RNA-DFM: MASKED DISCRETE FLOW MATCHING FOR 3D RNA INVERSE DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing RNA sequences that adopt desired three-dimensional structures is central to engineering functional RNA molecules. However, efficient generation with high structural fidelity remains challenging under limited structural supervision. We introduce RNA-DFM, a masked discrete flow matching model that predicts multiple nucleotide identities in parallel conditioned on the target backbone. To expand structural supervision beyond the limited coverage of experimental RNA structures, we augment training with Rfam sequences paired with filtered structure predictions. We further adopt best-of-N distillation, combining structurally selected candidates with native sequences to finetune the generator on training backbones. The resulting model improves structural quality on unseen backbones under limited candidate budgets. Experiments demonstrate improvements in both structural quality and sampling efficiency; Rfam augmentation further improves TM-score and native sequence recovery while reducing RMSD. Among the evaluated methods, pretrained RNA-DFM achieves the highest TM-score, the lowest RMSD, and the highest native sequence recovery at 55.9%. The code for RNA-DFM is available at `https://anonymous.4open.science/r/RNA-DFM-EE8F/`.

1 INTRODUCTION

Ribonucleic acid (RNA) molecules perform diverse biological functions, including catalysis, molecular recognition and gene regulation (Joshi et al., 2025). Designing RNAs with desired functions is therefore an important goal in therapeutics, biotechnology and synthetic biology (Huang et al., 2024). For many functional RNAs, these activities depend on a specific three-dimensional structure. Engineering such RNAs requires identifying nucleotide sequences capable of adopting the desired fold. RNA inverse folding addresses this problem by designing sequences conditioned on a target RNA structure.

Recent advances in geometric deep learning have enabled generative models to capture relationships between RNA sequences and structures. gRNAde uses autoregressive decoding over geometric representations, while RiboDiffusion and RIDE generate sequences through iterative denoising (Joshi et al., 2025; Huang et al., 2024; Hu et al., 2026). However, generating structurally compatible candidates efficiently remains challenging, particularly when multiple designs are required for each target. Sequential nucleotide prediction increases sampling cost with sequence length, motivating parallel generation that preserves structural fidelity. Limited experimental structures further constrain the sequence–structure relationships available for training (Adamczyk et al., 2022; Joshi et al., 2025). Augmentation with predicted structures offers additional supervision, as demonstrated in protein and RNA inverse folding (Hsu et al., 2022; Huang et al., 2024).

Sequence reconstruction trains RNA inverse folding models to recover native nucleotide identities from target structures. However, higher native sequence recovery does not necessarily imply greater structural agreement, as multiple sequences can support similar folds (Joshi et al., 2025; Hu et al., 2026). Structure prediction enables selection based on the agreement between a candidate's predicted structure and the target backbone. Evaluating larger candidate pools increases folding cost, motivating the use of structurally selected sequences as supervision to improve design quality under limited candidate budgets.

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.

We introduce RNA-DFM, a masked discrete flow matching model for RNA 3D inverse design. The model generates sequences by progressively revealing masked nucleotides in parallel, conditioned on the target backbone (Campbell et al., 2024; Gat et al., 2024). This formulation allows the number of sampling steps to be chosen independently of sequence length. To broaden structural supervision, we augment experimental RNA structures with Rfam sequences paired with filtered Boltz predictions (Ontiveros-Palacios et al., 2025; Wohlwend et al., 2024).

We further adopt best-of-N distillation (Sessa et al., 2025) to incorporate structural selection into the generator. Structurally selected candidates are combined with native sequences to finetune the pretrained model. A matched random-candidate control evaluates the contribution of structural ranking. The gains persist when selection and evaluation use different structure predictors, with fewer candidates achieving similar observed structural quality.

Our main contributions are:

- We develop RNA-DFM, a masked discrete flow matching model for RNA 3D inverse design that enables parallel nucleotide generation and achieves higher structural quality at lower sampling cost.
- We augment training with Rfam sequences and predicted structures, increasing native sequence recovery from 53.6% to 55.9% while improving TM-score and reducing RMSD.
- We adopt best-of-N distillation for RNA inverse folding, improving structural quality under limited candidate budgets relative to random candidate finetuning.

2 Related Work

RNA inverse folding. RNA inverse folding methods learn sequence distributions conditioned on the geometry of a target backbone. gRNAde uses geometric graph networks for autoregressive sequence generation from individual structures or conformational ensembles (Joshi et al., 2025). RhoDesign and NA-MPNN also exploit geometric representations for sequence prediction, whereas RiboDiffusion generates sequences through iterative denoising (Wong et al., 2024; Kubaney et al., 2025; Huang et al., 2024). RNAFlow couples inverse folding with structure prediction to jointly generate RNA sequences and structures conditioned on proteins (Nori & Jin, 2024).

Beyond supervised sequence prediction, reinforcement learning and preference optimization incorporate structural feedback into model training. The RIDE diffusion generator is further optimized for individual targets using structural similarity as a reinforcement learning reward (Hu et al., 2026). RiboPO applies direct preference optimization to gRNAde using preferences derived from structural fidelity and thermodynamic criteria (Sun et al., 2025). A related approach in language modeling is best-of-N distillation, which BOND formulates as matching a policy to the distribution induced by candidate selection (Sessa et al., 2025).

RNA structure prediction. RNA structure predictors support inverse folding by assessing the structural compatibility of designed sequences and supplying additional training examples. RhoFold+ integrates RNA language model representations into a network for predicting three-dimensional structures from sequences (Shen et al., 2024). Biomolecular models such as Boltz extend structure prediction to systems containing proteins and nucleic acids (Passaro et al., 2025; Wohlwend et al., 2024). Predicted structures enable self-consistency evaluation in gRNAde and provide feedback for reinforcement learning and preference optimization (Joshi et al., 2025; Hu et al., 2026; Sun et al., 2025). In these three cases the predictor supplies a training or evaluation signal, while the structures used for training remain experimental. A separate use of the same predictors expands inverse folding datasets beyond experimentally determined structures, as demonstrated for proteins and adopted in RiboDiffusion and RhoDesign (Hsu et al., 2022; Huang et al., 2024; Wong et al., 2024).

Masked discrete flow matching. Discrete flow matching models categorical data through continuous-time Markov processes whose transition rates transport a source distribution to a data distribution (Gat et al., 2024). With a masked source, generation progressively replaces mask tokens with categorical values predicted by a denoising model (Campbell et al., 2024; Gat et al.,

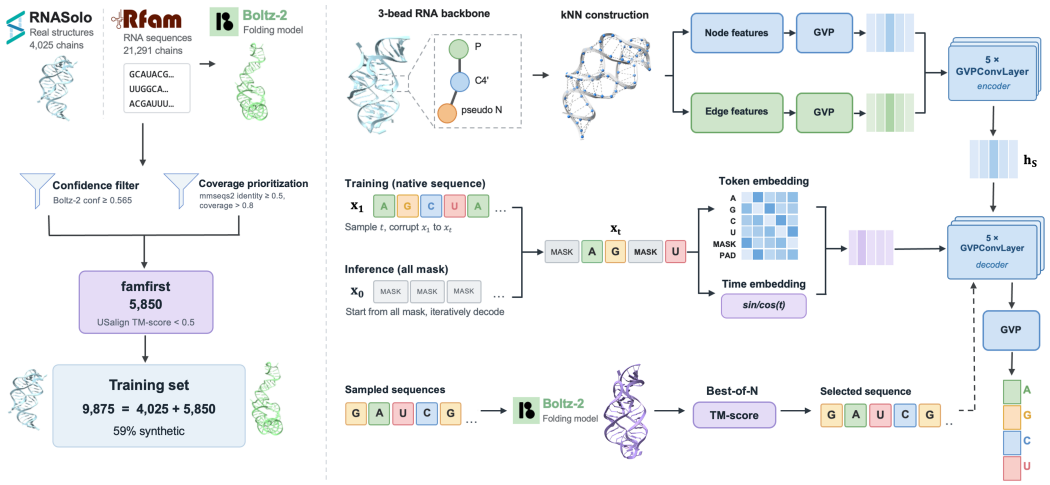


Figure 1: **Overview of RNA-DFM.** Filtered Rfam structure predictions augment experimental structures for pretraining. A GVP encoder maps the target backbone S to features h_S, which condition nucleotide predictions from a partially masked sequence x_t and time t. Generation starts from $x_0 = \mathtt{M}^L$ and reveals randomly selected positions in parallel. For post-training, we apply best-of-N distillation (Sessa et al., 2025), using Boltz scores to select sequence labels that are combined with native sequences for supervised fine-tuning.

2024). MultiFlow combines discrete sequence flows with continuous structural flows for joint protein sequence and structure generation (Campbell et al., 2024). Masked diffusion models similarly learn to reconstruct masked tokens, with training objectives expressed as weighted cross-entropy losses (Sahoo et al., 2024). These formulations support iterative sequence generation in which multiple positions can be updated within a single model evaluation (Campbell et al., 2024; Sahoo et al., 2024).

3 Methods

We introduce RNA-DFM, a masked discrete flow matching model for RNA sequence generation conditioned on 3D backbone structures. As illustrated in Figure 1, a geometric encoder extracts backbone features that condition a nucleotide denoiser. Starting from a fully masked sequence, the model progressively reveals randomly selected positions, predicting their nucleotide identities in parallel. We train the model on experimental RNA structures augmented with filtered structure predictions for Rfam sequences.

3.1 Model architecture

Given a backbone S of length L, we model sequences $x_1 \in \mathcal{A}^L$, where $\mathcal{A} = \{\mathrm{A}, \mathrm{G}, \mathrm{C}, \mathrm{U}\}$. We represent S as a geometric graph $\mathcal{G}_S = (\mathcal{V}, \mathcal{E})$ with nucleotide nodes and edges between spatial neighbours (Joshi et al., 2025). Each nucleotide uses P, $\mathrm{C4}'$, and a pseudoatom constructed from sugar geometry independently of native base identity (Kubaney et al., 2025). Node features encode local distances and directions; edge features encode internucleotide geometry and sequence offsets.

A geometric vector perceptron (GVP) encoder jointly processes scalar and vector features through message passing over spatial neighbours (Jing et al., 2021; Joshi et al., 2025). The resulting node embeddings capture each nucleotide's structural environment:

$$h_S = \mathrm{GVP}_\theta^{\mathrm{enc}}(\mathcal{G}_S) = \{(\mathbf{s}_i, \vec{\mathbf{v}}_i)\}_{i=1}^{L} \tag{1}$$

where $\mathbf{s}_i$ and $\vec{\mathbf{v}}_i$ denote scalar and vector node embeddings, respectively.

A bidirectional GVP decoder combines h_S with the partially masked sequence $x_t \in (\mathcal{A} \cup \{\mathtt{M}\})^L$ and time t to predict nucleotide distributions:

$$q_{\theta,i}(a \mid x_t, t, S) = \left[\mathrm{GVP}_\theta^{\mathrm{dec}}(x_t, t, h_S; \mathcal{G}_S)\right]_{i,a} \tag{2}$$

where the decoder output is a categorical distribution over $a \in \mathcal{A}$ at each position i. Message passing incorporates revealed nucleotides from both sequence directions, allowing all masked positions to be predicted in parallel. These probabilities are invariant to translations and proper rotations of S and parameterize the discrete flow in Section 3.2. Feature construction, layer updates, and conditioning details are provided in Appendix A.

3.2 Discrete flow matching for RNA sequence generation

Probability flow. We construct a discrete flow from the fully masked state $x_0 = \mathtt{M}^L$ to the distribution of native sequences conditioned on the backbone S (Campbell et al., 2024; Gat et al., 2024). Let X_t denote the random sequence at time $t \in [0, 1]$ and x_t its realization. Following the masked interpolation used in discrete flow models for sequence design (Campbell et al., 2024), we define

$$\begin{aligned} p_{t|1}(x_t \mid x_1) &= \prod_{i=1}^{L} \left[t\, \delta\{x_{t,i}, x_{1,i}\} + (1-t)\, \delta\{x_{t,i}, \mathtt{M}\} \right] \\ p_t(x_t \mid S) &= \sum_{x_1 \in \mathcal{A}^L} p_{\mathrm{data}}(x_1 \mid S) p_{t|1}(x_t \mid x_1) \end{aligned} \tag{3}$$

where $\delta\{a, b\}$ is the Kronecker delta, equal to one if $a = b$ and zero otherwise. Each position retains its native nucleotide with probability t and is masked otherwise. The marginal path $p_t(\cdot \mid S)$ connects the mask distribution at $t = 0$ to $p_{\mathrm{data}}(\cdot \mid S)$ at $t = 1$.

Transition rates. A continuous-time Markov chain generates this path by revealing one masked nucleotide at each jump. Let $\mathcal{M}(x_t)$ denote the masked positions and $x_t^{i \to a}$ the state obtained by replacing mask i with nucleotide a. Averaging the conditional transition rates over the denoising posterior gives u_t; replacing this posterior with the predictor from Section 3.1 gives u_t^θ:

$$\begin{aligned} u_t(x_t^{i \to a}, x_t \mid S) &= \frac{p_{1|t,i}(a \mid x_t, S)}{1-t} \\ u_t^\theta(x_t^{i \to a}, x_t \mid S) &= \frac{q_{\theta,i}(a \mid x_t, t, S)}{1-t} \end{aligned} \tag{4}$$

where $p_{1|t,i}(a \mid x_t, S) = \Pr(X_{1,i} = a \mid X_t = x_t, S)$, $i \in \mathcal{M}(x_t)$, $a \in \mathcal{A}$ and $0 < t < 1$. Rates are written with the destination first; all other off-diagonal transitions have zero rate. The shared factor $(1-t)^{-1}$ controls when positions are revealed, while $q_{\theta,i}$ determines nucleotide identities. Appendix B derives these rates and their relation to the probability path.

3.3 Training

Rfam augmentation. To expand the available structural supervision, we augment the experimental training set with Rfam sequences paired with predicted structures (Ontiveros-Palacios et al., 2025). Candidate sequences are sampled to approximate the length distribution of the experimental training set and folded with Boltz (Wohlwend et al., 2024). We select predictions to broaden sequence coverage and remove structures with high similarity to evaluation backbones.

The resulting dataset contains 5,850 predicted structures, which we combine with 4,025 experimental examples (Adamczyk et al., 2022). A sequence audit found no exact matches between the 5,850 Rfam augmentation sequences and the test sequences (Appendix D). For each predicted structure, the originating Rfam sequence provides the nucleotide labels, and the backbone is processed using the same representation as experimental structures. Experimental and predicted examples are sampled uniformly from the combined training set. Appendix D details the construction and filtering procedures for the augmented dataset.

Training objective. The model learns to recover native nucleotide identities from partially masked sequences conditioned on the target backbone. Given a pair (S, x_1) from the combined training set, we sample $t \sim \mathcal{U}[0, 0.9]$ and construct x_t from the conditional path in Equation (3). If no positions are masked, one position is selected uniformly and replaced with the mask token. The expectations

in the training losses are taken over the resulting states, ensuring $|\mathcal{M}(x_t)| \geq 1$. We optimize the predictor q_θ by minimizing the cross-entropy over masked positions:

$$\mathcal{L}_{\mathrm{CE}} = \mathbb{E}_{S,x_1,t,x_t}\left[-\frac{1}{|\mathcal{M}(x_t)|}\sum_{i\in\mathcal{M}(x_t)} \log q_{\theta,i}(x_{1,i} \mid x_t, t, S)\right] \tag{5}$$

We define positive auxiliary rates $\widetilde{r}_{\theta,i,a} = \zeta(\ell_{\theta,i,a}) + 10^{-6}$ from the decoder logits $\ell_{\theta,i,a}$, with a fixed additive floor of 10^{-6} for numerical stability. The conditional target is $r^\star_{i,a} = c(t)\delta\{a, x_{1,i}\}$, assigning rate $c(t)$ to the native nucleotide and zero to the other nucleotides. The auxiliary loss is the generalized KL divergence between these target and predicted rates, averaged over masked positions:

$$\mathcal{L}_{\mathrm{rate}} = \mathbb{E}_{S,x_1,t,x_t}\left[\frac{1}{|\mathcal{M}(x_t)|}\sum_{i\in\mathcal{M}(x_t)}\sum_{a\in\mathcal{A}} d(r^\star_{i,a}, \widetilde{r}_{\theta,i,a})\right] \tag{6}$$

where $d(r, v) = r\log(r/v) - r + v$, with $d(0, v) = v$, and $c(t) = \min\{20, (1 - t + 10^{-3})^{-1}\}$ uses a denominator offset of 10^{-3} and a rate cap of 20. This term regularizes rate magnitudes through the shared logits; sampling uses the categorical probabilities q_θ. The overall training objective is

$$\mathcal{L}_{\mathrm{pre}} = \mathcal{L}_{\mathrm{CE}} + \eta\mathcal{L}_{\mathrm{rate}} \tag{7}$$

where $\eta = 0.05$ controls the contribution of the auxiliary term. Appendix C details the numerical conventions and treatment of empty masks; Appendix B analyzes the auxiliary objective. Appendix C.3 reports a comparison with CE-only training.

