Deep Learning Framework for RNA Inverse Folding with Geometric Structure Potentials

Annabelle Yao

The Lawrenceville School 2500 Main St, Lawrence Township, NJ 08648

ayhk2017@gmail.com

Abstract. RNA's functional diversity arises from its rich three-dimensional structural landscape, yet designing RNA sequences that adopt a target 3D conformation (inverse folding) remains a fundamental challenge. I present a geometric deep learning framework that enables end-to-end RNA inverse folding by integrating Geometric Vector Perceptron (GVP) layers into a Transformer architecture, thereby explicitly encoding the spatial and directional features of RNA structures. A curated dataset of experimentally solved RNA 3D structures was constructed from the BGSU RNA database through rigorous filtering and deduplication. Model performance was evaluated using sequence recovery rate and TM-score to assess sequence accuracy and structural fidelity, respectively. Across standard benchmarks and RNA-Puzzles, the proposed method achieves state-of-the-art performance, improving the recovery rate by 85% and the TM-Score by 40% compared to existing approaches. Masked family-level validation using Rfam annotations demonstrates strong generalization to unseen RNA families, with a 132% improvement in generalization performance. Furthermore, inverse-folded sequences refolded using AlphaFold3 closely recapitulate native structures, underscoring the importance of explicit geometric representations in RNA design. This work establishes a "3D-aware" paradigm for RNA inverse folding, enabling the rapid generation of structurally valid RNA sequences in hours rather than months. By combining geometric learning with sequence modeling, the framework offers a scalable foundation for accelerating mRNA vaccine development, RNA-based therapeutics, and synthetic biology, and positions geometric deep learning as a key technology for next-generation biomolecular design.

Keywords: Geometric Vector Perceptron; Transformer; Deep Learning.

1. Introduction

RNA is a highly versatile biomolecule that plays essential roles in a broad spectrum of biological processes. Beyond serving as a messenger between DNA and proteins, RNA is central to gene regulation, catalysis, and cellular signaling. Its primary structure comprises four nucleotides—adenine (A), cytosine (C), guanine (G), and uracil (U)—and RNA molecules are generally classified into two major types: coding RNAs (mRNAs), which direct protein synthesis, and non-coding RNAs (ncRNAs), which function primarily through their intricate three-dimensional (3D) structures. The folding of RNA into these structures is largely driven by base-pairing interactions at the secondary structure level. This tight coupling between structure and function underscores the importance of accurately characterizing RNA 3D conformations, which is key to elucidating the molecular basis of many diseases and fundamental cellular processes.

Although understanding RNA structure is of profound biological significance, the conformational flexibility of RNA molecules poses a major challenge for experimental determination of their three-dimensional (3D) structures. As of December 2023, RNA-only structures constitute less than 1.0% of the approximately 214,000 entries in the Protein Data Bank (PDB), while RNA-containing complexes account for just 2.1%. This scarcity reflects the intrinsic difficulty of resolving RNA structures, which often adopt a diverse ensemble of conformations due to their dynamic nature. Despite advances in traditional experimental techniques, including X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, and cryogenic electron microscopy (cryo-EM), these methods remain limited by low throughput, labor-intensive sample preparation, and specialized equipment requirements. Consequently, the large-scale elucidation of RNA structures remains impractical through experimental means alone. To address these limitations, computational

approaches have emerged as essential tools for RNA structure prediction, offering scalable and efficient alternatives to experimental methods. The recent rise of artificial intelligence (AI) has further revolutionized the field, enabling significant gains in prediction accuracy and structural insight. In particular, the 2024 Nobel Prize in Chemistry awarded to the developers of AlphaFold [1] exemplifies the transformative power of AI in structural biology. Simultaneously, I have seen breakthroughs in RNA structure prediction, with deep learning-based methods such as Rhofold+ [2], trRosettaRNA [3], and others demonstrating remarkable improvements over traditional approaches like FARFAR2 [4] and SimRNA [5].

