.

Yikun Zhang, Mei Lang, et al. Multiple sequence alignment-based RNA language model and its application to structural inference. *Nucleic Acids Res.*, 52(1), 11 2023. ISSN 0305-1048. doi: 10.1093/nar/gkad1031. URL https://doi.org/10.1093/nar/gkad1031.

Yichong Zhao, Kenta Oono, Hiroki Takizawa, and Masaaki Kotera. GenerRNA: A generative pre-trained language model for de novo RNA design. *PLOS ONE*, 19(10):1–18, 10 2024. doi: 10.1371/journal.pone.0310814. URL https://doi.org/10.1371/journal.pone.0310814.

Tianshuo Zhou, Ning Dai, Sizhen Li, Max Ward, David H Mathews, and Liang Huang. RNA design via structure-aware multifrontier ensemble optimization. *Bioinfo.*, 39, 2023.

Tianshuo Zhou, Wei Yu Tang, David H Mathews, and Liang Huang. Undesignable RNA Structure Identification via Rival Structure Generation and Structure Decomposition. *Proceedings of RECOMB*, 2024.

Yuxin Zuo et al. TTRL: Test-time reinforcement learning. *arXiv:2504.16084*, 2025.

APPENDIX: RNA DESIGN FOR TARGET STRUCTURE BY LANGUAGE MODEL

S1 WHY WE OPTIMIZE $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ RATHER THAN MFE

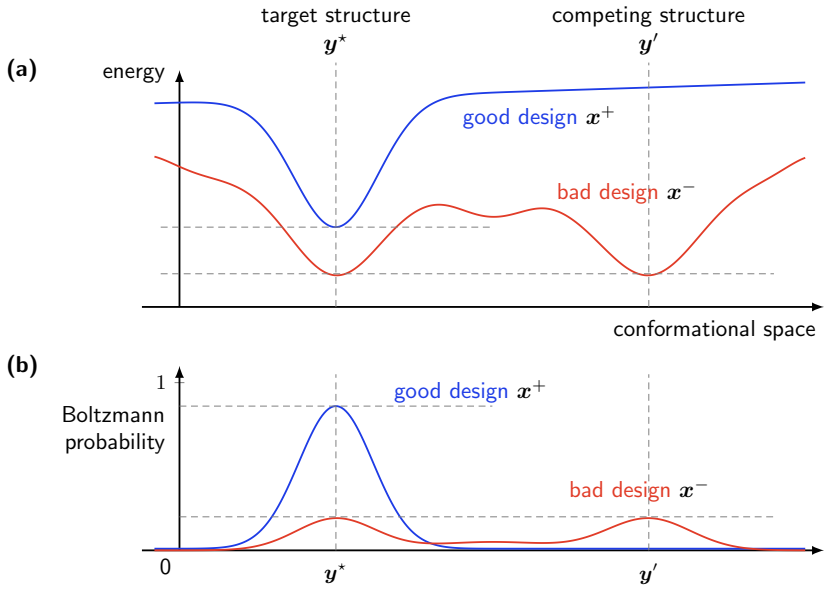


Figure S1: A good design $\boldsymbol{x}^+$ and a bad design $\boldsymbol{x}^-$ for the same target structure $\boldsymbol{y}^\star$. (a) Free energy over conformational space: both designs have their lowest energy at $\boldsymbol{y}^\star$, so both pass the MFE test, but $\boldsymbol{x}^-$ has an equally deep minimum at a competing structure. (b) Boltzmann probability: $\boldsymbol{x}^+$ concentrates its ensemble on $\boldsymbol{y}^\star$, whereas $\boldsymbol{x}^-$ splits it between $\boldsymbol{y}^\star$ and the competing structure.

Why we optimize $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$. Fig. S1 illustrates the difference between the two criteria.