3.4 Sampling

The model starts from $x_0 = \mathtt{M}^L$ and uses a uniform time grid $t_k = k/K$ for $k = 0, \ldots, K$, where K is the prescribed number of sampling steps. At step $k = 0, \ldots, K-1$, the model predicts nucleotide distributions conditioned on the current sequence and target backbone. For a masked position, the reveal probability is $\rho_k = (t_{k+1} - t_k)/(1 - t_k) = 1/(K - k)$. The corresponding categorical transition mixes the mask state with the predicted nucleotide distribution (Campbell et al., 2024):

$$\widetilde{x}_{t_{k+1},i} \sim \mathrm{Cat}\Big((1-\rho_k)\delta\{\mathtt{M}, \cdot\} + \rho_k q^{(T)}_{\theta,i}(\cdot \mid x_{t_k}, t_k, S)\Big) \tag{8}$$

where $i \in \mathcal{M}(x_{t_k})$, Cat denotes a categorical distribution, and $q^{(T)}_{\theta,i}$ is the nucleotide predictor at temperature T. Proposals are drawn independently at masked positions, while revealed nucleotides are copied unchanged. If the proposal reveals no nucleotides, the model selects one masked position uniformly and draws its identity from the same predictor. The resulting proposal is accepted as $x_{t_{k+1}}$, with all new identities committed simultaneously.

Sampling ends when the sequence is complete; the final step has $\rho_{K-1} = 1$ and reveals all remaining masks. Appendix E gives the equivalent implementation using update counts, the corrected transition law and the sampling algorithm.

3.5 Best-of-N Distillation

Best-of-N selection ranks candidates by the agreement between their predicted structures and the target backbone, requiring multiple structure predictions per target. To reduce reliance on large candidate pools, we adopt best-of-N distillation (Sessa et al., 2025), using structurally selected sequences as supervision for the masked discrete flow model. We perform one round of candidate generation, structural selection and supervised fine-tuning on training backbones. The updated model is used directly for inference, without regenerating candidates for a second distillation round. For a training backbone S, let $\mathcal{C}(S)$ denote a fixed pool of generated sequences and define the structural score

$$r(x, S) = \frac{1}{2}\sum_{b=1}^{2} \mathrm{TM}(F_b(x), S) \tag{9}$$

Table 1: **Comparison across methods on the test set under Boltz selection@16 and Protenix evaluation.** For each target, 16 candidates are generated, selected by Boltz TM-score, and independently evaluated with Protenix. Reported estimates weight the 22 BGSU equivalence classes equally; brackets denote 95% wild cluster bootstrap-t confidence intervals over structural families, excluding seed variability. Paired statistical comparisons are reported in Table 6.

Method	TM-score (↑)	lDDT (↑)	RMSD (Å ↓)	Recovery (↑)
RNA-DFM	**0.445** [0.270, 0.553]	0.619 [0.498, 0.743]	**9.380** [3.088, 19.304]	**0.559** [0.470, 0.614]
RNA-DFM w/o Rfam	0.439 [0.212, 0.490]	0.611 [0.471, 0.727]	10.541 [3.774, 21.892]	0.536 [0.474, 0.598]
NA-MPNN	0.441 [0.227, 0.636]	**0.638** [0.443, 0.823]	9.738 [2.320, 19.922]	0.522 [0.413, 0.646]
RIDE	0.394 [0.245, 0.497]	0.572 [0.441, 0.704]	11.302 [3.933, 23.563]	0.443 [0.318, 0.532]
gRNAde	0.408 [0.212, 0.554]	0.613 [0.436, 0.760]	10.289 [4.439, 19.178]	0.457 [0.309, 0.609]

where $F_b(x)$ is the bth Boltz prediction and TM-score is normalized by the target length. To control the contribution of native sequence recovery, selection is restricted to candidates whose recovery does not exceed the pool median. Writing this eligible subset as $\mathcal{C}_{\text{elig}}(S)$, the supervision target is

$$x^\star(S) = \underset{x \in \mathcal{C}_{\text{elig}}(S)}{\arg\max}\, r(x, S) \tag{10}$$

The candidate pool is drawn from the same checkpoint that initializes fine-tuning, at a higher sampling temperature than deployment; Appendix G.2 describes temperature selection and evaluation on disjoint sequence-cluster partitions. Every backbone whose pool passes the eligibility filters contributes one label; we apply no margin threshold on the selected score, because such a threshold is computed from the same structure predictions that define the score and therefore favours backbones whose apparent margin is largest by prediction noise.

The resulting pairs form $\mathcal{D}_{\text{sel}} = \{(S, x^\star(S))\}$. Each selected sequence is paired with the original target backbone; its predicted structure is used only for scoring. We combine these pairs with an equal number of native sequence pairs from the same backbones to form $\mathcal{D}_{\text{dist}}$. Initialized from pretrained RNA-DFM, the model fits the sequence labels by minimizing the regularized masked cross-entropy

$$\begin{aligned}\mathcal{L}_{\text{distill}}(\theta) = \\ -\,\mathbb{E}_{S,x_1,t,x_t}\left[\frac{1}{|\mathcal{M}(x_t)|}\sum_{i\in\mathcal{M}(x_t)}\log q_{\theta,i}(x_{1,i} \mid x_t, t, S)\right] \\ +\,\eta\mathcal{L}^{\text{dist}}_{\text{rate}}(\theta)\end{aligned} \tag{11}$$

where (S, x_1) is sampled uniformly from $\mathcal{D}_{\text{dist}}$, with time and masking sampled as in pretraining, including at least one masked position. The auxiliary loss $\mathcal{L}^{\text{dist}}_{\text{rate}}$ evaluates Equation (6) on $\mathcal{D}_{\text{dist}}$, with $\eta = 0.05$. Sampling follows Section 3.4. The recovery restriction applies only to training label construction; deployment selects candidates by TM-score against the target backbone without requiring its native sequence. Appendix G specifies the selection filter, matched control and optimization settings.

4 Experiments

We evaluate RNA-DFM in terms of sequence recovery, structural fidelity and sampling efficiency. We first benchmark the pretrained model and examine the contributions of masked discrete flow matching and Rfam augmentation (Section 4.2).

4.1 Settings

Dataset and split. We use the RNA inverse folding dataset distributed with gRNAde and derived from RNASolo (Adamczyk et al., 2022; Joshi et al., 2025), with 4,025 experimental training records.

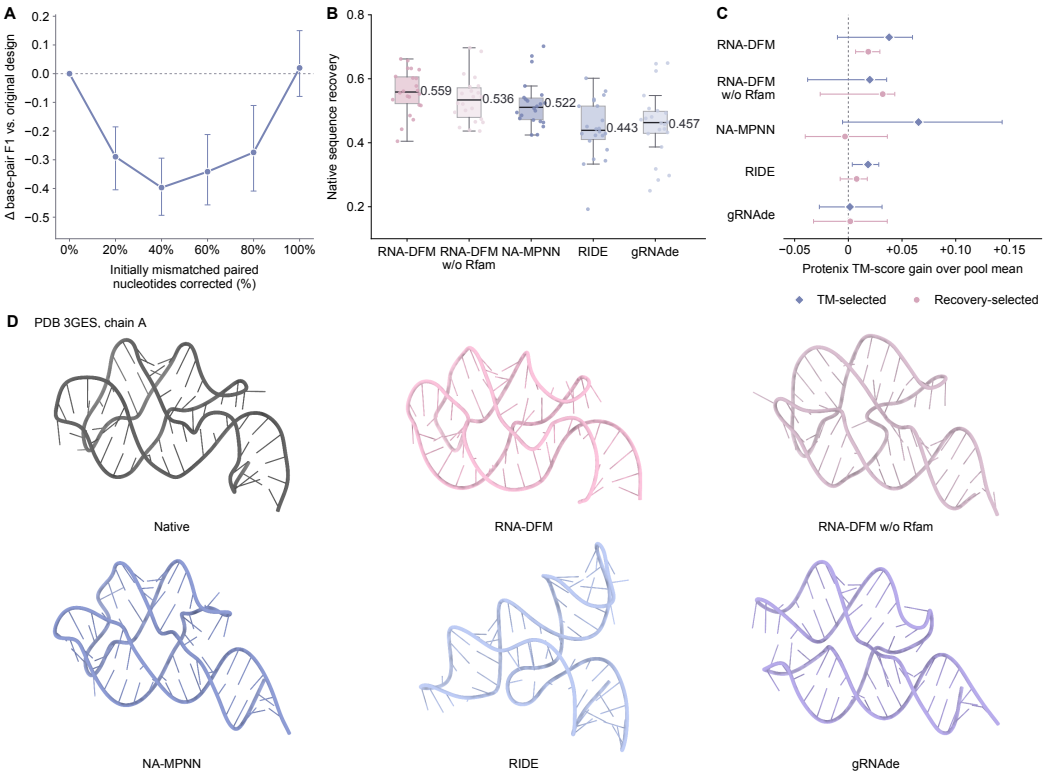


Figure 2: **Sequence recovery and structural evaluation.** (A) Change in ViennaRNA base pair F1 from the original design as mismatched paired nucleotides are replaced by native bases. Bars show paired 95% bootstrap intervals over sequence clusters. (B) Native recovery after Boltz TM-score selection from 16 candidates ($n = 22$ BGSU equivalence classes). Boxes show interquartile ranges and medians; whiskers follow the $1.5\times$ interquartile range rule. (C) Protenix TM-score gains from Boltz TM-score (diamonds) or recovery (circles) selection, relative to *each model's mean over distinct candidates with valid scores*. Bars show paired 95% wild cluster bootstrap intervals over seven structural families. (D) Structural comparison for PDB 3GES, chain A. Plotting and evaluation details are in Appendices F.4 and F.5.

The original `das` split partitions individual sequences. From its 159 test backbones, we exclude 71 whose sequence clusters at 80% identity also contain experimental training records. The resulting test set contains 88 backbones corresponding to 53 unique native sequences (Appendix F.1). For augmented training, we add 5,850 predicted structures of Rfam sequences after excluding predictions with TM-score ≥ 0.5 against evaluation backbones, normalized by target length. Appendix D describes the construction and filtering of the augmentation set.

Metrics. We evaluate sequence recovery and structural fidelity on the same selected designs. Native sequence recovery is the fraction of generated nucleotides matching the native sequence. Structural fidelity is measured by TM-score for global fold similarity, lDDT for local distance agreement, and RMSD for coordinate deviation after alignment. For the main benchmark, we select the candidate with the highest Boltz TM-score against the target backbone from 16 generated sequences for each target and seed (Passaro et al., 2025). The selected sequence is then folded with Protenix to compute structural metrics (Protenix Team et al., 2026). Figure 2C compares this rule with retrospective selection by native recovery on the same candidate pools, measuring gains against each model's mean Protenix TM-score over distinct candidates with valid scores. TM-score selection gives a positive point estimate for all five models, but the paired contrast between the two rules is unresolved for every model and its sign is model dependent, from $+0.0685$ for NA-MPNN to -0.0120 for RNA-DFM without Rfam. Selection by recovery should not be treated as a substitute for selection based on structural agreement with the target backbone. Both native recovery and structural fidelity should therefore be reported. Appendix F.5 reports the gains and intervals for each model. Main benchmark results are averaged across seeds and conformations, with equal weight assigned to each of the 22 BGSU equivalence classes. Metrics and candidate selection are detailed in

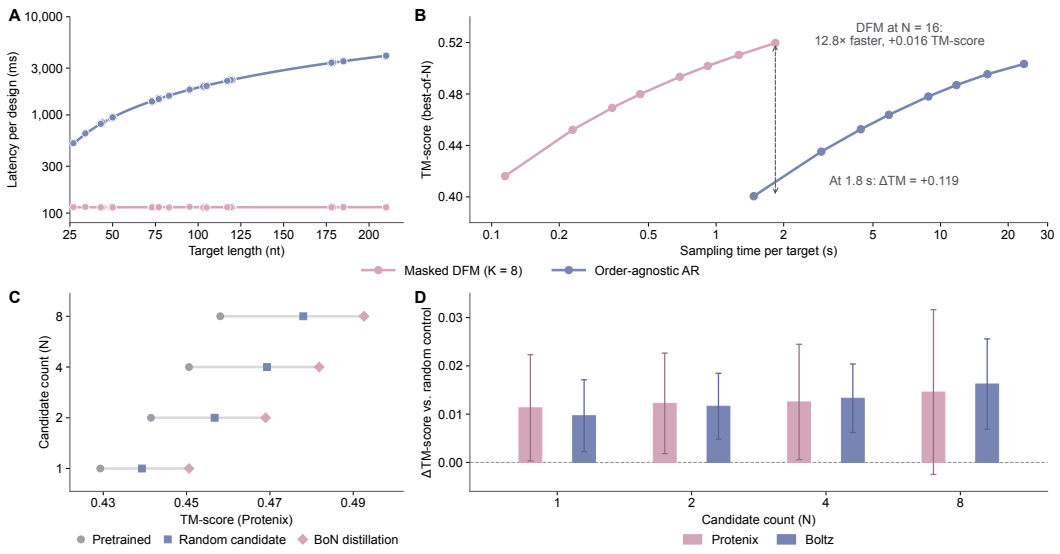


Figure 3: **Sampling efficiency and best-of-N distillation.** (A) Sampling latency across RNA lengths; points show median timings and lines show linear fits. (B) Boltz TM-score versus sampling time for masked DFM and order-agnostic AR. (C) Protenix TM-scores after Boltz selection, grouped by candidate count. (D) TM-score gains over random-candidate fine-tuning, evaluated with Protenix (pink) and Boltz (blue); Boltz supplies both selection and evaluation scores. Panels C–D evaluate partition B (45 backbones, 29 unique native sequences), weighting sequences equally; error bars use $t(28) = 2.048$. Panels C–D use fine-tuning seed 0, and N counts candidate entries, including duplicates. Details are provided in Appendices F.6 and G.

Appendix C.2; aggregation and confidence intervals over structural families appear in Appendix F.2.

Baselines. We compare with NA-MPNN, RIDE and gRNAde (Kubaney et al., 2025; Hu et al., 2026; Joshi et al., 2025). To assess Rfam augmentation, we evaluate RNA-DFM w/o Rfam, trained exclusively on the 4,025 RNASolo training records. All models use the same base-neutral three-atom representation and are trained from scratch with three random seeds. This representation consists of P, $C4'$ and a pseudoatom derived from sugar geometry independently of native base identity (Appendix A). Each model generates 16 candidates per target and seed at temperature $T = 0.1$, with selection performed separately for each seed. RNA-DFM uses random unmasking with $K = 8$ planned steps. Training data, representations and candidate budgets are detailed in Appendix C.

4.2 Pretraining Results

Benchmark performance. RNA-DFM achieves the highest native sequence recovery and TM-score and the lowest RMSD among the evaluated models (Table 1). Native recovery reaches 55.9%, compared with 52.2% for NA-MPNN, 45.7% for gRNAde and 44.3% for RIDE (Figure 2B). For structural fidelity, RNA-DFM obtains a TM-score of 0.445 and an RMSD of 9.380 Å, compared with 0.441 and 9.738 Å for NA-MPNN. NA-MPNN achieves the highest lDDT of 0.638, followed by RNA-DFM at 0.619. Paired analysis identifies significant improvements over RIDE in lDDT ($\Delta = 0.048$, $p = 0.003$) and recovery ($\Delta = 0.115$, $p = 0.028$). The remaining paired confidence intervals include zero, so the corresponding mean differences do not establish statistical superiority (Table 6). Figure 2D provides a qualitative structural comparison for PDB 3GES, chain A using Boltz predictions.