Another crucial aspect of RNA study lies in inverse folding, which has gained increasing significance with advances in RNA structure prediction. While elucidating the relationship between RNA structure and function remains fundamental to understanding its biological roles, inverse folding takes this one step further by seeking RNA sequences that adopt a predefined 3D conformation. This capability is essential for the rational design of synthetic RNA molecules with desired structural and functional properties. Applications span a wide range of fields, including RNA-based therapeutics, gene regulation, biosensing, and synthetic biology. Moreover, accurate inverse folding not only facilitates the engineering of functional RNA elements but also enables the discovery of novel structural motifs involved in critical cellular processes. By bridging the gap between structure prediction and functional design, inverse folding empowers researchers to manipulate RNA with precision, paving the way for breakthroughs in biotechnology, medicine, and the development of RNA-based nanodevices.

Despite the growing potential of RNA inverse folding, designing sequences that reliably adopt a target 3D structure remains a formidable challenge. This difficulty stems from the enormous combinatorial space of nucleotide sequences and the complex network of physical and chemical interactions that govern RNA folding. Traditionally, researchers have addressed this problem through methods like RNA Origami, which involves engineering nanostructures that self-assemble co-transcriptionally with the aid of software such as NUPACK [6] and ViennaRNA [7]. These methods have proven effective for in vitro applications and have supported the development of increasingly complex RNA architectures. However, they are often limited by their reliance on secondary structure approximations and may not generalize well to the full three-dimensional complexity required for functional RNA design in more diverse biological contexts.

The potential of RNA inverse folding extends far beyond the scope of traditional computational strategies. Recent advances in deep learning have introduced powerful new approaches for navigating the immense complexity of RNA folding. While protein inverse folding has seen rapid progress through groundbreaking models such as AlphaFold and ProteinMPNN, RNA presents unique challenges, including non-canonical base-pairing, intricate tertiary interactions, and highly diverse structural motifs. These complexities have limited the widespread adoption of deep learning for RNA inverse folding. As a result, fewer studies have applied deep learning in this domain, with notable exceptions including Monte Carlo tree search–based frameworks [8] and the generative diffusion model RiboDiffusion [9]. In addition, successful applications of machine learning, such as transformers in computer vision [10], have shown that deep learning models work best when sharing features with the domain's problem structure. Building upon these insights, this work proposes a deep learning approach that integrates geometric considerations directly into the model architecture by leveraging a Geometric Vector Perceptron (GVP) with a Transformer to address a new significant problem within the RNA field. [11]. My approach achieves substantially improved performance on both standard benchmarks and RNA-Puzzles, outperforming existing methods in terms of sequence recovery and structural fidelity, and highlighting its effectiveness in accurate RNA sequence design and 3D structure prediction.

2. Methods

2.1. Data Preprocessing

The dataset used for this study and evaluation was obtained from the BGSU RNA list version 3.321, which contains a collection of experimentally solved RNA 3D structures. The initial dataset included over 6300 RNA 3D structures, downloaded from the RCSB Protein Data Bank (PDB). To preprocess the dataset, the RNA sequences were first filtered to retain only those with lengths between 20 and 512 nucleotides. Following this, single-chain RNA sequences were extracted from the resulting filtered 3D structures. Redundancy was then removed using the CD-Hit tool with an 80% sequence similarity cutoff, resulting in approximately 2500 unique single-chain RNA 3D structures. For each nucleotide in these sequences, key atoms (P', C5', O5', C4', O4', C3', O3', C2', O2', C1', N1, C2, O2, N3, C4, N4, C5, C6, OP1, and OP2), which are essential for my model, were extracted. Any missing atoms were filled in using Arena [12]. The sequence of these single chains formed the desired output, which was used for both structural and sequence comparisons. This pre-processing pipeline ultimately produced a dataset of around 2500 RNA single-chain 3D structures, which was utilized for evaluating the recovery rate and structural prediction. A diagram of my data pre-processing method is shown below in Figure 1