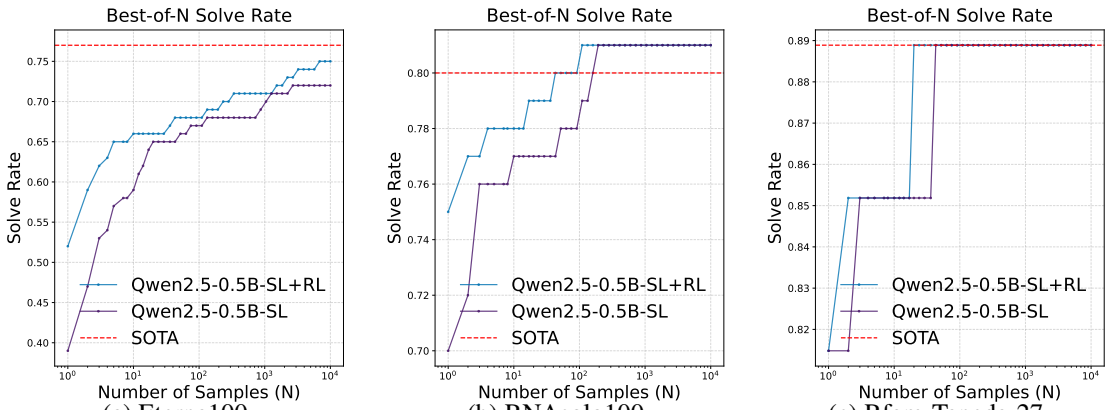
1. **Both designs pass the MFE test, so MFE cannot tell them apart.** In panel (a), $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ each have their lowest point at $\boldsymbol{y}^\star$. By the MFE criterion both are correct solutions.

2. **But $\boldsymbol{x}^-$ has a second structure that is just as stable.** In panel (a), $\boldsymbol{x}^-$ has an energy minimum at $\boldsymbol{y}^\star$ and an equally deep minimum at a competing structure $\boldsymbol{y}'$, so it has two lowest-energy structures instead of one. It therefore has no unique MFE (Sec. 2). $\boldsymbol{x}^+$ has only one minimum, at $\boldsymbol{y}^\star$.

3. **Panel (b) shows what this costs.** $\boldsymbol{x}^+$ puts almost all of its probability on $\boldsymbol{y}^\star$, giving one tall peak. $\boldsymbol{x}^-$ splits its probability between $\boldsymbol{y}^\star$ and $\boldsymbol{y}'$, giving two low bumps of about the same height. So $\boldsymbol{x}^-$ folds into the competing structure roughly as often as into the target, despite passing the MFE test.

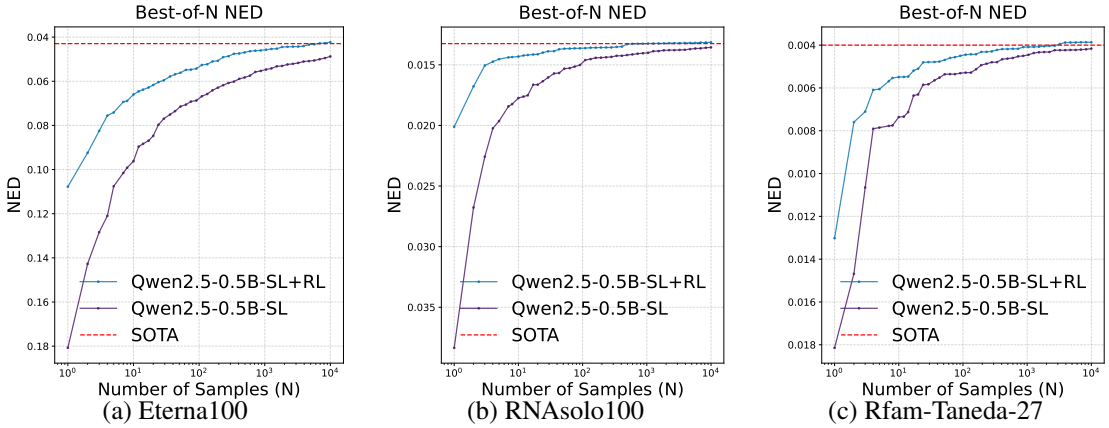
4. **MFE does not tell how many copies fold into the target.** A designed RNA is billions of copies that do not all fold the same way. MFE tells you which shape ranks first, not how many copies take it. What makes an RNA work is how many copies fold into the target, and that is what $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ measures.