Sampling efficiency. The comparable structural performance of NA-MPNN motivates a comparison with order-agnostic autoregression using the same geometric network architecture. DFM reveals multiple nucleotides per step, reducing sequential computation relative to AR. With eight unmasking steps, DFM has a median sampling latency of 115 ms, whereas AR latency increases with sequence length (Figure 3A). Under Boltz selection and evaluation at $N = 16$, DFM achieves a higher TM-score than AR (0.520 versus 0.503; Figure 3B). The latency fits give a $12.8\times$ sampling speedup. At a matched mean sampling budget of 1.84 s, DFM generates 16 candidates versus one for AR,

improving TM-score by 0.119. Appendix F.6 specifies the checkpoints, scoring protocol and timing calculations, which exclude structure prediction and candidate selection.

Contribution of Rfam data. The two RNA-DFM variants in Table 1 assess the contribution of Rfam augmentation. Adding Rfam data increases native recovery from 53.6% to 55.9% and TM-score from 0.439 to 0.445, while reducing RMSD from 10.541 to 9.380 Å. lDDT also increases from 0.611 to 0.619. Figure 2B shows the corresponding recovery distributions. All four paired intervals include zero, so these improvements in point estimates do not establish statistically significant gains under the evaluation protocol. This comparison concerns the complete training configurations, including their changed training distributions.

Recovery and structural agreement. Native recovery measures agreement with one observed sequence, whereas alternative nucleotide combinations can support similar base pairing patterns (Joshi et al., 2025). Following prior work that evaluates recovery alongside structural consistency (Huang et al., 2024; Hu et al., 2026), we examine this distinction through controlled sequence perturbations.

Figure 2A tests whether increasing recovery within the same design improves predicted base pair agreement. We progressively replace initially mismatched paired nucleotides with their native identities and refold each resulting sequence with ViennaRNA. F1 compares the predicted minimum free energy pair sets of the modified and native sequences. At 80% correction, recovery increases from 0.5155 to 0.7608, while F1 decreases by 0.2743. The paired confidence intervals for all intermediate correction levels (20–80%) lie below zero. Because each comparison uses the same starting design and target, differences in target identity or length cannot explain this decline.

These results show that increasing sequence recovery does not necessarily improve predicted base pair agreement. Base pairing depends on the identities of both nucleotides, whereas the correction procedure replaces bases individually. Restoring a native nucleotide can therefore alter its compatibility with an unchanged partner, even as sequence recovery increases. ViennaRNA avoids dependence on a learned folding model, but the measured outcome remains predicted secondary structure agreement. This distinction is compatible with the augmentation results: improving a generator can benefit both metrics, while individual sequence corrections need not.

4.3 Distillation Results

Distillation protocol. We finetune RNA-DFM on 159 training backbones, pairing each backbone with one structurally selected sequence and its native sequence. Candidate pools are sampled at temperature 0.5 from the pretrained checkpoint used to initialize finetuning. This temperature is selected on partition A of a sequence cluster split, with partition B reserved for evaluating the selected setting (Appendix G.2). To assess the contribution of structural ranking, a matched control replaces each selected sequence with a uniformly sampled alternative from the same eligible subset. Both models use the same native training pairs, initialization and optimization settings, and are finetuned for 12 epochs. We evaluate the final checkpoints from seed 0 on the 45 backbones in partition B, generating 16 candidates per backbone with eight sampling steps and temperature 0.1.

Structural quality. On partition B, Boltz evaluation gives a mean candidate TM-score of 0.411 for distillation versus 0.404 for the random-label control. Expected TM-score after selection from eight candidates increases from 0.499 to 0.510. Positive mean differences are also observed at candidate counts of two and four (Table 9). This readout is circular because Boltz also supplies the selection scores.

Independent evaluation. To assess structural improvements beyond the selection predictor, we evaluate Boltz-selected sequences with Protenix. Distillation yields higher mean TM-scores than the matched random-candidate control across all evaluated candidate budgets (Figure 3D). With two candidates, distillation achieves a mean TM-score close to that of the control with four candidates (Figure 3C). These results support improved candidate efficiency when selection and evaluation use different structure predictors. Appendix G reports the paired comparisons, aggregation procedures and uncertainty estimates.

AI USE STATEMENT

We used OpenAI Codex to assist with literature review, manuscript drafting and revision, and the preparation and explanation of mathematical derivations, proofs and pseudocode. AI assistance also supported discussions of experimental methodology, interpretation of results and refinement of the paper's claims. We additionally used it to create and revise Python code for data processing, analysis and visualization, and to format figures, tables and the LaTeX manuscript. Scripts for figure generation were executed using the supplied result files, with checks of numerical consistency and rendered layouts. The authors directed these activities and retained control over the research objectives, methodological choices and scientific conclusions. We take responsibility for verifying text, code, mathematical arguments, references and quantitative claims developed with AI assistance, and for the final content of this work.

ETHICS STATEMENT

This work develops computational methods for RNA inverse folding using publicly available RNA sequence and structural resources. The study does not involve human participants, identifiable personal data or animal experiments. Our evaluation is computational; predicted structural compatibility does not establish the biological function or safety of a designed sequence. Any downstream experimental or therapeutic use requires validation for the intended application and appropriate biosafety assessment.

REPRODUCIBILITY STATEMENT

An anonymous code repository is available at `https://anonymous.4open.science/r/RNA-DFM-EE8F/`. The appendices detail the model architecture, training procedure, data augmentation and sampling algorithms. Metric definitions and candidate selection are provided in Appendix C.2; evaluation protocols and statistical analyses appear in Appendix F.

REFERENCES

Bartosz Adamczyk, Maciej Antczak, and Marta Szachniuk. RNAsolo: a repository of cleaned PDB-derived RNA 3D structures. *Bioinformatics*, 38(14):3668–3670, 2022. doi: 10.1093/bioinformatics/btac386. URL https://doi.org/10.1093/bioinformatics/btac386.

A. Colin Cameron, Jonah B. Gelbach, and Douglas L. Miller. Bootstrap-based improvements for inference with clustered errors. *The Review of Economics and Statistics*, 90(3):414–427, 2008. doi: 10.1162/rest.90.3.414. URL https://doi.org/10.1162/rest.90.3.414.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5453–5512. PMLR, 2024. URL https://proceedings.mlr.press/v235/campbell24a.html.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, pp. 133345–133385, 2024. doi: 10.52202/079017-4239. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f0d629a734b56a642701bba7bc8bb3ed-Abstract-Conference.html.

Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8946–8970. PMLR, 2022. URL https://proceedings.mlr.press/v162/hsu22a.html.

Tianmeng Hu, Yongzheng Cui, Biao Luo, and Ke Li. RIDER: 3D RNA inverse design with reinforcement learning-guided diffusion. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.16548.

Han Huang, Ziqian Lin, Dongchen He, Liang Hong, and Yu Li. RiboDiffusion: tertiary structure-based RNA inverse folding with generative diffusion models. *Bioinformatics*, 40(Suppl. 1): i347–i356, 2024. doi: 10.1093/bioinformatics/btae259. URL https://doi.org/10.1093/bioinformatics/btae259.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J. L. Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.01411.

Chaitanya K. Joshi, Arian R. Jamasb, Ramon Viñas, Charles Harris, Simon V. Mathis, Alex Morehead, Rishabh Anand, and Pietro Liò. gRNAde: Geometric deep learning for 3D RNA inverse design. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2305.14749.

Andrew Kubaney, Andrew Favor, Lilian McHugh, Raktim Mitra, Robert Pecoraro, Justas Dauparas, Cameron Glasscock, and David Baker. RNA sequence design and protein–DNA specificity prediction with NA-MPNN. *bioRxiv*, 2025. doi: 10.1101/2025.10.03.679414. URL https://www.biorxiv.org/content/10.1101/2025.10.03.679414v2. Preprint.

Neocles B. Leontis and Craig L. Zirbel. Nonredundant 3D structure datasets for RNA knowledge extraction and benchmarking. In *RNA 3D Structure Analysis and Prediction*, volume 27 of *Nucleic Acids and Molecular Biology*, pp. 281–298. Springer, 2012. doi: 10.1007/978-3-642-25740-7_13. URL https://link.springer.com/chapter/10.1007/978-3-642-25740-7_13.

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

Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, 2013. doi: 10.1093/bioinformatics/btt473. URL https://academic.oup.com/bioinformatics/article/29/21/2722/195896.

Divya Nori and Wengong Jin. RNAFlow: RNA structure & sequence design via inverse folding-based flow matching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 38395–38408. PMLR, 2024. URL https://proceedings.mlr.press/v235/nori24a.html.

Nancy Ontiveros-Palacios, Emma Cooke, Eric P Nawrocki, Sandra Triebel, Manja Marz, Elena Rivas, Sam Griffiths-Jones, Anton I Petrov, Alex Bateman, and Blake Sweeney. Rfam 15: RNA families database in 2025. *Nucleic Acids Research*, 53(D1):D258–D267, 2025. doi: 10.1093/nar/gkae1023. URL https://academic.oup.com/nar/article/53/D1/D258/7889252.

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, David Kwabi-Addo, Dominique Beaini, Tommi Jaakkola, and Regina Barzilay. Boltz-2: Towards accurate and efficient binding affinity prediction. *bioRxiv*, 2025. doi: 10.1101/2025.06.14.659707. URL https://doi.org/10.1101/2025.06.14.659707. Preprint.

Protenix Team, Yuxuan Zhang, Chengyue Gong, Hanyu Zhang, Wenzhi Ma, Zhenyu Liu, Xinshi Chen, Jiaqi Guan, Lan Wang, Yanping Yang, Yu Xia, and Wenzhi Xiao. Protenix-v1: Toward high-accuracy open-source biomolecular structure prediction. *bioRxiv*, 2026. doi: 10.64898/2026.02.05.703733. URL https://doi.org/10.64898/2026.02.05.703733. Preprint.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184, 2024. doi: 10.52202/079017-4135. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb0b13cc515724ab8015bc978fdde0ad-Abstract-Conference.html.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot-Desenonges, Johan Ferret, Nino Vieillard, Alexandre Rame, Bobak Shahriari, Sarah Perrin, Abram Friesen, Geoffrey Cideron, Sertan Girgin, Piotr Stanczyk, Andrea Michi, Danila Sinopalnikov, Sabela Ramos Garea, Amélie Héliou, Aliaksei Severyn, Matthew Hoffman, Nikola Momchev, and Olivier Bachem. BOND: Aligning LLMs with best-of-N distillation. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=0tAXMiSufG.

Tao Shen, Zhihang Hu, Siqi Sun, Di Liu, Felix Wong, Jiuming Wang, Jiayang Chen, Yixuan Wang, Liang Hong, Jin Xiao, Liangzhen Zheng, Tejas Krishnamoorthi, Irwin King, Sheng Wang, Peng Yin, James J. Collins, and Yu Li. Accurate RNA 3D structure prediction using a language model-based deep learning approach. *Nature Methods*, 21(12):2287–2298, 2024. doi: 10.1038/s41592-024-02487-0. URL https://www.nature.com/articles/s41592-024-02487-0.

Minghao Sun, Hanqun Cao, Zhou Zhang, Chen Wei, Liang Wang, Tianrui Jia, Zhiyuan Liu, Tianfan Fu, Xiangru Tang, Yejin Choi, Pheng-Ann Heng, Fang Wu, and Yang Zhang. RiboPO: Preference optimization for structure- and stability-aware RNA design. *arXiv preprint arXiv:2510.21161*, 2025. doi: 10.48550/arXiv.2510.21161. URL https://arxiv.org/abs/2510.21161.

Matthew D. Webb. Reworking wild bootstrap-based inference for clustered errors. *Canadian Journal of Economics*, 56(3):839–858, 2023. doi: 10.1111/caje.12661. URL https://onlinelibrary.wiley.com/doi/10.1111/caje.12661.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, and Regina Barzilay. Boltz-1: Democratizing biomolecular interaction modeling. *bioRxiv*, 2024. doi: 10.1101/2024.11.19.624167. URL https://www.biorxiv.org/content/10.1101/2024.11.19.624167. Preprint.

Felix Wong, Dongchen He, Aarti Krishnan, Liang Hong, Alexander Z. Wang, Jiuming Wang, Zhihang Hu, Satotaka Omori, Alicia Li, Jiahua Rao, Qinze Yu, Wengong Jin, Tianqing Zhang, Katherine Ilia, Jack X. Chen, Shuangjia Zheng, Irwin King, Yu Li, and James J. Collins. Deep generative design of RNA aptamers using structural predictions. *Nature Computational Science*, 4(11):829–839, 2024. doi: 10.1038/s43588-024-00720-6. URL https://www.nature.com/articles/s43588-024-00720-6.

Chengxin Zhang, Morgan Shine, Anna Marie Pyle, and Yang Zhang. US-align: universal structure alignments of proteins, nucleic acids, and macromolecular complexes. *Nature Methods*, 19(9): 1109–1115, 2022. doi: 10.1038/s41592-022-01585-1. URL https://www.nature.com/articles/s41592-022-01585-1.

APPENDIX

A Backbone representation and architecture **15**

B Flow matching analysis **16**

B.1 Notation and masking paths 16

B.2 Marginal transition rates 17

B.3 Cross-entropy and rate regularization 19

B.4 Numerical conventions 20

C Pretraining and model selection **21**

C.1 Pretraining protocol 21

C.2 Metrics and candidate selection 22

C.3 Rate regularization 23

D Rfam data augmentation **24**

D.1 Dataset construction 24

D.2 Augmentation comparisons 26

E Random unmasking **26**

F Evaluation protocols and analyses **28**

F.1 Datasets and evaluation units 29

F.2 Aggregation and statistical inference 29

F.3 Paired model comparisons 30

F.4 Recovery and structural agreement 31

F.5 Candidate selection 32

F.6 Sampling latency and structural quality 33

G Best-of-N distillation **34**

G.1 Supervision and finetuning 34

G.2 Temperature of the supervision pool 35

G.3 Evaluation protocol 35

G.4 Evaluation limitations 36

A BACKBONE REPRESENTATION AND ARCHITECTURE

Coordinates and residue filtering. Each RNA residue must contain finite coordinates for P, $C4'$, $O4'$, $C1'$ and $C2'$. Residues missing any required atom are removed before constructing the graph. The same indices of retained positions must be applied to sequence labels and native coordinates used for structural scoring. To represent the sugar–base attachment geometry without using nucleotide identity, we construct a virtual nitrogen site n_i^* from the sugar atoms, following the NA-MPNN implementation (Kubaney et al., 2025). Let $o_i, c_i, d_i \in \mathbb{R}^3$ be the Cartesian coordinates, in Å, of $O4'$, $C1'$ and $C2'$ in residue i, respectively. The displacement vectors $b_i = c_i - o_i$ and $v_i = d_i - c_i$ point from $O4'$ to $C1'$ and from $C1'$ to $C2'$. The virtual site is placed relative to $C1'$ as

$$n_i^* = c_i + w_\times(b_i \times v_i) + w_b b_i + w_v v_i \tag{12}$$

where $\times$ denotes the vector cross product, so $b_i \times v_i$ is perpendicular to the plane of the three sugar atoms. The displacement vectors and their cross product are used without normalization. For coordinates expressed in Å, the fixed NA-MPNN coefficients are $w_\times \approx -0.570$, $w_b \approx 0.511$ and $w_v \approx -0.531$. The implementation retains their original numerical precision, and these coefficients are not optimized during training. The superscript $*$ marks a constructed coordinate rather than an observed atom or an optimization solution. The base-neutral three-atom representation supplied to every method in the main benchmark is $(p_i, \mathbf{r}_i, n_i^*)$, where p_i and $\mathbf{r}_i$ denote the P and $C4'$ coordinates. The additional sugar coordinates are used to construct n_i^*, rather than supplied as additional coordinate sites to the models. The construction uses neither native N1/N9 coordinates nor the residue label, so changing the sequence labels while holding sugar coordinates fixed leaves the representation unchanged. The resulting representation transforms consistently under proper rotations and translations.