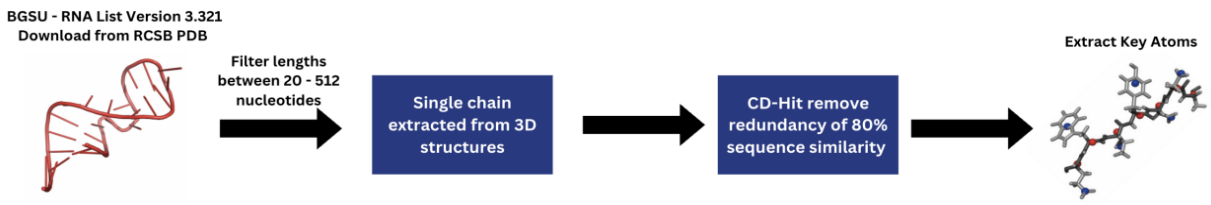

Figure. 1 The overall method used to preprocess my data. (Graph created by the student researcher using Canva and the structures downloaded from RCSB PDB)

Given an RNA tertiary structure, I extracted its RNA sequence along with the coordinates of the key atoms for each nucleotide. The key atoms include: P', C5', O5', C4', O4', C3', O3', C2', O2', C1', N1, C2, O2, N3, C4, N4, C5, C6, OP1, and OP2. Based on this information, I frame my RNA inverse folding problem: Given the coordinates of these key atoms, the goal is to generate RNA sequences that fold into the specified structure. To solve this problem, my aim was to construct a deep learning model capable of predicting RNA sequences that correspond to a given structural configuration. I used a Geometric Vector Perceptron (GVP) and a transformer to generate RNA sequences. The general method is shown in Figure 2.

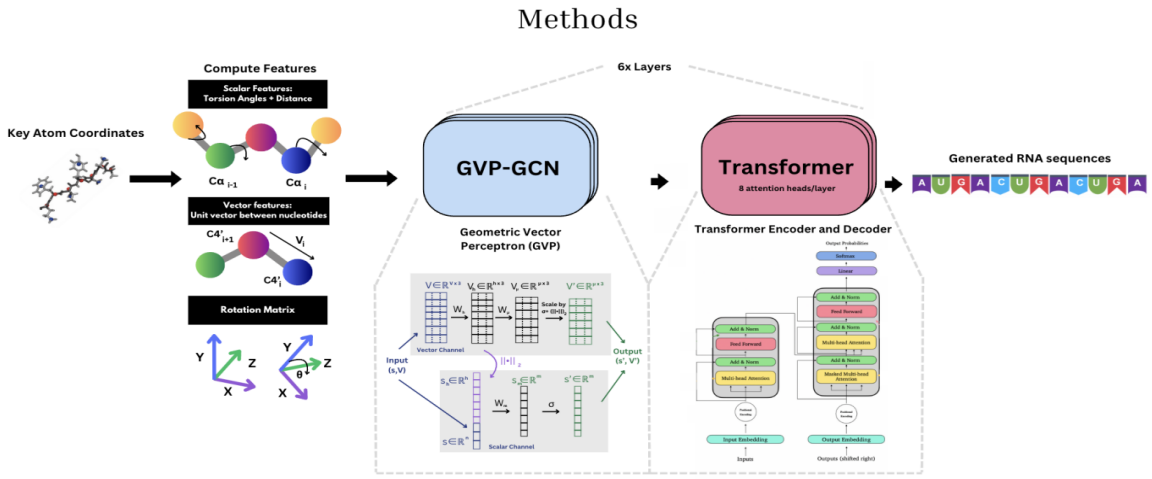

Figure. 2 The workflow of my framework. First, key atom coordinates are extracted from the pdb file. Then, the scalar and vector features of the key atom are computed along with the rotation matrix to ensure rotational invariance. These features will be used to make a graph where the nodes are each key atom and the weights are the features. The graph is then passed through the GVP and a transformer to generate an RNA sequence. (Graph created by the student researcher using Canva and the structures downloaded from RCSB PDB)