This is why we treat ensemble metrics as primary (Sec. 2) and follow Ward et al. (2023) in using $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ as the RL reward (Sec. 4.3).



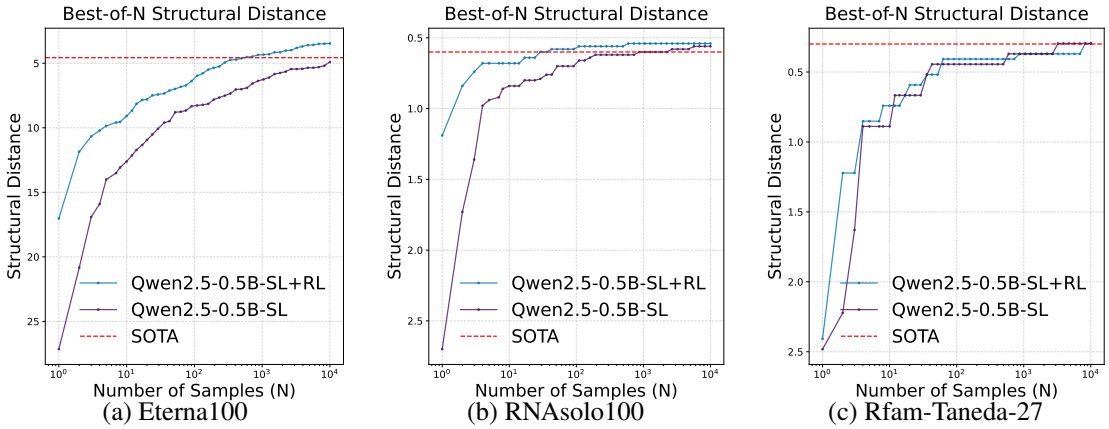
(a) Eterna100 (b) RNAsolo100 (c) Rfam-Taneda-27

Figure S2: Best-of-N curves for SL and SL+RL models in terms of MFE solve rate on three data sets. See Fig. 5 in the main text for the corresponding Best-of-N curves on Boltzmann probability.



(a) Eterna100 (b) RNAsolo100 (c) Rfam-Taneda-27

Figure S3: Best-of-N curves for SL and SL+RL models in terms of Normalized Ensemble Defect (NED) on three data sets. See Fig. 5 in the main text for the corresponding Best-of-N curves on Boltzmann probability.



(a) Eterna100 (b) RNAsolo100 (c) Rfam-Taneda-27

Figure S4: Best-of-N curves for SL and SL+RL models in terms of structural distance on three data sets. See Fig. 5 in the main text for the corresponding Best-of-N curves on Boltzmann probability.

Method	# of MFE/uMFE (↑)	
	Eterna100	Rfam-Taneda-27
Eastman *et al.* (Eastman et al., 2018)	60 / 42	– / –
LEARNA (Runge et al., 2019)	67 / –	23 / –
Meta-LEARNA (Runge et al., 2019)	68 / –	**24** / –
Ours (SL+RL)	**75 / 72**	**24** / 24

Table S1: Comparison of RL methods on Eterna100 and Rfam-Taneda-27. The baselines' numbers are taken from those papers and they only reported MFE metrics. Our work substantially outperforms them on Eterna100.