Node and edge features. Write $\mathbf{r}_i$ for the $C4'$ coordinate and p_i for phosphate. The four displacement channels are

$$\mathbf{r}_{i+1} - \mathbf{r}_i, \qquad \mathbf{r}_{i-1} - \mathbf{r}_i, \qquad n_i^* - \mathbf{r}_i, \qquad p_i - \mathbf{r}_i$$

Missing forward or backward neighbours receive zero displacement vectors. Each length is expanded into 16 radial basis functions,

$$\phi_j(d) = \exp\left[-\left(\frac{d-\mu_j}{\sigma}\right)^2\right], \qquad \mu_j = \frac{20j}{15}, \quad j = 0, \ldots, 15 \tag{13}$$

where d is a displacement length in Å, and the 16 centers μ_j are uniformly spaced from 0 to 20 Å, including both endpoints. The denominator 15 therefore gives the spacing between adjacent centers as $20/15$ Å. The fixed width $\sigma = 20/16 = 1.25$ Å controls how rapidly each basis function decays away from its center; the centers and width are not learned. The four distances contribute $4 \times 16 = 64$ radial features. These are concatenated with the 21-dimensional type encoding retained from the shared protein–RNA implementation. For every RNA node this encoding is the fixed vector $(0, \ldots, 0, 1)$, giving $64 + 21 = 85$ scalar input channels without revealing nucleotide identity. The four normalized displacements form the vector channels, with undefined normalizations of zero vectors replaced by zero.

Edges are formed by computing $k = \min(32, L-1)$ nearest neighbours and taking the union with their reverse edges. For an edge from j to i, the displacement is $\mathbf{r}_j - \mathbf{r}_i$. Its scalar features concatenate 16 radial features of $\|\mathbf{r}_j - \mathbf{r}_i\|$ computed using Equation (13) with 16 sinusoidal features encoding the sequence offset $j - i$. The normalized displacement supplies one vector feature; symmetrization can increase the degree beyond 32. Our model uses a backbone graph containing only RNA and a bidirectional decoder, without optional base pair edges or additional backbone channels.

Message passing and symmetry. Write $(\mathbf{s}_i, \vec{\mathbf{v}}_i)$ for the scalar–vector node embedding and $\mathbf{e}_{ji}$ for the encoded scalar–vector edge features, omitting layer indices. Let $\mathcal{N}_i$ be the set of nodes with directed edges into node i. The encoder and bidirectional decoder use mean aggregation and two residual updates per layer:

$$(\bar{\mathbf{s}}_i, \bar{\vec{\mathbf{v}}}_i) = \mathrm{LN}_1(\mathbf{s}_i, \vec{\mathbf{v}}_i)$$

$$
\begin{aligned}
(\mathbf{m}_i, \vec{\mathbf{m}}_i) &= \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathrm{GVP}_{\mathrm{msg}}\left([(\bar{\mathbf{s}}_j, \bar{\vec{\mathbf{v}}}_j), \mathbf{e}_{ji}, (\bar{\mathbf{s}}_i, \bar{\vec{\mathbf{v}}}_i)]\right) \\
(\mathbf{s}'_i, \vec{\mathbf{v}}'_i) &= (\mathbf{s}_i, \vec{\mathbf{v}}_i) + \mathrm{Dropout}(\mathbf{m}_i, \vec{\mathbf{m}}_i) \\
(\mathbf{s}_i^{\mathrm{next}}, \vec{\mathbf{v}}_i^{\mathrm{next}}) &= (\mathbf{s}'_i, \vec{\mathbf{v}}'_i) + \mathrm{Dropout}\left(\mathrm{GVP}_{\mathrm{ff}}(\mathrm{LN}_2(\mathbf{s}'_i, \vec{\mathbf{v}}'_i))\right)
\end{aligned} \tag{14}
$$

where LN_1 and LN_2 are layer normalization operators, concatenation acts separately on scalar and vector channels, and empty neighbourhoods contribute a zero message. Equation (14) specifies the layer updates of the encoder in Equation (1), including normalization before each update, residual connections, and dropout. The message function comprises three GVP blocks, while the feedforward function uses two blocks with intermediate dimensions $(1024, 48)$. The same graph topology is used throughout the encoder and decoder, and encoded edge features remain fixed across their layers.

Translation cancels from displacement features, and proper rotations preserve their lengths and rotate their vector components consistently. The pseudoatom contains a cross product, so reflection equivariance is not assumed for the complete input representation. GVP operations preserve the scalar/vector transformation rules, yielding nucleotide logits invariant to rotations after the scalar output projection. The edge positional encoding is $[\cos(d\nu_h), \sin(d\nu_h)]_{h=0}^{7}$, where $d = j - i$ for the edge from j to i and $\nu_h = 10000^{-2h/16}$. The constant 10000 sets the frequency scale; eight fixed frequencies each contribute a cosine and sine component, yielding 16 channels. Graph sequence offsets follow indices of retained nodes after filtering for valid atoms; original residue indices are preserved separately for scoring correspondence.

Encoder and decoder. Input scalar/vector tuples are normalized and projected with geometric vector perceptrons (Jing et al., 2021). Five encoder layers use geometric message passing, residual connections, normalization before each update and feedforward updates. The encoder processes a single target conformation; no conformational ensemble is required. The final states give $h_S = \{(\mathbf{s}_i^{(5)}, \vec{\mathbf{v}}_i^{(5)})\}_{i=1}^{L}$, corresponding to $(\mathbf{s}_i, \vec{\mathbf{v}}_i)$ in Equation (1). For each node, the decoder adds an embedding of token identity and time to the encoder's scalar channels:

$$
e_t = \mathrm{MLP}_t\left([t, \sin(2\pi\omega t), \cos(2\pi\omega t)]\right) \tag{15}
$$

$$
\mathbf{s}_{i,\mathrm{dec}}^{(0)} = \mathbf{s}_i^{(5)} + \mathrm{MLP}_{\mathrm{in}}([e_{\mathrm{tok}}(x_{t,i}), e_t]) \tag{16}
$$

$$
\vec{\mathbf{v}}_{i,\mathrm{dec}}^{(0)} = \vec{\mathbf{v}}_i^{(5)} \tag{17}
$$

where e_{tok} is the token embedding and $\omega \in \mathbb{R}^8$ contains learned time frequencies. The time input has 17 features, including t. The time MLP has widths $17 \to 512 \to 256$, and the input MLP has widths $512 \to 256 \to 256$. Both use SiLU activations between their linear layers. Five decoder layers operate on these node features and the encoded edges. The decoder in Equation (2) includes the scalar output projection and categorical normalization:

$$
\begin{aligned}
\ell_{\theta,i} &= \mathrm{GVP}_{\mathrm{out}}\left(\mathbf{s}_{i,\mathrm{dec}}^{(5)}, \vec{\mathbf{v}}_{i,\mathrm{dec}}^{(5)}\right) \\
q_{\theta,i}(a \mid x_t, t, S) &= \left[\mathrm{softmax}(\ell_{\theta,i})\right]_a
\end{aligned} \tag{18}
$$

where $\ell_{\theta,i} \in \mathbb{R}^4$ contains the nucleotide logits at position i.

B Flow matching analysis

B.1 Notation and masking paths

Fix a backbone S and a finite sequence length L throughout this section. We denote the random sequence state by X_t and its realized value by x_t; x, y denote generic states when the time is specified separately. The symbol x_1 denotes a complete sequence, while X_1 denotes the random terminal sequence. The data distribution is supported on $\mathcal{A}^L$, and the evolving state lies in $(\mathcal{A} \cup \{\mathtt{M}\})^L$. We use $\delta\{a, b\}$ for equality of discrete states: it equals one when $a = b$ and zero otherwise. For sequences of equal length, $\delta\{x, y\} = \prod_{i=1}^{L} \delta\{x_i, y_i\}$, so it tests equality at every position. Write $\mathcal{R}(x) = \{i : x_i \neq \mathtt{M}\}$ and $\mathcal{M}(x) = \{i : x_i = \mathtt{M}\}$ for the revealed and masked sets, respectively.

Table 2: **RNA-DFM architecture and hyperparameters for pretraining with Rfam augmentation.**

Setting	Value
Node input (scalar, vector)	$(85, 4)$
Edge input (scalar, vector)	$(32, 1)$
Node hidden dimensions	$(256, 24)$
Edge hidden dimensions	$(128, 4)$
Encoder / decoder layers	$5/5$
Token embedding dimension	256
Output vocabulary	A, G, C, U
Dropout	0.3
Training time interval	$[0, 0.9]$
Rate regularization weight	0.05
Optimizer	AdamW
Peak / minimum learning rate	$10^{-4}/10^{-6}$
Weight decay	5×10^{-4}
Gradient norm clipping	1.0
Warmup / total epochs	$5/100$
Graphs per device / training devices	$1/3$
Coordinate jitter	Disabled

The operation $x^{i \to a}$ replaces the token at position i with a and leaves all other positions unchanged. Reveal transitions restrict i to $\mathcal{M}(x)$ and a to $\mathcal{A}$. We orient $u_t(y, x \mid S)$ from source x to destination y, so columns corresponding to source states of the generator sum to zero.

The following derivation specializes the conditional-to-marginal construction of Campbell et al. (2024), also developed in Gat et al. (2024), to masked sequences conditioned on the backbone. It concerns the ideal path and exact posterior, before introducing learned approximations or safeguards for sampling with finite steps. These standard identities provide the mathematical basis for our parameterization and are not claimed as new flow matching results.

Let $\kappa : [0, 1] \to [0, 1]$ be continuously differentiable and nondecreasing, with $\kappa(0) = 0$, $\kappa(1) = 1$ and $0 < \kappa(t) < 1$ for $0 < t < 1$. Define the conditional and marginal paths by

$$\begin{aligned} p^{\kappa}_{t|1}(x \mid x_1) &= \prod_i \left[\kappa(t)\, \delta\{x_i, x_{1,i}\} + (1 - \kappa(t))\, \delta\{x_i, \mathtt{M}\}\right] \\ p^{\kappa}_t(x \mid S) &= \sum_{x_1} p_{\mathrm{data}}(x_1 \mid S) p^{\kappa}_{t|1}(x \mid x_1) \end{aligned} \tag{19}$$

Setting $\kappa(t) = t$ recovers the probability path in Equation (3) and the hazard $(1 - t)^{-1}$ used in Equation (4).

B.2 Marginal transition rates

Proposition B.1 (Masked conditional-to-marginal construction)**.** *Under the preceding assumptions, the posterior $p_{1|t}(x_1 \mid x, S)$ is independent of t for each fixed state with positive path probability. For $0 < t < 1$, a generator of the marginal path has off-diagonal rates*

$$u_t(x^{i \to a}, x \mid S) = \lambda_\kappa(t) p_{1|t,i}(a \mid x, S), \qquad \lambda_\kappa(t) = \frac{\dot\kappa(t)}{1 - \kappa(t)} \tag{20}$$

where $i \in \mathcal{M}(x)$ and $a \in \mathcal{A}$. All other off-diagonal rates vanish, and the diagonal is $u_t(x, x \mid S) = -|\mathcal{M}(x)|\lambda_\kappa(t)$. The resulting path starts at the distribution concentrated on the fully masked state and approaches $p_{\mathrm{data}}(\cdot \mid S)$ as $t \uparrow 1$.

Proof. Write $r(x) = |\mathcal{R}(x)|$, $m(x) = |\mathcal{M}(x)|$, and $C_x(x_1) = \prod_{i \in \mathcal{R}(x)} \delta\{x_{1,i}, x_i\}$. The empty product equals one, so every complete sequence is compatible with the fully masked state. For

$0 < t < 1$, Bayes' rule gives

$$
\begin{aligned}
p^{\kappa}_{t|1}(x \mid x_1) &= \kappa(t)^{r(x)}(1-\kappa(t))^{m(x)} C_x(x_1) \\
p_{1|t}(x_1 \mid x, S) &= \frac{p_{\text{data}}(x_1 \mid S) p^{\kappa}_{t|1}(x \mid x_1)}{\sum_{x'_1} p_{\text{data}}(x'_1 \mid S) p^{\kappa}_{t|1}(x \mid x'_1)} \\
&= \frac{p_{\text{data}}(x_1 \mid S) \kappa(t)^{r(x)}(1-\kappa(t))^{m(x)} C_x(x_1)}{\kappa(t)^{r(x)}(1-\kappa(t))^{m(x)} \sum_{x'_1} p_{\text{data}}(x'_1 \mid S) C_x(x'_1)} \\
&= \frac{p_{\text{data}}(x_1 \mid S) C_x(x_1)}{\sum_{x'_1} p_{\text{data}}(x'_1 \mid S) C_x(x'_1)}
\end{aligned}
\tag{21}
$$

The final expression has no time dependence, and its denominator is positive whenever $p^{\kappa}_t(x \mid S) > 0$.

The probability flux at a single coordinate satisfies

$$
\begin{aligned}
\partial_t p(X_{t,i} = \mathtt{M} \mid x_1) &= -\dot{\kappa}(t) = -\lambda_\kappa(t)(1-\kappa(t)) \\
\partial_t p(X_{t,i} = x_{1,i} \mid x_1) &= \dot{\kappa}(t) = \lambda_\kappa(t)(1-\kappa(t))
\end{aligned}
\tag{22}
$$

Define the corresponding sequence generator by

$$
\begin{aligned}
u_t(y, x \mid x_1) &= \lambda_\kappa(t) \sum_{i \in \mathcal{M}(x)} \delta\{y, x^{i \to x_{1,i}}\}, \qquad y \neq x \\
u_t(x, x \mid x_1) &= -m(x)\lambda_\kappa(t)
\end{aligned}
\tag{23}
$$

For a compatible destination y, write $y^{i \to \mathtt{M}}$ for its predecessor obtained by remasking a revealed position. Its incoming flux and outgoing flux give the full conditional forward equation:

$$
\begin{aligned}
\sum_x u_t(y, x \mid x_1) p^{\kappa}_{t|1}(x \mid x_1) &= \lambda_\kappa(t) \sum_{i \in \mathcal{R}(y)} p^{\kappa}_{t|1}(y^{i \to \mathtt{M}} \mid x_1) - m(y)\lambda_\kappa(t) p^{\kappa}_{t|1}(y \mid x_1) \\
&= \lambda_\kappa(t) \left[\frac{r(y)(1-\kappa(t))}{\kappa(t)} - m(y) \right] p^{\kappa}_{t|1}(y \mid x_1) \\
&= \left[\frac{r(y)\dot{\kappa}(t)}{\kappa(t)} - \frac{m(y)\dot{\kappa}(t)}{1-\kappa(t)} \right] p^{\kappa}_{t|1}(y \mid x_1) \\
&= \partial_t \Big[\kappa(t)^{r(y)}(1-\kappa(t))^{m(y)} \Big] = \partial_t p^{\kappa}_{t|1}(y \mid x_1)
\end{aligned}
\tag{24}
$$

For an incompatible y, both sides vanish: every predecessor with positive mass is compatible, and the generator only reveals native tokens.