2.2. Structural Features

After extracting the coordinates of the key atoms from the PDB file, I calculate the torsion angles and the vectors between the coordinates. Torsion angles are the angles between two planes formed by specific key atoms and describe the flexibility of the backbone, playing a crucial role in defining a RNA's 3D structure. RNA has six backbone torsion angles per nucleotide, defined around the covalent bonds, the glycosidic bond, and the sugar pucker. These angles are measured around the bonds connecting different backbone atoms as below. Each angle is calculated by finding the difference between four points p0 - p1 - p2 - p3. Then, the difference between p2 and p1, the unit vector b1 along the axis of rotation, is normalized to ensure that it has a magnitude of 1. I calculate the orthogonal vectors by removing the component along b1 using the dot product. These vectors are then used to calculate the torsion angles with the equations below, with (i-1) and (i+1) indicating atoms from adjacent nucleotides in the RNA chain.

- α (alpha): O3' (i-1) → P → O5' → C5'
- β (beta): P → O5' → C5' → C4'
- γ (gamma): O5' → C5' → C4' → C3'
- δ (delta): C5' → C4' → C3' → O3'
- ϵ (epsilon): C4' → C3' → O3' → P (i+1)
- ζ (zeta): C3' → O3' → P (i+1) → O5' (i+1)

These angles serve as the scalar features of each nucleotide. The vector values for each nucleotide are then calculated by finding the displacement vector, a 3D spatial vector, between the C4' atoms of nucleotide i and nucleotide i+1. The scalar and vector features are stored in matrices S and V were

$$\begin{aligned} S &\in \mathbb{R}^{N \times d_s} \\ V &\in \mathbb{R}^{N \times d_v \times 3} \end{aligned} \tag{1}$$

Where N is the number of nucleotides, d_s is the number of scalar features, and d_v is the number of vector features.

Moreover, to ensure that the model is learning rotation-invariant features, I extract 3 atoms per nucleotide, specifically N1, C4', and C1', to construct a local coordinate frame. I use these atoms to compute a rotation matrix $\mathbb{R}$ for each nucleotide, ensuring that all vector features are expressed in a consistent local frame. First, I compute the Euclidean norm that I call L2 along a given dimension. The three orthogonal base vectors are calculated through these equations:

$$\mathbf{e}_1 = \text{normalize}(\mathbf{v}_1, \text{dim} = -1) \quad \text{(unit vector along C} \rightarrow \text{CA)} \tag{2}$$

$$\mathbf{e}_2 = \text{normalize}(\mathbf{u}_2, \text{dim} = -1) \quad \text{(adjusted vector orthogonalized against} \mathbf{e}_1) \tag{3}$$

$$\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2 \quad \text{(cross product ensuring a right-handed coordinate system)} \tag{4}$$

These vectors form the rotation matrix:

$$R = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \mathbf{e}_3] \tag{5}$$

Which defines the local coordinate frame. After I generated these structural features, I applied an Encoder-Decoder-based network to process my input.

2.3. Feature Encoding

I use a model architecture combining Geometric Vector Perception (GVP) and a Transformer encoder framework to encode RNA structural information. The RNA structure is represented as a graph, where each nucleotide is treated as a node and the node feature is calculated using torsion angles and rotation matrices. The edges between nodes are determined by the spatial proximity of nucleotides, calculated through pairwise distances. To construct this graph, I calculate a distance matrix based on the coordinates of the C4' atoms of each nucleotide, which serves as a reference for the relative positioning of nucleotides. From this matrix, I compute the Euclidean distances between all pairs of nucleotides, storing these distances in another matrix.

Using the calculated distance matrix, I identify the five nearest neighbors for each nucleotide and establish edges between the corresponding nodes. This process ensures that only relevant, short-range connections are included in the graph, representing the structural relationships between adjacent nucleotides. The RNA molecule is now fully encoded as a graph, with nodes (nucleotides), edges (connections between neighboring nucleotides), and node features (scalar and vector features calculated from the structural data).