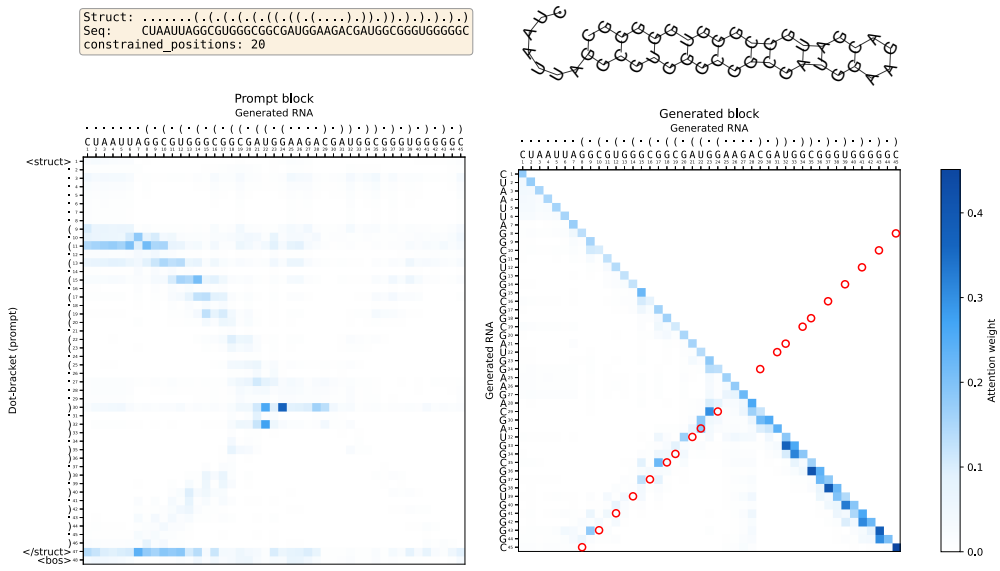
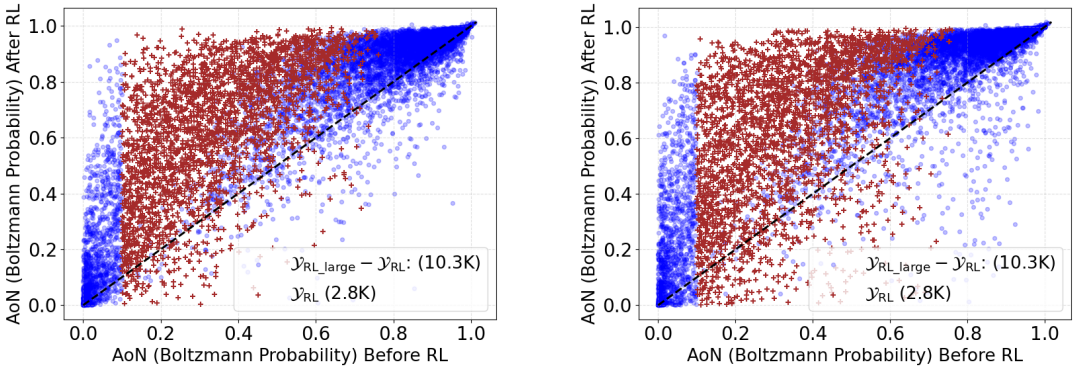


Figure S5: For puzzle #33 of the Eterna100 test set, the figure visualizes causal self-attention in our decoding. The attention weights shown are averaged over all heads of the last self-attention layer. The left panel shows attention between the prompt (dot-bracket structure) and generated RNA tokens, while the right panel shows attention among generated RNA tokens. The target structure (dot-bracket) and generated RNA sequence appear at the top left, and the corresponding secondary structure is shown at the top right; red circles mark constrained base-pair positions.



(a) RL on $\mathcal{Y}_{\text{RL}}$ (b) RL on $\mathcal{Y}_{\text{RL_large}}$

Figure S6: (a, b) Decoding results on $\mathcal{Y}_{\text{RL_large}}$ using RL models trained on $\mathcal{Y}_{\text{RL}}$ and $\mathcal{Y}_{\text{RL_large}}$.