On source states with positive mass, define $u_t(y, x \mid S) = \sum_{x_1} p_{1|t}(x_1 \mid x, S) u_t(y, x \mid x_1)$. Differentiating the finite mixture and applying Bayes' rule yields

$$
\begin{aligned}
\partial_t p^{\kappa}_t(y \mid S) &= \partial_t \sum_{x_1} p_{\text{data}}(x_1 \mid S) p^{\kappa}_{t|1}(y \mid x_1) \\
&= \sum_{x_1} p_{\text{data}}(x_1 \mid S) \partial_t p^{\kappa}_{t|1}(y \mid x_1) \\
&= \sum_{x_1} p_{\text{data}}(x_1 \mid S) \sum_x u_t(y, x \mid x_1) p^{\kappa}_{t|1}(x \mid x_1) \\
&= \sum_x \sum_{x_1} u_t(y, x \mid x_1) \underbrace{p_{\text{data}}(x_1 \mid S) p^{\kappa}_{t|1}(x \mid x_1)}_{p^{\kappa}_t(x|S) p_{1|t}(x_1|x,S)} \\
&= \sum_x p^{\kappa}_t(x \mid S) \sum_{x_1} p_{1|t}(x_1 \mid x, S) u_t(y, x \mid x_1)
\end{aligned}
$$

$$= \sum_x u_t(y, x \mid S) p_t^\kappa(x \mid S) \tag{25}$$

Source states with zero mass contribute zero and may be assigned any valid generator column. For a masked position i, the nonzero off-diagonal entries and column sums are

$$\begin{aligned} u_t(x^{i\to a}, x \mid S) &= \sum_{x_1} p_{1|t}(x_1 \mid x, S)\lambda_\kappa(t)\,\delta\{x_{1,i}, a\} \\ &= \lambda_\kappa(t) p_{1|t,i}(a \mid x, S) \geq 0 \\ u_t(x, x \mid S) &= \sum_{x_1} p_{1|t}(x_1 \mid x, S)[-m(x)\lambda_\kappa(t)] \\ &= -m(x)\lambda_\kappa(t) \\ \sum_y u_t(y, x \mid S) &= -m(x)\lambda_\kappa(t) + \lambda_\kappa(t) \sum_{i\in\mathcal{M}(x)} \underbrace{\sum_a p_{1|t,i}(a \mid x, S)}_{1} \\ &= 0 \end{aligned} \tag{26}$$

Finally, the finite mixture has the boundary values

$$\begin{aligned} p_0^\kappa(x \mid S) &= \sum_{x_1} p_{\text{data}}(x_1 \mid S)\delta\{x, \mathtt{M}^L\} = \delta\{x, \mathtt{M}^L\} \\ \lim_{t\uparrow 1} p_t^\kappa(x \mid S) &= \sum_{x_1} p_{\text{data}}(x_1 \mid S)\delta\{x, x_1\} = \begin{cases} p_{\text{data}}(x \mid S), & x \in \mathcal{A}^L \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{27}$$

The endpoint follows by a limit of probabilities; it does not require evaluating the rate at $t = 1$. □

Learned rates and jumps. Replacing the posterior in Equation (20) by $q_{\theta,i}$ gives the learned rates $u_t^\theta(x^{i\to a}, x \mid S) = \lambda_\kappa(t) q_{\theta,i}(a \mid x, t, S)$. These rates define a valid generator, although exact recovery of the prescribed marginal path is no longer guaranteed. For a nonterminal state and $\lambda_\kappa(t) > 0$, its total exit rate and instantaneous jump probabilities are

$$\begin{aligned} \Lambda_\theta(t, x) &= \sum_{i\in\mathcal{M}(x)} \sum_a \lambda_\kappa(t) q_{\theta,i}(a \mid x, t, S) \\ &= |\mathcal{M}(x)|\lambda_\kappa(t) \\ Q_\theta(x^{i\to a} \mid x, t, S) &= q_{\theta,i}(a \mid x, t, S)/|\mathcal{M}(x)| \end{aligned} \tag{28}$$

Thus, nucleotide probabilities at each position differ from complete jump probabilities by the factor for selecting a masked position. The exact posterior's time independence also does not imply that a finite network explicitly conditioned on time satisfies the same property.

B.3 Cross-entropy and rate regularization

For $r \geq 0$ and $v > 0$, we use the scalar generalized KL divergence $d(r, v) = r\log(r/v) - r + v$, with $d(0, v) = v$.

Lemma B.2 (Categorical supervision and normalized rate matching). *Fix (x, t, S, i), and write $\pi_a = p_{1|t,i}(a \mid x, S)$ for its token posterior. Categorical cross-entropy equals $\mathcal{H}(\pi) + \mathrm{KL}(\pi\|q)$ and is minimized over categorical distributions at $q = \pi$. For any $\lambda > 0$, strictly positive categorical q, and native token $x_{1,i}$, normalized rates satisfy*

$$\sum_a d(\lambda\delta\{a, x_{1,i}\}, \lambda q_a) = -\lambda \log q_{x_{1,i}} \tag{29}$$

Proof. Using $0 \log 0 = 0$ and the extended-value convention when $\pi_a > 0 = q_a$,

$$\begin{aligned}\mathbb{E}_{a\sim\pi}[-\log q_a] &= -\sum_a \pi_a \log q_a \\ &= -\sum_a \pi_a \log \pi_a + \sum_a \pi_a(\log \pi_a - \log q_a) \\ &= \mathcal{H}(\pi) + \mathrm{KL}(\pi\|q) \geq \mathcal{H}(\pi)\end{aligned} \tag{30}$$

Equality holds precisely when

$$\mathbb{E}_{a\sim\pi}[-\log q_a] = \mathcal{H}(\pi) \quad \Longleftrightarrow \quad \mathrm{KL}(\pi\|q) = 0 \quad \Longleftrightarrow \quad q = \pi$$

For the rate identity, let $b = x_{1,i}$. Substituting $d(r, v) = r \log(r/v) - r + v$ and $d(0, v) = v$ gives

$$\begin{aligned}\sum_a d(\lambda\delta\{a, b\}, \lambda q_a) &= d(\lambda, \lambda q_b) + \sum_{a\neq b} d(0, \lambda q_a) \\ &= \lambda \log \frac{\lambda}{\lambda q_b} - \lambda + \lambda q_b + \sum_{a\neq b} \lambda q_a \\ &= -\lambda \log q_b - \lambda + \lambda \sum_a q_a \\ &= -\lambda \log q_b\end{aligned} \tag{31}$$

Consequently, conditional averaging over the native token also yields

$$\begin{aligned}\mathbb{E}_{b\sim\pi}\left[\sum_a d(\lambda\delta\{a, b\}, \lambda q_a)\right] &= -\lambda \sum_b \pi_b \log q_b \\ &= \lambda\mathcal{H}(\pi) + \lambda\,\mathrm{KL}(\pi\|q)\end{aligned} \tag{32}$$

Thus the normalized rate loss has the same categorical optimum. □

Softplus rate regularization. Equation (5) applies categorical supervision to sampled masked contexts, with normalization over each example's masked positions. A positive weighting determined by the conditioning context preserves the pointwise categorical optimum while changing the emphasis across contexts. This statement does not establish recovery of that optimum by a finite shared network.

For a fixed masked position i and context (x, t, S), omit the parameter and position indices and write $\pi_a = p_{1|t,i}(a \mid x, S)$. Our auxiliary rates are $\widetilde{r}_a = \zeta(\ell_a) + 10^{-6}$, whereas our categorical predictions are $q_a = \mathrm{softmax}_a(\ell)$. The conditional targets are $r^\star_a = c(t)\delta\{a, x_{1,i}\}$, with $c(t)$ as defined in Section 3.3. These rates need not sum to the prescribed hazard, so Lemma B.2 does not identify the auxiliary term with weighted CE. Differentiating its divergence at each position directly gives

$$\frac{\partial}{\partial \ell_a} \sum_b d(r^\star_b, \widetilde{r}_b) = \mathrm{sigmoid}(\ell_a)\left(1 - \frac{r^\star_a}{\widetilde{r}_a}\right) \tag{33}$$

This expression includes the additive rate floor through $\widetilde{r}_a$. For unrestricted nonnegative rates, the conditional expected optimum is $c(t)\pi_a$, using the continuous extension at zero. The softplus floor and shared logits restrict that solution, so the combined objective need not attain the categorical optimum. We therefore use the term auxiliary rate regularization and do not claim that the full training loss equals a likelihood bound.

B.4 Numerical conventions

The independent path assigns probability t^L to observing no masked positions. We redirect that event to a state with one uniformly selected mask, ensuring a nonzero supervised loss. Time is restricted to $[0, 0.9]$. The auxiliary predicted rates have a fixed additive floor of 10^{-6}, while the

target hazard uses a denominator offset of 10^{-3} and an upper cap of 20. The cap is inactive on this interval, but denominator regularization still changes the ideal target rate slightly. Proposition B.1 concerns the original probability path; it does not assert that these numerical modifications preserve that path exactly. No remasking corrector or learned change of the reveal schedule is enabled in this configuration.

C PRETRAINING AND MODEL SELECTION

C.1 PRETRAINING PROTOCOL

Training data and representations. The final augmented training manifest contains 9,875 records: 4,025 experimental structures and 5,850 predicted structures of Rfam sequences. Predicted structures are filtered against evaluation backbones before training; Appendix D details this filter and a sequence similarity audit. RNA-DFM w/o Rfam is trained exclusively on the 4,025 RNA-Solo training records. Every method in Table 1, including both RNA-DFM variants, NA-MPNN, RIDE and gRNAde, uses the same base-neutral three-atom representation and is trained from scratch with three random seeds. Each model receives P, $C4'$ and the sugar-derived pseudoatom defined in Equation (12), with the same residue filtering and coordinate construction. These records are the experimental training instances of the `das` split distributed with gRNAde (Joshi et al., 2025). All retrained baselines use this same experimental training manifest; only the augmented RNA-DFM receives the additional Rfam examples. The shared coordinate construction, model configurations, checkpoint identifiers and three-seed manifests specify the benchmark protocol.

Masked supervision. A single time t is sampled for each training example. Positions are retained independently with probability t and otherwise replaced with `M`. If every position is retained, one uniformly chosen position is masked. This safeguard against empty masks modifies the independent path in Equation (3) only on that event. Losses are averaged over masked positions within each example. The native labels remain categorical even when their associated training structure is predicted.

Auxiliary rate loss. The rate loss in Equation (6) uses the same masked states as CE, with weight $\eta = 0.05$. For the native-token target rate $r = c(t)$, its contribution at position i is

$$r \log \frac{r}{\widetilde{r}_{\theta,i,x_{1,i}}} - r + \sum_{a \in \mathcal{A}} \widetilde{r}_{\theta,i,a} \tag{34}$$

The rate cap is inactive for $t \leq 0.9$; sampling uses softmax probabilities rather than these auxiliary rates. Appendix B analyzes the relationship between categorical supervision and rate regularization.

Optimization. The augmented RNA-DFM configuration is trained from scratch for 100 epochs per seed using AdamW with the settings in Table 2. Training uses three distributed workers, each processing one graph per optimization step. Distributed gradient averaging therefore gives an effective batch of three graphs, apart from sampler padding. These workers participate in one training run; the three random seeds correspond to separate runs. All experimental and augmented examples use the same supervised objective. There is no loss weighted by confidence, coordinate jitter or supervised structural reward term in this configuration.

Let $E = 100$ be the total number of training epochs and $W = 5$ the warmup length. For zero-based epoch e, the learning rate is

$$\alpha_e = \begin{cases} \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})(e+1)/W, & 0 \leq e < W \\ \alpha_{\min} + \frac{\alpha_{\max} - \alpha_{\min}}{2}\left[1 + \cos\left(\pi \frac{e-W}{E-W-1}\right)\right], & W \leq e < E \end{cases} \tag{35}$$

where $\alpha_{\max} = 10^{-4}$ and $\alpha_{\min} = 10^{-6}$. The denominator $E - W - 1$ scales the cosine schedule from its peak at $e = W$ to its minimum at the final epoch $e = E - 1$. Training uses gradient clipping and runs to completion without effective early stopping.

Checkpoint selection. Checkpoint selection uses 54 examples from the designated clean validation split, comprising 23 recorded sequence cluster groups. Validation starts from fully masked

sequences and uses eight planned steps, greedy nucleotide choices and position selection based on confidence. The selection metric is recovery weighted by token counts across validation examples. A saved best checkpoint is replaced when recovery improves by more than 0.001 in absolute fraction, equivalent to 0.1 percentage points. Each training seed uses its own checkpoint selected on validation data; the checkpoint identifiers for each seed are recorded with the final evaluation artifacts. Random unmasking is the generation setting studied here; it was not used during checkpoint selection.

Training algorithm. Algorithm 1 summarizes one training run and is repeated for each of the three seeds. Each minibatch contains one graph per device; $\mathcal{L}_{\mathrm{pre}}(S, x_1, t, x; \theta)$ denotes the sampled loss in Equation (7). Optimizer settings and the validation procedure are specified above.

Algorithm 1 One run of RNA-DFM pretraining with Rfam augmentation

Require: Training set $\mathcal{D}$; validation set $\mathcal{D}_{\mathrm{val}}$; initial parameters θ.
Require: Epoch budget $E = 100$; devices $D = 3$; rate weight $\eta = 0.05$.
Ensure: Pretrained parameters θ_0 selected on validation data.
$v_{\mathrm{best}} \leftarrow -\infty$; initialize AdamW.
for $e = 0, \ldots, E-1$ **do**
  Set learning rate α_e using Equation (35); shuffle $\mathcal{D}$.
  for each distributed minibatch $\{(S^{(d)}, x_1^{(d)})\}_{d=1}^{D}$ **do**
    for all devices $d = 1, \ldots, D$ **in parallel do**
      Sample $t_d \sim \mathcal{U}[0, 0.9]$ and $x_{t_d}^{(d)} \sim p_{t_d|1}(\cdot \mid x_1^{(d)})$.
      If no positions are masked, mask one uniformly selected position.
      $\mathcal{L}_d \leftarrow \mathcal{L}_{\mathrm{pre}}(S^{(d)}, x_1^{(d)}, t_d, x_{t_d}^{(d)}; \theta)$.
    end for
    Update θ with AdamW on $D^{-1} \sum_d \mathcal{L}_d$, clipping the gradient norm at 1.
  end for
  $v_e \leftarrow \mathrm{Validate}(\theta, \mathcal{D}_{\mathrm{val}})$.
  if $v_e > v_{\mathrm{best}} + 0.001$ **then**
    $(\theta_0, v_{\mathrm{best}}) \leftarrow (\mathrm{copy}(\theta), v_e)$.
  end if
end for
return θ_0.

C.2 Metrics and candidate selection

For the main benchmark, we report TM-score, local distance difference test (lDDT), RMSD and native sequence recovery. Candidates are folded with Boltz for selection, and the selected sequences are folded with Protenix for structural evaluation. RNA TM-scores normalized by target length are computed using US-align (Zhang et al., 2022). lDDT measures agreement between local interresidue distances without requiring a global superposition (Mariani et al., 2013). RMSD is reported in Å, while TM-score, lDDT and recovery are reported on a scale from zero to one. For each target and seed, selection maximizes the TM-score of the Boltz prediction against the target backbone. Protenix scores do not enter selection; all reported metrics refer to the same selected sequence. This selector is deployable: the target backbone is the input to the design problem, and US-align scores are invariant to nucleotide identity, so the rule uses no information about the native sequence.

Sequence recovery. For seed r, backbone j and candidate n, write $x_1^{(r,j,n)}$ for the generated sequence. With valid positions I_j and native sequence $x_{1,\mathrm{nat}}^{(j)}$, recovery is

$$\mathrm{Recovery}_{r,j,n} = \frac{1}{|I_j|} \sum_{i \in I_j} \delta\Big\{ x_{1,i}^{(r,j,n)}, x_{1,\mathrm{nat},i}^{(j)} \Big\}$$

Table 1 reports recovery as a fraction; differences expressed in percentage points multiply the difference in fractions by 100.

TM-score and RMSD. For the main table, Boltz predictions determine selection and Protenix predictions supply the reported structural metrics against the target backbone. For US-align, the

predicted chain is the first structure and the target is the second, with RNA mode enabled. We use the TM-score normalized by target length, corresponding to the output labelled `Structure_2`. RMSD is computed from corresponding C4′ coordinates after Kabsch superposition and is expressed in Å. The residue mapping and treatment of incomplete structures are retained with the metric outputs.

lDDT. We compute lDDT from C4′ distances, using a 15 Å cutoff on reference distances and tolerances of 0.5, 1, 2 and 4 Å. For the evaluated residue pairs $\mathcal{P}_j = \{(a,b) : 0 < d_{ab}^{\mathrm{ref}} < 15\,\text{Å}\}$, the score is

$$\mathrm{lDDT} = \frac{1}{4|\mathcal{P}_j|} \sum_{(a,b)\in\mathcal{P}_j} \sum_{\epsilon\in\{0.5,1,2,4\}\,\text{Å}} \mathbf{1}\left[\left|d_{ab}^{\mathrm{pred}} - d_{ab}^{\mathrm{ref}}\right| < \epsilon\right]$$

where d_{ab}^{ref} and d_{ab}^{pred} are the target and predicted C4′ distances, and $\mathbf{1}[\cdot]$ is the indicator of the enclosed condition. This implementation restricted to the selected coordinates measures preservation of local distances without rigid alignment; it is not an all-atom stereochemical assessment.