The constructed graph is then fed into a Geometric Vector Perception (GVP) layer. GVP is a powerful module designed to learn vector- and scalar-valued functions over geometric vectors and scalars. Given a tuple $(\mathbf{s}, \mathbf{V})$ of scalar features $\mathbf{s} \in \mathbb{R}^n$ and vector features $\mathbf{V} \in \mathbb{R}^{\nu \times 3}$, the GVP module computes new features $(s', \mathbf{V}') \in \mathbb{R}^m \times \mathbb{R}^{\mu \times 3}$. The overall diagram of a GVP is shown below in Figure 3.

The GVP module first applies nonlinear transformations to the features while ensuring geometric consistency. Specifically, ReLU activations are applied to the scalar features, and a gating mechanism is applied to the vector features, ensuring rotational invariance. This design allows the model to refine nucleotide representations by incorporating local information from neighboring nucleotides. After the feature transformation, the GVP graph convolution (GVP) aggregates information from each nucleotide's nearest neighbors, using the previously computed scalar and vector features $(\mathbf{s}, \mathbf{V})$ to update the features $(\mathbf{s}', \mathbf{V}')$ based on local geometric relationships.

To further enhance the model's ability to capture complex inter-nucleotide dependencies, random edge features are introduced at each convolution step. This helps the model learn additional nuances between the nucleotides. The resulting GVP layer performs the graph convolution as described in Algorithm 1, refining the features with each layer.

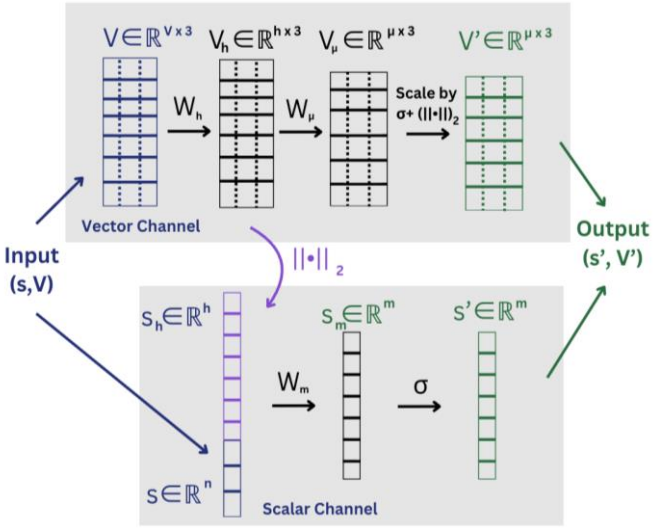

Figure. 3 GVP structure. (Graph recreated by the student researcher using Canva based on [11])

$$\mathrm{GVP}(\mathbf{s}, \mathbf{V}) = (\mathbf{s}', \mathbf{V}') \tag{6}$$

Algorithm 1: Geometric Vector Perceptron Algorithm [11]

Input: Scalar and vector features $(\mathbf{s}, \mathbf{V}) \in \mathbb{R}^n \times \mathbb{R}^{\nu \times 3}$

Output: Scalar and vector features $(\mathbf{s}', \mathbf{V}') \in \mathbb{R}^m \times \mathbb{R}^{\mu \times 3}$.

$$h \leftarrow \max(\nu, \mu)$$

GVP:

$$V_h \leftarrow W_h V \qquad \in \mathbb{R}^{h \times 3} \quad (6)$$

$$V_\mu \leftarrow W_\mu V_h \qquad \in \mathbb{R}^{\mu \times 3} \quad (7)$$

$$s_h \leftarrow \| V_h \|_2 \text{ (row-wise)} \qquad \in \mathbb{R}^{h} \quad (8)$$

$$v_\mu \leftarrow \| V_\mu \|_2 \text{ (row-wise)} \qquad \in \mathbb{R}^{\mu} \quad (9)$$

$$s_{h+n} \leftarrow \mathrm{concat}(s_h, s) \qquad \in \mathbb{R}^{h+n} \quad (10)$$