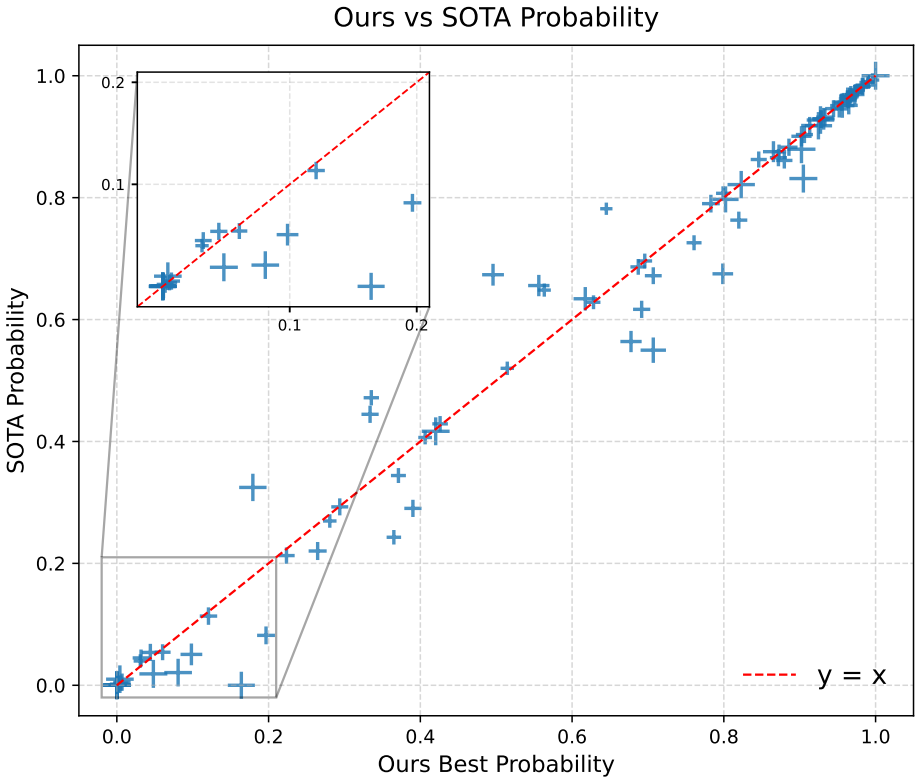


Figure S7: X-Y plot of Boltzmann probability comparing SAMFEO and Ours (SL+RL) model on the Eterna100 test set. Our advantage is most salient on long and hard-to-design puzzles.

S2 GEOMETRIC-MEAN ANALYSIS ON ETERNA100

Design quality spans several orders of magnitude in $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$. An arithmetic mean is driven by the many puzzles where both systems already sit near 1, and it is insensitive to the puzzles where a design fails outright. The geometric mean weights ratios and therefore reflects the full range. For each target we take the best-of-N ViennaRNA probability ($N = 10^4$, same samples as Tab. 1) and report the geometric mean $\exp\left(\frac{1}{n}\sum_i \log p_i\right)$ over all n targets, with no floor and no targets omitted. Since $p(\boldsymbol{y}^\star \mid \boldsymbol{x}) > 0$ for every valid design, the geometric mean is always defined, and we keep very small probabilities at full precision rather than rounding them. We report the full benchmark and the subset excluding the 19 Eterna100-v1 puzzles that are not MFE-solvable under ViennaRNA v2, which Eterna100-v2 (Koodli et al., 2021) subsequently revised to be solvable.

Table S2: Geometric mean of best-of-N $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ on Eterna100.

Set	n	SAMFEO	SL+RL	ratio
All puzzles	100	0.0046	**0.0361**	**7.89**×
Excl. the 19 unsolvable Eterna100-v1 puzzles	81	0.2692	**0.4756**	**1.77**×

The gain is concentrated on long targets. Tab. S3 restricts the comparison to puzzles longer than 200 nt, where design is hardest and where Fig. S7 already shows our largest advantage.

Table S3: Geometric mean of best-of-N $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ on Eterna100 puzzles longer than 200 nt.