Candidate budgets and seeds. Each of the five benchmark methods generates 16 candidates per target for each of its three training seeds, at temperature $T = 0.1$. Boltz TM-score selection is performed separately within each target–seed pool, and the selected scores are then averaged across the three seeds. The candidate budget is therefore 16 per seed; candidates are not pooled across seeds for selection. RNA-DFM uses random unmasking with $K = 8$ planned steps. Sampling parameters for each baseline and all random seeds are recorded alongside the generated sequences.

Candidate selection and scoring. Let $T_{r,j,n}^{\mathrm{B}}$ be the TM-score, normalized by target length, of candidate n predicted by Boltz for backbone j and seed r. Let $F_{r,j,n}^{\mathrm{P}}$ denote TM-score, lDDT or RMSD computed from the Protenix prediction of that candidate. Selection and structural reporting are

$$n_{r,j}^{*} = \arg\max_{1\le n\le 16} T_{r,j,n}^{\mathrm{B}}$$

$$B_{r,j}^{F} = F_{r,j,n_{r,j}^{*}}^{\mathrm{P}}$$

Recovery is computed from the selected nucleotide sequence, rather than from either structural predictor. Ties retain the first candidate in generation order. Neither Protenix scores nor native sequence recovery enter the selection rule, and seeds remain separate during selection. Using different predictors for selection and scoring does not imply their prediction errors are statistically independent.

Structure prediction. All compared methods use the same structural predictor settings; Protenix uses ten recycling iterations and 200 diffusion steps. Candidate coverage and missing-score handling for the selector and distillation analyses are specified in Appendices F.5 and G.

C.3 Rate regularization

Protocol. We compare CE-only training ($\eta = 0$) with CE + rate ($\eta = 0.05$) using the RNA-DFM configuration without Rfam augmentation. This separate objective comparison uses seed 0 for both models and the same 4,025 RNAsolo training records, geometric architecture and 130-epoch training schedule. Both models use an effective batch size of one graph per optimizer step, matching the CE + rate configuration. It does not use the three-seed aggregation of Table 1. The CE-only model is trained from scratch, while the CE + rate reference reuses its existing checkpoint. Checkpoints are selected using token-weighted recovery from fully masked validation sequences, yielding epochs 94 and 87 for CE-only and CE + rate, respectively.

Both models generate 16 candidates for each of the 88 clean test backbones, using eight sampling steps, temperature 0.1 and random position selection. We use matched sampling seeds and retain duplicate candidate entries within each pool. Boltz TM-score determines selection, and Protenix evaluates the selected candidates. Scores are averaged over conformations within each of 53 native sequences, then over sequences within each BGSU equivalence class, with equal weights across 22

classes. Paired 95% intervals invert two-sided wild cluster bootstrap-t tests using 20,000 draws of Webb weights over seven structural families.

Table 3: **CE-only comparison without Rfam augmentation, using training seed 0.** The upper block reports BGSU class-weighted scores; the lower block reports paired differences, defined as CE + rate minus CE-only.

Method	TM-score (↑)	lDDT (↑)	RMSD (Å, ↓)	Recovery (↑)
CE-only ($\eta = 0$)	0.4165	0.6212	9.8132	0.4989
CE + rate ($\eta = 0.05$)	0.4698	0.6558	9.1577	0.5341

Metric	Δ	95% CI	p
TM-score (↑)	$+0.0533$	$[-0.0431, +0.0819]$	0.2745
lDDT (↑)	$+0.0345$	$[-0.0752, +0.0664]$	0.6495
RMSD (Å, ↓)	-0.6555	$[-1.7470, +2.7920]$	0.9250
Recovery (↑)	$+0.0352$	$[-0.0045, +0.0845]$	0.0547

Negative RMSD differences and positive differences in the other metrics favor CE + rate. Recovery and its differences are reported as fractions.

Results and interpretation. CE + rate yields higher TM-score, lDDT and recovery, together with lower RMSD (Table 3). The paired differences are $+0.0533$ TM-score, $+0.0345$ lDDT, -0.6555 Å RMSD and $+3.52$ percentage points in recovery. All four paired intervals include zero, so these point estimates do not establish statistically significant improvements under the family-level inference protocol.

The intervals condition on seed 0 and exclude variation across training seeds. These results complement the analysis of the auxiliary objective in Appendix B.

D RFAM DATA AUGMENTATION

D.1 DATASET CONSTRUCTION

We supplement experimental backbones with predicted structures paired with their originating Rfam sequences (Ontiveros-Palacios et al., 2025; Wohlwend et al., 2024). Table 4 summarizes the recorded construction stages and dataset sizes. The final training manifest contains 5,850 predicted structures and 4,025 experimental records.

Sequence sources and eligibility. We use the Rfam 14.9 representative FASTA distributed after upstream clustering at 90% sequence identity and 80% coverage. The input contains 871,599 records, which are normalized to uppercase with thymine replaced by uracil. We restrict lengths to 40–300 nucleotides and reject sequences containing symbols outside A/G/C/U. The candidate builder excludes exact sequence matches to the experimental training set and Rfam sequences previously used in folding calibration. A shared set of previously encountered sequences also prevents duplicate sequence draws. These exclusions concern exact sequences and do not establish separation of sequence families from evaluation targets.

Length matching. The builder samples 30,000 desired lengths, with replacement, from experimental training records within the eligible length range, using random seed 7727. Eligible Rfam records are grouped by length, and each length bin is shuffled using the same random number generator. For each desired length, the builder takes the first unused sequence from the corresponding bin. If necessary, it searches successively wider offsets, considering the shorter bin before the longer bin at each offset. Offsets range from zero to 19 nucleotides, and a sequence is accepted only once. This procedure approximates the experimental length distribution while accommodating the available sequence pool. The candidate manifest stores sequence identifiers, original FASTA headers, nucleotide sequences, lengths and shard assignments.

Table 4: **Rfam augmentation pipeline.** Counts refer to sequence or structure records, rather than curated Rfam families. The pool passing the confidence threshold is counted using the rounded lower bound observed in the selected set. Selected membership is fixed by the augmentation manifest.

Stage	Records
Rfam 14.9 FASTA of representative sequences	871,599
Candidate sampling matched by length after eligibility checks	30,000
Available Boltz predictions for single chains	21,291
Predictions with confidence ≥ 0.5646	13,341
Manifest of predictions selected for coverage	6,380
Retained predictions after structural filtering	5,850
Combined predicted and experimental training records	9,875

Folding and confidence calibration. Boltz folding of the 30,000 candidates as individual RNA chains yields 21,291 predictions with confidence records. The workflow reads the model-0 structure and its associated `confidence_score`. A separate calibration folds 150 experimental training sequences and compares their predictions with experimental structures using US-align. Confidence and TM-score have Spearman correlation $\rho = 0.796$, supporting confidence as a quality indicator without guaranteeing accuracy for individual predictions. The selected set has a rounded lower confidence bound of 0.5646; 13,341 available predictions meet this bound (Table 4). Figure 1 displays this value rounded to three decimal places.

Selection provenance. The augmentation manifest selects 6,380 predictions to broaden sequence coverage within the available structure budget. The retained artifacts lack the complete selection script and clustering of the eligible pool, so membership is specified by this fixed manifest. The selected set does not establish one representative per sequence cluster or curated Rfam family.

Structural filtering. The recorded decontamination procedure removes a selected prediction if its maximum US-align TM-score against an evaluation backbone, normalized by target length, is at least 0.5. The normalization uses the evaluation backbone length, rather than the length of the predicted training structure. Writing $\mathcal{S}_{\text{eval}}$ for the reference evaluation backbones and $\widetilde{S}$ for a predicted training backbone, the retention criterion is

$$\max_{S \in \mathcal{S}_{\text{eval}}} \mathrm{TM}_{\mathrm{norm}\,|S|}\big(\widetilde{S}, S\big) < 0.5 \tag{36}$$

The final manifest excludes 530 of the 6,380 selected records, leaving 5,850 predicted structures for training. This structural screen is separate from screening for sequence similarity and cannot establish independence at the sequence level.

Sequence similarity audit. We compare all 5,850 Rfam sequences with the 95 unique sequences represented by the 159 original test backbones, including the 53 unique sequences in the 88-backbone clean test set. After converting sequences to uppercase and replacing thymine with uracil, we find no exact full-length matches and no cases in which one complete sequence is an exact substring of another.

For similarity screening, we compute local and global affine-gap alignments for all 555,750 sequence pairs using Biopython 1.82, without candidate prefiltering. The primary scoring scheme assigns $+2$ to matches, -1 to mismatches, -3 to gap opening and -0.5 to gap extension, retaining the first maximum-score alignment in the forward orientation. Identity is the fraction of alignment columns containing identical bases, including gap columns in the denominator; coverage is the aligned span divided by the full sequence length, computed separately for each sequence. Table 5 summarizes exact matches and similarities under the specified identity and coverage criteria.

At 80% local identity and 80% coverage on both sides, one Rfam sequence matches PDB 2OEU, chain A in the clean test set. This alignment has 33 identical bases across 40 columns (82.5% identity), covering 36 of 43 Rfam nucleotides (83.7%) and 38 of 42 target nucleotides (90.5%). A sensitivity analysis repeats all pairwise local alignments with match, mismatch, gap-opening and gap-extension scores of $+1$, -1, -2 and -0.5, respectively. All local matches in the clean test set

under either scoring scheme involve 2OEU, chain A. The audit therefore identifies local similarities whose threshold classification depends on alignment settings; absence of exact matches does not establish absence of homology. This audit leaves the training manifest and reported evaluations unchanged.

Table 5: **Sequence similarity audit of the 5,850 Rfam augmentation sequences.** Entries count distinct Rfam sequences with at least one match in each reference set.

Criterion	Original test	Clean test
Exact full-length match	0	0
Exact containment of a complete sequence	0	0
Primary alignment scoring		
Global identity $\geq 80\%$	0	0
Local identity $\geq 90\%$, coverage $\geq 80\%$ on both sides	0	0
Local identity $\geq 80\%$, coverage $\geq 80\%$ on both sides	1	1
Local identity $\geq 80\%$, coverage $\geq 80\%$ on the shorter sequence	36	10
Alternative alignment scoring		
Local identity $\geq 80\%$, coverage $\geq 80\%$ on both sides	0	0
Local identity $\geq 80\%$, coverage $\geq 80\%$ on the shorter sequence	23	3

The original and clean test sets contain 95 and 53 unique sequences, respectively. Scoring schemes and coverage definitions are specified in the text. Criteria overlap, so counts are not additive. All local matches in the clean test set involve PDB 2OEU, chain A.

Structure preprocessing. The structure parser reads the first predicted model and RNA chain, mapping recognized nucleotide names to A/G/C/U. The intermediate structure records store coordinates in 27 atom slots per nucleotide, with a Boolean mask distinguishing present atoms from missing entries. Model inputs are subsequently restricted to the same base-neutral three-atom representation used for experimental structures. A prediction is accepted only if its parsed nucleotide sequence exactly matches the source sequence. All 6,380 selected predictions pass this conversion; the final training manifest then restricts membership to the 5,850 retained records. Downstream filtering for valid atoms and backbone featurization follow Appendix A, identically for predicted and experimental examples. The Rfam sequence provides categorical supervision, while the predicted structure provides geometric conditioning. Probabilities from the folding model are not substituted for nucleotide labels, and confidence does not multiply the training loss.

Data mixture and manifests. The 4,025 experimental and 5,850 predicted records are sampled uniformly from their concatenation, without additional source weights. Both sources share atom filtering, graph construction, mask sampling and the training objective. The files `keep.json` and `train_synth.csv` identify the initial selection and converted structures, respectively. The final training membership is fixed by `train_mixed_decontam.csv`.

D.2 Augmentation comparisons

Table 6 compares the augmented and unaugmented models using the main benchmark protocol in Appendix F. The paired estimates favor augmentation on all four metrics, although their intervals include zero. The comparison concerns complete training configurations, including their data distributions and optimization budgets; it does not isolate data composition at a matched update budget.

E Random unmasking

Reveal counts and positions. At step $k \in \{0, \ldots, K-1\}$, let m_k be the number of masked positions. For the ideal independent masked path, the probability of revealing a masked position

between t_k and t_{k+1} is

$$\rho_k = \frac{t_{k+1} - t_k}{1 - t_k} = \frac{1}{K - k} \tag{37}$$

Independent categorical proposals in Equation (8) reveal each masked position with probability ρ_k. Their reveal count is binomial, and conditional on that count, every position subset of the corresponding size has equal probability. The implementation samples this count and subset directly, then enforces at least one reveal whenever $m_k > 0$. Specifically, it draws $B_k \sim \mathrm{Binomial}(m_k, \rho_k)$ and sets

$$\begin{aligned} n_k &= \min\{m_k, \max(1, B_k)\} \\ J_k &\sim \mathrm{Uniform}\{J \subseteq \mathcal{M}(x_{t_k}) : |J| = n_k\} \end{aligned} \tag{38}$$

At the final step, all remaining positions are selected. The requirement of at least one update excludes empty updates and therefore changes the ideal discretization of the independent masking path.

For $m > 0$ and $\rho = \rho_k$, the probabilities of update counts are

$$\begin{aligned} p(n = 1) &= (1 - \rho)^m + m\rho(1 - \rho)^{m-1} \\ p(n = j) &= \binom{m}{j} \rho^j (1 - \rho)^{m-j}, \qquad 2 \le j \le m \end{aligned} \tag{39}$$

Conditional on n, every subset of that size has probability $\binom{m}{n}^{-1}$. Nucleotide choices within the selected subset use the same model predictions computed before the update. Positions committed together cannot condition on each other's newly sampled identities during that step.

Nucleotide sampling. Given logits $\ell_{\theta,i}$ computed from (x_{t_k}, t_k, S), temperature scaling and state updates are defined by

$$\begin{aligned} q^{(T)}_{\theta,i}(\cdot \mid x_{t_k}, t_k, S) &= \mathrm{softmax}(\ell_{\theta,i}/T) \\ a_i &\sim q^{(T)}_{\theta,i}(\cdot \mid x_{t_k}, t_k, S), \qquad i \in J_k \\ x_{t_{k+1},i} &= \begin{cases} a_i, & i \in J_k \\ x_{t_k,i}, & i \notin J_k \end{cases} \end{aligned} \tag{40}$$

The temperature $T > 0$ rescales nucleotide logits without changing the uniform distribution over selected position subsets.

Transition kernel. Let $\pi_k(n \mid m)$ denote the probability of an update count in Equation (39). For an active state x, let $m = |\mathcal{M}(x)| > 0$. For a valid transition $x \to x'$ that reveals exactly the set J and preserves every other token, the step kernel is

$$\mathcal{K}_{\theta,k}(x' \mid x, S) = \frac{\pi_k(|J| \mid m)}{\binom{m}{|J|}} \prod_{i \in J} q^{(T)}_{\theta,i}(x'_i \mid x, t_k, S) \tag{41}$$

Invalid transitions have probability zero, and a completed sequence is treated as an absorbing state. Summing this kernel over nucleotide assignments, subsets and update counts gives one. For $m > 0$, the count rule increases the expected number of updates relative to independent Bernoulli reveals:

$$\mathbb{E}[n_k \mid m_k = m] = m\rho_k + (1 - \rho_k)^m \tag{42}$$

The additional term is exactly the probability of an empty Bernoulli update, redirected into an update at a single position. It follows that revealed fraction and network conditioning time need not track each other during generation. The same distribution of update counts applies on the remaining grid when generation begins at a nonzero step index.