$$s_m \leftarrow W_m s_{h+n} + b \qquad \in \mathbb{R}^{m} \quad (11)$$

$$s' \leftarrow \sigma(s_m) \qquad \in \mathbb{R}^{m} \quad (12)$$

$$V' \leftarrow \sigma^+(v_\mu) \odot V_\mu \text{(row-wise multiplication)} \qquad \in \mathbb{R}^{\mu \times 3} \quad (13)$$

return $(\mathbf{s}', \mathbf{V}')$

To increase the model's ability to capture hierarchical and local patterns, I use six GVP convolution layers. These layers progressively expand the feature set with additional learned representations before condensing them into a lower-dimensional space while retaining the critical structural information necessary for RNA folding prediction. The output of the GVP are the post-processing high-level features from key atoms.

2.4. Transformer

Then, the new scalar and vector features are fed through a transformer encoder. I convert token indices from the RNA sequence input X to embeddings. The sequence length features go through a fully connected layer and are then added to each token embedding. Then, I add positional encoding to each embedding to ensure the transformer can capture local and global dependencies, making the final input embedding as below.

$$H = \text{token\_embeddings} + \text{position\_embeddings} \tag{7}$$

The inputs are then run through a transformer. I have a total of 6 layers, with 8 attention heads in each. A diagram of my transformer encoder-decoder is below in Figure 4.

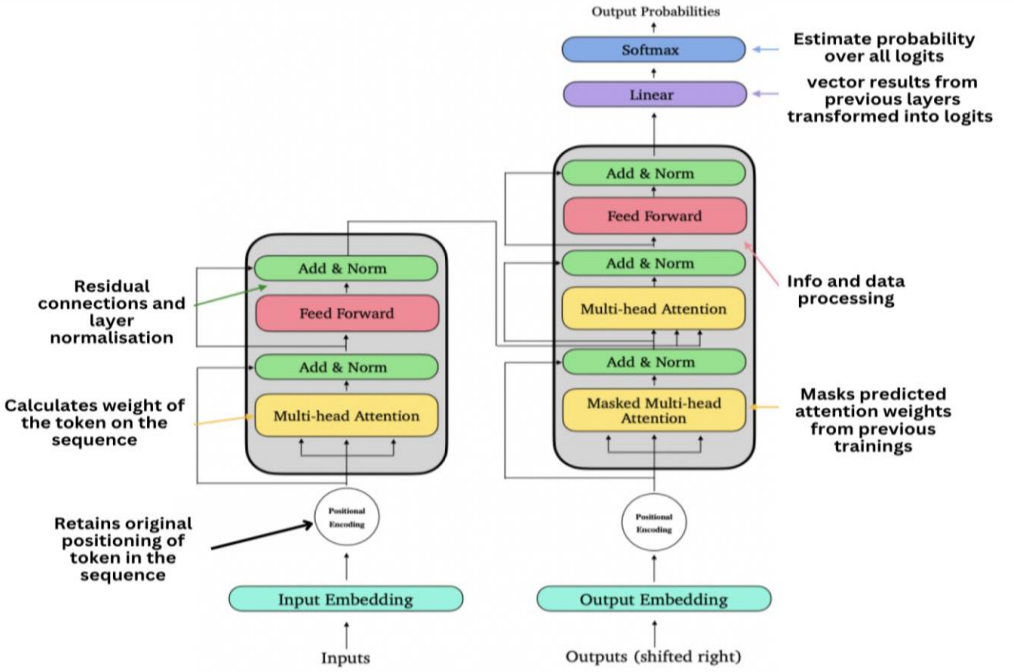

Figure. 4 Transformer structure. (Graph created by the student researcher using Canva based on prior research [13])

The transformer processes the embeddings, with the model taking the output of the current sequence as the input for the decoder, and then autoregressively outputs the predicted sequence. For every autoregressive generation of a fixed sequence length, a forward pass through the transformer decoder is conducted for the sequence length feature. The model will output probability distributions over the vocabulary for every position in the sequence, generating the sequence one token at a time by evaluating the previous token and selecting the most probable token as the next token in the sequence. The newly predicted token is then appended to the generated sequence. The decoder will continuously generate tokens for the sequence until an 'end' token is generated or there are 50 tokens in the sequence.