Set	n	SAMFEO	SL+RL	ratio
Length $>$ 200 nt, all	29	1.23×10^{-7}	$\mathbf{1.50 \times 10^{-4}}$	**1215**×
Length $>$ 200 nt, excl. the 19 unsolvable	19	1.53×10^{-2}	$\mathbf{1.63 \times 10^{-1}}$	**10.7**×

On long unsolvable puzzles, SAMFEO's best designs have near-zero probability, which drives its geometric mean to 1.23×10^{-7} and gives the $1215\times$ ratio in the first row. Removing the 10 long puzzles that are among the 19 unsolvable ones leaves 19 MFE-solvable long puzzles, on which our designs are still $10.7\times$ better in geometric mean, compared with $1.77\times$ across all 81 solvable puzzles (Tab. S2). The advantage is thus concentrated on long targets.

S3 CROSS-FOLDING-ENGINE EVALUATION OF DESIGNS

A natural concern with reward-based fine-tuning is that the policy may learn to exploit quirks of the scoring model rather than produce designs that are genuinely better. Because our reward is computed with ViennaRNA's thermodynamic parameters, we check whether the improvements persist when the same designs are scored by folding models built on different principles, neither of which was seen during training or used as the RL reward: (i) LinearFold-C and LinearPartition-C (Huang et al., 2019; Zhang et al., 2020), which use the machine-learned CONTRAfold parameters (Do et al., 2006) (beam size 100); and (ii) MXfold2 (Sato et al., 2021), a hybrid model combining neural network scores with Turner thermodynamic parameters. Both evaluations use the same sample pools as Tab. 1, and we compare our SL+RL model against SAMFEO.

S3.A PER-ENGINE BEST-OF-N

For each independent engine and each target, we select the best design under that engine's own scores from the fixed sample pool (no additional sampling), then average over the targets of each benchmark. SAMFEO returns one sequence per target (Sec. 6.4), so its rows below are the same as under freeze-then-evaluate.

Table S4: Eterna100 under independent engines (per-engine BoN; 100 targets).

Method	**LinearFold-C**			**MXfold2**		
	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %
SAMFEO	0.082	0.282	21	0.132	0.216	53
Ours: SL+RL	**0.134**	**0.183**	**40**	**0.203**	**0.132**	**75**

Table S5: RNAsolo100 under independent engines (per-engine BoN; 100 targets).

Method	**LinearFold-C**			**MXfold2**		
	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %
SAMFEO	0.305	0.107	55	0.353	0.073	78
Ours: SL+RL	**0.395**	**0.061**	**63**	**0.499**	**0.036**	**87**

Table S6: Rfam-Taneda-27 under independent engines (per-engine BoN; 27 targets).

Method	**LinearFold-C**			**MXfold2**		
	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %
SAMFEO	0.162	0.099	44	0.205	0.078	**89**
Ours: SL+RL	**0.238**	**0.048**	**56**	**0.310**	**0.034**	**89**

Under this protocol, our designs match or exceed SAMFEO on every metric across all three benchmarks and both predictors: higher p and strictly lower ensemble defect in every case, and a higher MFE match rate under LinearFold-C throughout. The single tie is the MFE match rate under MXfold2 on Rfam-Taneda-27 (both $24/27$), and even there our NED is lower. Notably, GRPO optimizes p under ViennaRNA, so if the model were exploiting ViennaRNA-specific artifacts, p is precisely the quantity that should collapse under an independent engine; instead it remains our strongest off-ViennaRNA metric.

S3.B FREEZE-THEN-EVALUATE

We also report a stricter transfer protocol: for each target, we take the single design with the highest ViennaRNA $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ from each method's pool and evaluate that same design under LinearFold-C and MXfold2, with no re-ranking under the new engine. This asks whether the designs selected by ViennaRNA remain competitive when scored by an independent model.

Table S7: Eterna100 freeze-then-evaluate (Vienna-selected designs; 100 targets).