Proposition E.1 (Law and termination of random unmasking)**.** *For $m \ge 1$, $0 < \rho \le 1$, and normalized token probabilities, Equation* (38) *defines the count law in Equation* (39)*. Its transition kernel is Equation* (41)*, which is normalized, and its expected update count is Equation* (42)*. Starting with m_0 masks, the sampler completes the sequence in at most $\min(K, m_0)$ updating passes.*

Proof. For an active state, $n = \max(1, B)$ with $B \sim \mathrm{Binomial}(m, \rho)$. Combining the events $B = 0$ and $B = 1$ gives Equation (39); all other count probabilities remain unchanged. Conditional on n, a uniform permutation selects each subset J with probability

$$p(J \mid n, x) = \frac{n!(m-n)!}{m!} = \binom{m}{n}^{-1} \tag{43}$$

Multiplying by the independent nucleotide probabilities gives Equation (41). Summing over nucleotide assignments and then over subsets yields

$$\begin{aligned} \sum_{x'} \mathcal{K}_{\theta,k}(x' \mid x, S) &= \sum_{n=1}^{m} \frac{\pi_k(n \mid m)}{\binom{m}{n}} \sum_{\substack{J \subseteq \mathcal{M}(x) \\ |J|=n}} \prod_{i \in J} \sum_{a_i \in \mathcal{A}} q_{\theta,i}^{(T)}(a_i \mid x, t_k, S) \\ &= \sum_{n=1}^{m} \pi_k(n \mid m) = 1 \end{aligned} \tag{44}$$

The absorbing kernel for a completed state is also normalized. Since $n = B + \delta\{B, 0\}$, the expected reveal count is

$$\mathbb{E}[n \mid m] = \mathbb{E}[B \mid m] + p(B = 0) = m\rho + (1-\rho)^m \tag{45}$$

Each active pass removes at least one mask, while the final scheduled pass removes every remaining mask:

$$\begin{aligned} 1 \le n_k \le m_k \quad &\Longrightarrow \quad 0 \le m_{k+1} \le m_k - 1 \\ \rho_{K-1} = 1 \quad &\Longrightarrow \quad m_K = 0 \quad \text{almost surely} \\ N := \min\{k : m_k = 0\} &\le \min(K, m_0) \quad \text{almost surely} \end{aligned} \tag{46}$$

Completed trajectories are padded by absorbing steps, with $N = 0$ when $m_0 = 0$. □

The result establishes a well-defined sampler with a finite number of steps, not preservation of the ideal path marginals from Proposition B.1. In particular, forcing progress changes the expected update count by the explicitly positive term $(1-\rho)^m$ whenever $\rho < 1$.

Time conditioning and termination. The clock indexed by sampling step supplies $t_k = k/K$ to the network. This clock is not capped at the maximum training time. The largest scheduled input time is $t_{K-1} = 1 - 1/K$, which exceeds 0.9 only when $K > 10$. For the reported setting $K = 8$, all network inputs satisfy $t_k \le 7/8 < 0.9$. The sampler commits at least one nucleotide at every active step and requires at most $\min(K, L)$ updating passes. The implementation can perform an additional forward pass to check termination when it finishes before the final scheduled step.

Sampling algorithm. Algorithm 2 generates a sequence from the fully masked state $x_0 = \mathtt{M}^L$. At each step, a random permutation of the masked positions determines the subset to reveal. Nucleotide identities are sampled independently from the current predictions and committed simultaneously. At $k = K-1$, the reveal probability equals one, so every remaining mask is filled without an additional branch.

Sampling settings. Our sampler with random unmasking uses $K = 8$, $T = 0.1$ and 16 sequences per target and seed. The main benchmark averages three independently trained seeds for every method, as described in Appendix F. It loads the checkpoint selected on validation data described in Appendix C.

F Evaluation protocols and analyses

This section specifies the evaluation population, aggregation and statistical inference for the main benchmark and supplementary analyses. Metric definitions and candidate selection appear in Appendix C.2; augmentation comparisons appear in Appendix D.2.

Algorithm 2 $\mathrm{RandomUnmask}(\vartheta, S, K, T)$

Require: Fixed model parameters ϑ; backbone S with L nucleotides.
Require: Step budget $K \geq 1$; temperature $T > 0$.
Ensure: Completed sequence x_1.
$x \leftarrow \mathtt{M}^L$; disable model dropout.
for $k = 0, \ldots, K-1$ **do**
  $t_k \leftarrow k/K$; $q_i \leftarrow q^{(T)}_{\vartheta,i}(\cdot \mid x, t_k, S)$ for all i.
  $\mathcal{M} \leftarrow \{i : x_i = \mathtt{M}\}$; $m \leftarrow |\mathcal{M}|$.
  if $m = 0$ **then**
    break.
  end if
  $B_k \sim \mathrm{Binomial}(m, 1/(K-k))$; $n_k \leftarrow \min\{m, \max(1, B_k)\}$.
  $\sigma_k \sim \mathrm{UniformPerm}(\mathcal{M})$; $J_k \leftarrow \{\sigma_{k,1}, \ldots, \sigma_{k,n_k}\}$.
  $a_i \sim \mathrm{Categorical}(q_i)$ independently for all $i \in J_k$.
  $x_i \leftarrow a_i$ simultaneously for all $i \in J_k$.
end for
return x.

F.1 Datasets and evaluation units

Split and test filtering. The original das split assigns individual sequences to subsets, without enforcing separation of entire sequence clusters. Its test set contains 159 backbone records, of which 71 belong to 11 sequence clusters also represented in the experimental training set. These cluster assignments use the existing labels for 80% sequence identity, recorded as `cluster_seqid0.8`. We exclude every test record whose cluster identifier appears in the experimental training manifest. The remaining 88 records form `test_clean`, covering 53 unique native sequences and 15 sequence clusters. Thus, separation of clusters at 80% sequence identity results from filtering after splitting, rather than the original split construction. Sun et al. (2025) also report overlap between their evaluation set and the training sets of RiboDiffusion and RDesign, with 100% sequence recovery on examples such as 7D7W. This check concerns overlap with experimental training records; the Rfam augmentation set undergoes the separate procedures described in Appendix D.

Evaluation hierarchy. The evaluation set contains 88 backbone records corresponding to 53 unique native sequences, grouped into 15 sequence clusters at an identity threshold of 0.8. Multiple conformations are averaged within each native sequence, followed by sequence averaging within BGSU equivalence classes. The 22 equivalence classes receive equal weight in the point estimates in the main table. The sequence clusters at 80% identity define the criterion for cleaning the test set, whereas the seven structural families define the units for statistical inference. We use the existing `cluster_structsim0.45` labels distributed with gRNAde (Joshi et al., 2025). Its preprocessing clusters the first recorded structure of each unique sequence using qTMclust with a TM-score cutoff of 0.45 (Zhang et al., 2022). The upstream call retains qTMclust's defaults of C3′ coordinates for RNA and TM-score normalization by the longer of the two structures. We inherit these labels without reclustering the test set; this grouping is separate from the target-length-normalized 0.5 cutoff used to filter Rfam predictions. Neither the structural families nor the BGSU equivalence classes define the original das split.

Each method is evaluated on the same backbone records; matching sequence identifiers alone does not guarantee matching input conformations. Recovery uses the native sequence restricted to the generated positions, preserving the residue mapping established during preprocessing. Evaluation structures and sequence labels must use the same mapping to avoid shifts caused by missing backbone atoms.

F.2 Aggregation and statistical inference

Aggregation and uncertainty. For every method, we average across its three training seeds and the conformations of each native sequence, then give equal weight to each of the 22 BGSU nonredundant equivalence classes represented in the evaluation set (Leontis & Zirbel, 2012). Equivalence classes are curated externally and are not derived from any threshold chosen here; each class receives weight $1/22$, approximately 4.55%. Confidence intervals use structural families as the re-

sampling units. The 53 unique sequences nest exactly within 15 sequence clusters at 80% identity, which in turn nest exactly within the seven inherited structural families, and equivalence classes nest within families without crossing. Assigning a shared bootstrap weight to each family allows for dependence among its constituent sequences and equivalence classes. With only $G = 7$ families, inference is sensitive to the bootstrap procedure. We use the wild cluster bootstrap-t with six-point Webb weights, drawing on methods developed for inference with few clusters (Cameron et al., 2008; Webb, 2023). Intervals are obtained by inverting that test, so an interval excludes zero exactly when the test rejects at the 5% level.

BGSU class weighting. Let $\mathcal{J}_u$ contain the backbone records for native sequence u, and let $\mathcal{U}_e$ contain the sequences in equivalence class e. With $R = 3$ training seeds for every method and $E = 22$ classes, the scores for each sequence and estimate for the main table are

$$A_u^F = \frac{1}{|\mathcal{J}_u|} \sum_{j \in \mathcal{J}_u} \frac{1}{R} \sum_{r=1}^{R} B_{r,j}^F$$

$$\widehat{F} = \frac{1}{E} \sum_{e=1}^{E} \frac{1}{|\mathcal{U}_e|} \sum_{u \in \mathcal{U}_e} A_u^F$$

This aggregation gives each equivalence class equal weight after averaging seeds, conformations and sequences within the class. The same hierarchy applies to every model and metric in the main table.

Wild cluster bootstrap. The 22 equivalence classes nest within seven structural families. For each metric, class means are averaged within families, with family weights proportional to their numbers of equivalence classes. This produces the same point estimate as equal weighting of the 22 classes. Confidence intervals invert a wild cluster bootstrap-t test using 20,000 draws of Webb weights with six support points at the family level. Scores from the three training seeds are averaged before inference; the intervals condition on those runs and do not incorporate training-seed variability. Paired comparisons apply the same procedure to differences between models on the same target, preserving the shared evaluation targets. Marginal intervals are not intervals for model differences, and their overlap does not determine paired significance.

F.3 Paired model comparisons

Table 6 reports paired differences between the augmented RNA-DFM and each comparator under the main evaluation protocol, using three training seeds for every method. Write $A_{u,m}^F$ for the sequence-level score A_u^F of method m, as defined in Appendix F.2. For comparator m, we form matched differences for each sequence and aggregate them with the same equal weights for classes:

$$\widehat{\Delta}_m^F = \frac{1}{22} \sum_{e=1}^{22} \frac{1}{|\mathcal{U}_e|} \sum_{u \in \mathcal{U}_e} \left(A_{u,\text{RNA-DFM}}^F - A_{u,m}^F \right)$$

The paired interval is obtained by wild cluster bootstrap-t test inversion with Webb weights over structural families. Positive differences favor RNA-DFM for TM-score, lDDT and recovery; negative differences favor it for RMSD. Intervals crossing zero do not establish a directional difference or equivalence between models.

The paired comparisons favor RNA-DFM over RIDE on lDDT and native recovery. The remaining fourteen intervals cross zero, including every comparison with NA-MPNN, gRNAde and the unaugmented variant. These unresolved differences do not establish equivalence or noninferiority.

Table 6: **Paired differences relative to RNA-DFM.** Differences are computed as RNA-DFM minus comparator under the evaluation protocol of Table 1.

Method	Metric	Δ	95% CI	p
RNA-DFM w/o Rfam	TM-score (↑)	$+0.006$	$[-0.022, +0.077]$	0.763
	lDDT (↑)	$+0.008$	$[-0.025, +0.054]$	0.332
	RMSD (Å, ↓)	-1.161	$[-2.728, +0.307]$	0.063
	Recovery (↑)	$+0.023$	$[-0.082, +0.044]$	0.583
NA-MPNN	TM-score (↑)	$+0.004$	$[-0.104, +0.095]$	0.829
	lDDT (↑)	-0.018	$[-0.131, +0.079]$	0.461
	RMSD (Å, ↓)	-0.358	$[-1.394, +3.703]$	0.785
	Recovery (↑)	$+0.037$	$[-0.074, +0.073]$	0.660
RIDE	TM-score (↑)	$+0.051$	$[-0.002, +0.062]$	0.055
	lDDT (↑)	$+0.048$	$[+0.030, +0.064]$	**0.003**
	RMSD (Å, ↓)	-1.922	$[-4.689, +0.261]$	0.063
	Recovery (↑)	$+0.115$	$[+0.018, +0.212]$	**0.028**
gRNAde	TM-score (↑)	$+0.038$	$[-0.014, +0.102]$	0.206
	lDDT (↑)	$+0.007$	$[-0.032, +0.073]$	0.541
	RMSD (Å, ↓)	-0.908	$[-1.767, +1.594]$	0.346
	Recovery (↑)	$+0.102$	$[-0.024, +0.233]$	0.133

Intervals are paired 95% wild cluster bootstrap-t confidence intervals over seven structural families; p-values are two-sided. Bold p-values indicate $p < 0.05$. Positive differences favor RNA-DFM for TM-score, lDDT and recovery; negative differences favor it for RMSD. Recovery differences are fractions, and all differences are computed before rounding.

F.4 Recovery and structural agreement

Native nucleotide replacement (Figure 2A). Figure 2A uses a separate perturbation dataset of 102 design records spanning 20 sequence clusters at 80% identity. This perturbation analysis uses a different evaluation population from the 88-backbone benchmark in Table 1 and panels B–C. For each original RNA-DFM design r, let E_r contain positions paired in the assignment derived from the native structure and mismatched to the native sequence. For the six correction fractions $d = 0, 0.2, \ldots, 1$, a nested subset of $\mathrm{round}(d|E_r|)$ positions is replaced by the native bases; all other positions remain unchanged. The denominator is therefore the number of initially mismatched paired nucleotides. Replacement acts on individual positions and may leave a base pair partially corrected. The 0% stage is the original design, whereas 100% corrects all eligible positions without necessarily recovering the full native sequence. Two of the 102 records have no eligible positions and remain unchanged at every dose.

Base pair F1 and uncertainty. Each corrected sequence and its native sequence are folded with ViennaRNA (Lorenz et al., 2011). Base pair F1 compares their predicted minimum free energy pair sets; these are thermodynamic predictions rather than experimentally determined structures. For each record we compute the paired change $\Delta F1_r(d) = F1_r(d) - F1_r(0)$. Records are averaged within each of 20 sequence clusters at 80% identity, and the cluster means receive equal weight. Each of 20,000 bootstrap replicates resamples 20 clusters with replacement, retaining all records and the pairing of each dose with the original design. The 2.5th and 97.5th percentiles of the replicate means define the reported confidence intervals. The difference at zero dose is identically zero. Recovery is computed from these same corrected sequences: at 80% correction, its mean increases from 0.5155 to 0.7608 ($\Delta = +0.2453$), while mean F1 changes by -0.2743. All four intervals for F1 changes at intermediate doses exclude zero on the negative side.

Recovery distributions (Figure 2B). For each target and seed, we select the candidate with the highest Boltz TM-score normalized by target length among 16 generated sequences, following the rule in the main table. Native recovery is measured for that selected candidate, without selecting again by recovery. For each method, we average across its three training seeds and conformations

Table 7: **Selection gains behind Figure 2C.** Gains are relative to each method's mean Protenix TM-score over distinct candidates with valid scores from 16 generated samples. Pool means appear in parentheses; the difference row reports TM-score selection minus recovery selection.

Method	Selection rule	Δ	95% CI	p
RNA-DFM (0.4073)	TM-score	$+0.0380$	$[-0.0102, +0.0598]$	0.288
	Recovery	$+0.0186$	$[+0.0067, +0.0294]$	**0.009**
	Difference	$+0.0194$	$[-0.0390, +0.0427]$	0.822
RNA-DFM w/o Rfam (0.4190)	TM-score	$+0.0200$	$[-0.0379, +0.0357]$	0.642
	Recovery	$+0.0320$	$[-0.0262, +0.0432]$	0.175
	Difference	-0.0120	$[-0.0334, +0.0109]$	0.096
NA-MPNN (0.3758)	TM-score	$+0.0654$	$[-0.0054, +0.1432]$	0.066
	Recovery	-0.0031	$[-0.0401, +0.0363]$	0.895
	Difference	$+0.0685$	$[-0.0343, +0.1822]$	0.154
RIDE (0.3760)	TM-score	$+0.0183$	$[+0.0037, +0.0283]$	**0.029**
	Recovery	$+0.0077$	$[-0.0076, +0.0177]$	0.173
	Difference	$+0.0106$	$[-0.0001, +0.0216]$	0.051
gRNAde (0.4060)	TM-score	$+0.0015$	$[-0.0270, +0.0314]$	0.838
	Recovery	$+0.0018$	$[-0.0324, +0.0363]$	0.782
	Difference	-0.0003	$[-0.0621, +0.0556]$	0.984

TM-score selection uses the Boltz prediction against the target backbone and is deployable; selection by recovery requires the native sequence and serves only as a retrospective reference. Intervals are paired 95% wild cluster bootstrap-t confidence intervals over seven structural families; p-values are two-sided and bold indicates $p < 0.05$.

of each native sequence, then across sequences within each BGSU equivalence class. Each model contributes 22 equivalence class observations. Boxes span the 25th–75th percentiles, lines mark medians, and whiskers reach the most extreme observations within 1.5 times the interquartile range. All observations, including outliers, are plotted; numbers beside the boxes give means. The boxes describe variation across equivalence classes, rather than uncertainty in the mean.