2.5. Model Training and Evaluation

I train my GVP-Transformer model end-to-end on a curated RNA dataset, splitting it into training, validation, and test sets. All RNA structures are preprocessed to form graphs suitable for the GVP encoder, and the corresponding sequences are tokenized to produce indices for the Transformer. Training is conducted on two NVIDIA GeForce RTX 4090 GPUs, allowing for efficient parallelization of both the GVP and Transformer components. I optimize the model parameters using the Adam optimizer with an initial learning rate of 1×10^{-4}, a batch size of 16, and a weight decay coefficient of 5×10^{-5} to control overfitting. Dropout at a rate of 0.2 is applied to the Transformer layers, and gradient clipping is set to a global norm of 1.0 to stabilize updates. The model is trained for up to 80 epochs, with early stopping triggered if the validation loss fails to improve for 5 consecutive epochs. I use cross-entropy as the loss function.

I evaluate my RNA sequence design using two key performance metrics: recovery rate and structural recovery. These metrics were calculated based on the comparison between the designed RNA sequences and their natural counterparts, as well as the structural prediction of the designed sequences. For each metric, a higher score indicates a superior outcome, with 0.5 considered a high score.

- The recovery rate (R) is calculated by comparing the designed RNA sequence with the natural RNA sequence. The recovery rate is determined by the formula:

$$R = \frac{\text{Number of correctly recovered nucleotides}}{\text{Total number of nucleotides}} \times 100 \quad (8)$$

Where the "number of correctly recovered nucleotides" refers to the nucleotides in the designed RNA sequence that match exactly with the natural RNA sequence. The "total number of nucleotides" refers to the length of the RNA sequence.

• **Structural recovery.** I evaluated the structural similarity between the predicted structure (S_{pred}) and the experimental structure (S_{exp}) using the TM-score [14]. After optimal rigid-body superposition, the TM-score is

$$\text{TM} = \max\left[\frac{1}{L_{\text{ref}}}\sum_{i=1}^{L_{\text{ali}}}\frac{1}{1+\left(\frac{d_i}{d_0(L_{\text{ref}})}\right)^2}\right], \tag{9}$$

where d_i is the distance between the i-th pair of aligned residues in S_{pred} and S_{exp}, L_{ref} is the length of the reference structure, L_{ali} is the number of aligned residue pairs, and $d_0(\cdot)$ is a length-dependent scale (e.g., $d_0(L) = 1.24\sqrt[3]{L-15} - 1.8$ in Å). Higher TM-scores indicate greater structural similarity.

To obtain S_{pred}, I folded the designed sequences and then computed TM-scores to the corresponding experimentally solved structures.

3. Results

3.1. Model Performance and Benchmarking

Table 1 shows the comparison of recovery rates and TM-scores for RNA sequence recovery and structural prediction. My model outperforms the other models in all areas consistently, achieving a higher recovery rate and TM-score for both standard benchmark and RNA-puzzles.

Table 1. Comparison of recovery rates and TM-scores for RNA sequence recovery and structural prediction. The first two columns correspond to results from the standard benchmark, while the last two columns show the results from the RNA-Puzzles dataset.

Methods/Metrics	Recovery Rate		TM-score	
	Standard	RNA-Puzzles	Standard	RNA-Puzzles
Ours	**0.413**	**0.481**	**0.281**	**0.332**
RDesign	0.328	0.403	0.252	0.272
Ribologic [14]	0.142	0.242	0.190	0.250
Learna [15]	0.213	0.293	0.224	0.194

To evaluate the performance of my RNA sequence design, I conducted a 10-fold cross-validation. This process involves dividing the dataset into 10 subsets, where 9 subsets are used for training and the remaining subset is used for testing. The procedure is repeated 10 times, ensuring that each subset serves as the test set once. The results are then averaged over all 10 folds to provide a robust measure of performance. In my comparison, I first downgraded my 3D RNA structure into 2D and then applied 2D sequence design methods. This approach allowed me to compare my method with other established 2D design methods. For structural comparison, I used AlphaFold3 to fold back the designed sequences. The recovery rate and TM-score were used as key performance metrics. The recovery rate measures how well the designed sequence matches the natural RNA sequence, while the TM-score evaluates the similarity between the predicted and experimental structures. In my experiments, based on 80% sequence similarity cutoff for train-test split, the methods were compared.