Method	LinearFold-C			MXfold2		
	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %
SAMFEO	0.082	0.282	21	0.132	0.216	53
Ours: SL+RL	**0.089**	**0.259**	**25**	**0.134**	**0.192**	**57**

Table S8: RNAsolo100 freeze-then-evaluate (Vienna-selected designs; 100 targets).

Method	LinearFold-C			MXfold2		
	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %
SAMFEO	0.305	0.107	55	0.353	0.073	78
Ours: SL+RL	**0.320**	**0.095**	**57**	**0.370**	**0.068**	**81**

Table S9: Rfam-Taneda-27 freeze-then-evaluate (Vienna-selected designs; 27 targets).

Method	LinearFold-C			MXfold2		
	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %	$p(\boldsymbol{y}^\star\|\boldsymbol{x})$	NED$_\downarrow$	MFE %
SAMFEO	0.162	0.099	**44**	0.205	0.078	**89**
Ours: SL+RL	**0.180**	**0.087**	**44**	**0.221**	**0.058**	85

Under freeze-then-evaluate, the ViennaRNA-selected designs from our model score higher p and lower NED than SAMFEO in every case, across both predictors and all three benchmarks. SAMFEO leads on only one metric, the MXfold2 MFE match rate on Rfam-Taneda-27 (89% vs. 85%), and the two methods tie on the LinearFold-C MFE match rate for the same benchmark (44%). Together with the per-engine results above, this shows that our advantage does not depend on re-ranking under each engine: even a single design chosen by ViennaRNA transfers well to independent scoring models.

S4 Sequence diversity after GRPO vs. SFT

We compare SL (SFT) and RL (GRPO) on Eterna100 using the raw, non-deduplicated sampling output: 100 targets $\times$ 10^4 completions $= 10^6$ sequences per model. Retaining duplicates is essential for detecting mode collapse. Diversity among equal-length designs of the same target is measured by normalized Hamming distance.

Table S10: Diversity and composition on Eterna100: SL vs. RL.

Metric	SL	RL	Δ (RL$-$SL)
Mean GC %	57.1	56.3	-0.8
Nucleotide entropy (bits; max 2)	1.671	1.630	-0.040
Unique-sequence rate	0.956	0.840	-0.116
Mean top-1 sequence share	0.003	0.022	$+0.019$
Mean pairwise Hamming distance	0.378	0.291	-0.086

Table S11: Longest homopolymer run on Eterna100 (absolute nt and fraction of target length).

Statistic	SL (nt)	RL (nt)	SL (% of len)	RL (% of len)
Mean	7.88	8.74	7.86	9.32
Median	6	7	6.90	7.55

GRPO does not shift composition toward a simulator-exploiting regime: GC content decreases slightly ($57.1\% \rightarrow 56.3\%$) rather than drifting toward GC-rich sequences, and nucleotide-composition entropy falls only marginally ($1.671 \rightarrow 1.630$ bits). GRPO concentrates the distribution but does not collapse it: after RL, 84.0% of the 10^4 samples per target remain distinct, the single most frequent sequence accounts for only 2.2% of samples, and designs still differ at 29%

of positions on average. Concentration is confined to short targets, where the valid design space is genuinely small; for every target longer than 200 nt, all 10^4 RL samples are unique. Homopolymer runs do not become extreme: the typical longest run increases by less than one nucleotide ($+0.86$ nt, $+1.46$ percentage points of length).

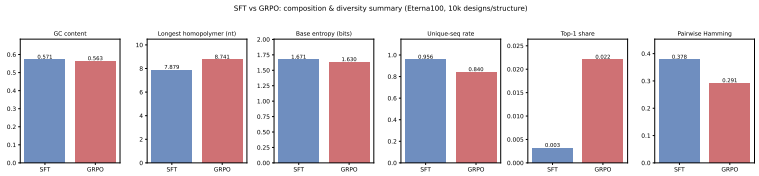


Figure S8: Summary of diversity / composition metrics on Eterna100 after SL vs. after RL (GRPO).