F.5 Candidate selection

TM-score gains (Figure 2C). For each backbone and training seed, we deduplicate the 16 candidates by nucleotide sequence, retaining the first occurrence. Entries without the required scores are excluded, and both selection rules use the same retained candidate set. TM-score selection ranks Boltz predictions against the target; recovery selection ranks candidates by native sequence recovery. Ties retain the first candidate in generation order, and Protenix evaluates both selected sequences. For each method m, the plotted gain is

$$\Delta\mathrm{TM}_m = \mathrm{TM}_{m,\mathrm{selected}} - \mathrm{TM}_{m,\mathrm{pool}}$$

where the pool mean gives equal weight to distinct retained sequences within the same backbone and seed. Repeated samples receive no additional weight, and the pool denominator may be smaller than 16. Gains follow the main benchmark aggregation over three seeds, conformations and 22 BGSU classes. Paired 95% intervals use the family-level wild cluster bootstrap in Appendix F.2, holding candidate pools and training seeds fixed. Each interval compares a selector with its own model's pool mean; it does not compare the two selectors. Protenix scores enter neither ranking, so Boltz selection need not maximize the plotted score.

Selector comparison. Table 7 reports gains and paired differences between the selectors for all five methods. TM-score selection has positive point estimates throughout, but only RIDE's interval excludes zero; recovery selection is resolved only for RNA-DFM. No paired interval for the difference between selectors excludes zero, so these results establish neither dominance nor equivalence.

F.6 SAMPLING LATENCY AND STRUCTURAL QUALITY

Paired latency measurements. Figure 3A times the same DFM and order-agnostic AR checkpoints used for panel B, both trained with seed 0 and selected at epochs 87 and 151, respectively. The models share their geometric network dimensions and experimental training data but retain their respective generative objectives and conditioning. Both models reside on the same NVIDIA H20-3e GPU and are measured consecutively for each backbone, with their measurement order alternating across pairs. Each measurement averages eight sequentially generated sequences at temperature 0.1 and random position order, using eight steps for DFM and one decision per nucleotide for AR. CUDA synchronization brackets the timed loop; prebuilt graphs reside on the GPU, and timing includes cloning the graph input but excludes graph construction, structure prediction and selection. We use 32 backbones of 27–210 nucleotides spanning eight length bins, randomizing their order for each of three timing repetitions. These are repeated measurements of the fixed checkpoints, rather than three independently trained seeds.

A fixed DFM probe is measured before and after each pair. After warm-up, 30 calibration probes determine a threshold of 1.10 times their tenth-percentile latency (168.69 ms). We retain pairs whose mean probe latency does not exceed this threshold, yielding 79 of 96 measurements over 32 backbones. This filter reduces measured load variation on the shared device without establishing exclusive GPU access. Points in panel A are within-backbone medians of retained measurements; lines are ordinary linear fits in milliseconds. The median DFM latency is 115 ms; the fitted AR latency increases by 19.14 ms per nucleotide.

Latency fits and budgets. Let $c_m(L)$ denote the fitted time in seconds to generate one candidate sequence of length L with method $m \in \{\mathrm{DFM}, \mathrm{AR}\}$, where L counts valid nucleotide positions in the input graph. We fit a separate linear model by ordinary least squares to each method's median per-sequence latencies on the 32 measured backbones in Figure 3A. These measurements use $K = 8$ unmasking steps for DFM and L sequential nucleotide decisions for AR. The resulting fits are

$$\begin{aligned} c_{\mathrm{DFM}}(L) &\approx 1.31 \times 10^{-6} L + 0.115 \\ c_{\mathrm{AR}}(L) &\approx 0.0191 L - 0.0122 \end{aligned} \tag{47}$$

The slopes are expressed in seconds per nucleotide and the intercepts in seconds. Coefficients are rounded to three significant figures for presentation; all sampling budgets use the fitted coefficients at full precision. For Figure 3B, we evaluate these fits at the length L_r of each of the 88 test backbones and average with equal weight per backbone. Because candidates are generated sequentially, the sampling time for N candidates is N times the mean per-sequence cost:

$$\begin{aligned} \bar{c}_m &= \frac{1}{88} \sum_{r=1}^{88} c_m(L_r) \\ t_m(N) &= N \bar{c}_m \end{aligned} \tag{48}$$

where $\bar{c}_m$ is the mean sampling time per candidate and $t_m(N)$ is the time coordinate plotted for candidate count N. This gives $\bar{c}_{\mathrm{DFM}} = 0.11475\,\mathrm{s}$ and $\bar{c}_{\mathrm{AR}} = 1.47080\,\mathrm{s}$; at $N = 16$, the corresponding times are $1.84\,\mathrm{s}$ and $23.53\,\mathrm{s}$. The mean-cost budget does not imply identical affordable candidate counts for every individual target. Panel B combines this fitted timing model with structural scores from the same checkpoints and sampling settings; it does not measure end-to-end design time.

Quality across candidate budgets. Both methods generate 16 candidates for each of the 88 clean test backbones, with temperature 0.1, random order and the same step settings used for timing. DFM candidates use the eight-step pools also generated for the objective comparison; the AR evaluation reuses its existing complete candidate pools. For each target and $N \in \{1, 2, 3, 4, 6, 8, 11, 16\}$, we compute the exact expected maximum Boltz TM-score over uniformly selected subsets using finite-pool order-statistic weights. Candidate multiplicities are retained, so N counts sampled entries and all 88 targets contribute at every candidate count. Scores are averaged over conformations within each native sequence, then equally over the 53 native sequences. Thus, this comparison uses Boltz for both selection and evaluation and weights native sequences equally, whereas the main benchmark evaluates Boltz-selected candidates with Protenix and weights BGSU equivalence classes equally.

Intervals in the source data use 10,000 percentile bootstrap draws over 15 sequence clusters at 80% identity for the paired gain over $N = 1$; uncertainty shading is omitted from the figure.

With one candidate, mean TM-scores are 0.416 for DFM and 0.401 for AR. At $N = 16$, mean TM-scores are 0.5196 for DFM and 0.5033 for AR, with fitted sampling times of 1.83596 s and 23.53282 s. At this candidate count, DFM improves TM-score by 0.016 with a $12.8\times$ sampling speedup. At the DFM $N = 16$ budget, the largest affordable listed AR candidate count is $N = 1$, giving a DFM-minus-AR difference of $+0.1191$. The two checkpoints were selected by validation recovery at different training epochs; this is a comparison of complete trained methods rather than an isolated causal effect of the sampling rule. DFM uses its masked denoising and rate objectives with time conditioning, whereas the autoregressive model uses its autoregressive objective and conditioning.

G BEST-OF-N DISTILLATION

Our procedure applies the supervised distillation idea discussed by Sessa et al. (2025). BOND derives supervised fitting of best-of-N samples from forward KL minimization and combines forward and reverse KL terms in its Jeffreys objective. We optimize the masked denoising and auxiliary rate losses on selected sequences and native labels, with the selected labels fixed throughout finetuning. This objective is a supervised adaptation for RNA inverse folding, not an implementation of J-BOND or an exact sequence-level KL objective.

G.1 SUPERVISION AND FINETUNING

Distillation target selection. The supervision pools are generated by the same decontaminated checkpoint that initializes fine-tuning, so the teacher and the student differ only in the sampling temperature used to draw the pools. The source pools use random position order, temperature 0.5 and eight sampling steps, matching the deployment schedule in every respect except temperature. Appendix G.2 describes the sequence cluster split used to select temperature 0.5 on partition A and evaluate it on partition B. For each backbone, only candidates with the correct sequence length and two available Boltz scores are considered, and at least eight scored candidates are required. Native sequence recovery is computed against the residue sequence extracted from the corresponding backbone graph. The eligible subset contains candidates at or below the median recovery of the full scored pool. We select its highest scoring candidate using Equation (9). No margin threshold is applied: every backbone whose pool passes the eligibility filters contributes one label. We avoid thresholding on the selected score because the margin between that score and the pool mean is computed from the same two structure predictions that define the score, so a threshold preferentially retains backbones whose apparent margin is largest by prediction noise; a split-half estimate that selects on one prediction and reads the other recovers roughly half of the apparent margin. Removing the filter also increases the number of supervised backbones, which raises how far fine-tuning moves the model away from its initialization. The recovery restriction controls the contribution of native sequence similarity to the training labels and is omitted during deployment. Thus, the supervision set incorporates structural ranking under a training filter rather than reproducing the deployment selection distribution exactly. The resulting set contains 159 backbones and has no record or 80% sequence identity cluster overlap with the clean test set.

Control and fine-tuning. For each retained backbone, the control samples an alternative uniformly from the eligible candidate entries after excluding the selected entry, with selection seed 20260926. Individual sequences are not matched in recovery between the two arms. The same 159 native pairs are added to each arm, yielding 318 equally weighted training examples per arm. All sequence labels use the masked cross-entropy and rate objective with $\eta = 0.05$. Both arms are fine-tuned for 12 epochs with batch size 1, peak learning rate 10^{-5}, one warm-up epoch, cosine decay to 10^{-6}, weight decay 5×10^{-4} and gradient clipping at norm 1. The final checkpoint is used, without selection on test performance. Consistent with Section 4.3 and Figure 3C–D, the reported distillation analysis uses finetuning seed 0.

Table 8: **Sequence cluster split for supervision temperature selection.** Partitions share no sequence clusters at 80% identity. Records denote backbone conformations, and sequences denote unique native sequences.

Partition	Clusters	Records	Sequences
A (temperature selection)	8	43	24
B (evaluation)	7	45	29

G.2 Temperature of the supervision pool

To separate temperature selection from evaluation, we divide the 88 clean backbones into two disjoint partitions using the 15 sequence clusters at 80% identity. All records from the same sequence cluster are assigned to the same partition. Partition A contains eight clusters, 43 backbone records and 24 unique native sequences; partition B contains seven clusters, 45 records and 29 unique native sequences (Table 8). Partition A is used to select the supervision temperature, and partition B is used to evaluate the selected setting.

We compare supervision temperatures $T \in \{0.1, 0.5, 1.0\}$ using pools drawn from the same pretrained checkpoint, with random position order, eight sampling steps and 16 candidates per training backbone. For each pool, a label is drawn uniformly from the eligible subset, separating the temperature comparison from structural ranking during label construction. Temperature selection uses the expected Protenix TM-score after Boltz selection from eight candidates, with equal weight assigned to the unique native sequences in partition A. This procedure selects $T = 0.5$, which is fixed for evaluation on partition B in Figure 3C–D and Table 9. The sweep uses finetuning seed 0; evaluation uses eight sampling steps and temperature 0.1.

G.3 Evaluation protocol

Independent evaluator. The pretrained reference and both finetuned models use eight random unmasking steps for evaluation. Every generated candidate from both arms and from the pretrained reference is additionally folded with Protenix using ten recycling iterations and 200 diffusion steps, matching the settings of our main structural table, and scored with the same length normalization. Coverage is complete for all three arms. Selection continues to use the Boltz score, so the evaluator does not participate in ranking. Expected scores after selection from N candidates use the exact order statistic weights $\binom{M-i}{N-1}/\binom{M}{N}$ for the ith ranked candidate in a pool of M scored candidates, which avoids resampling noise. The evaluated curves retain candidate multiplicities, so N counts candidate entries rather than distinct folding calls. Coverage and the finite-pool expectation are specified below.

Coverage and selection statistics. Each finetuned model generates 16 candidates for each of the 45 backbones in partition B using eight random unmasking steps and temperature 0.1. The current analysis retains candidate multiplicities and excludes entries without a cached Boltz score. Consequently, the number of scored candidates M_S can be less than 16. All 45 backbones in partition B have at least eight scored candidates in both arms. We report $N \leq 8$ on this common set to keep the evaluation population fixed. Candidate count here refers to sampled entries, not the number of unique sequences or distinct folding calls.

For a given backbone, order its scored candidate entries by decreasing Boltz TM-score. Let $v_{S,i}$ be the evaluation score of the entry at rank i. The expected score after selection from a uniformly sampled subset of N entries is

$$Q_N(S) = \sum_{i=1}^{M_S-N+1} \frac{\binom{M_S-i}{N-1}}{\binom{M_S}{N}} v_{S,i} \tag{49}$$

The weight follows by counting subsets that contain entry i, exclude every higher ranked entry and contain $N-1$ lower ranked entries. This computes the finite-pool expectation exactly and introduces no additional subset-sampling error. Table 9 reports the arithmetic mean over the 45

Table 9: **Distillation results on partition B, finetuning seed 0.** Entries are mean TM-scores over 45 held-out backbones (29 unique sequences in seven sequence clusters) after Boltz selection from N candidates. Differences are distillation minus the random-label control, with paired standard errors over backbones. Boltz evaluation uses the selection predictor; Protenix evaluates independently of selection. Figure 3C–D weights the 29 unique sequences equally.

Method	$N = 1$	$N = 2$	$N = 4$	$N = 8$
Boltz selection, Boltz evaluation				
Random-label control	0.4035	0.4424	0.4735	0.4991
Best-of-N distillation	0.4110	0.4509	0.4827	0.5100
Paired difference	+0.0074	+0.0085	+0.0092	+0.0109
Standard error	0.0031	0.0030	0.0034	0.0043
Boltz selection, Protenix evaluation				
Pretrained reference	0.4352	0.4472	0.4564	0.4634
Random-label control	0.4383	0.4549	0.4664	0.4744
Best-of-N distillation	0.4488	0.4665	0.4787	0.4889
Paired difference	+0.0105	+0.0116	+0.0123	+0.0145
Standard error	0.0046	0.0050	0.0058	0.0077

backbones in partition B. Paired standard errors are the sample standard deviation of per-backbone differences divided by $\sqrt{45}$. Table 9 reports the target-weighted summary, which does not aggregate by sequence cluster or adjust its standard error for within-cluster dependence. Figure 3C–D uses the 29 unique native sequences in partition B as the aggregation units, weighted equally. Paired differences are averaged within each native sequence and the sequence averages are then weighted equally. This weighting differs from the BGSU class weighting and structural-family inference used in the main table, so absolute values in this figure are not comparable with it. The standard error is the sample standard deviation of the sequence averages divided by $\sqrt{29}$, and intervals use the Student t quantile $t(28) = 2.048$. These intervals do not adjust for dependence among sequences within the seven sequence clusters. No summary accounts for variation across training seeds.

Paired comparisons across candidate budgets. For the backbone-weighted summary of partition B in Table 9, distillation improves Protenix TM-score at every evaluated candidate count. At $N = 8$, the paired difference from the random-candidate control is $+0.0145$ (SE 0.0077).

G.4 Evaluation limitations

The distillation results use finetuning seed 0; the reported intervals exclude training-seed variability. The pretrained reference and both finetuned arms are evaluated with eight sampling steps, matching the step count across all three models. Supervision pools use the same checkpoint that initializes fine-tuning, at temperature 0.5 with eight steps; evaluation uses temperature 0.1 with eight steps. The supervision temperature is selected on partition A and evaluated on partition B of the sequence cluster split in Appendix G.2. Figure 3C–D and Table 9 report partition B only; partition A does not contribute to these evaluation summaries. The temperature sweep uses a single finetuning seed and does not quantify training-seed variability. Each candidate receives a single Boltz prediction, leaving selection subject to folding variability even when Protenix supplies the evaluation score.