The calculated average Recovery Rate for all other models on the standard benchmark dataset is 0.223. Using this statistic, it can be calculated that my created model improved the recovery rate by

$$0.413/0.223-1 \approx 85\% \tag{10}$$

The results of my standard benchmark, as shown in Table 1, demonstrate that my approach outperforms other methods such as Ribologic and Learna in both recovery rate and TM-score.

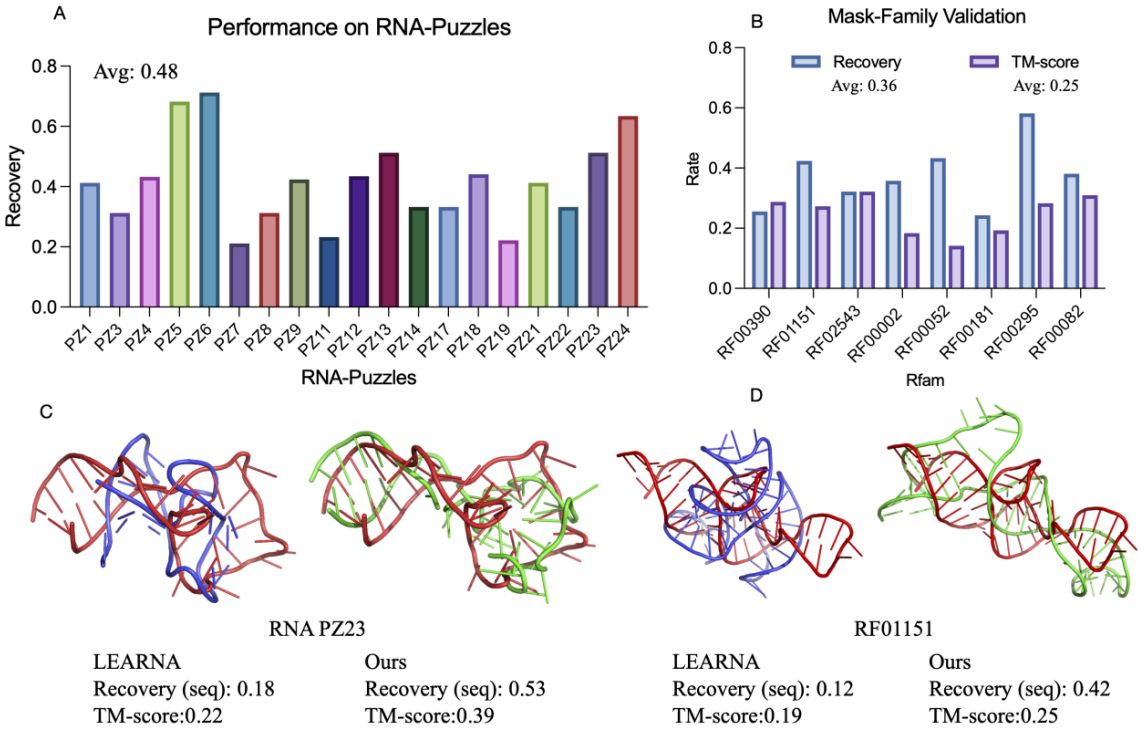

Figure. 5 Results on RNA-Puzzles and Mask-Family. (A) Bar chart of the recovery rate for each RNA-Puzzle ID using my method. (B) BarResults on RNA-Puzzles and Mask-Family. (A) Bar chart of the recovery rate for each RNA-Puzzle ID using my method. (B) Bar chart of the validation results for the Mask-Family task, reporting both recovery rate and fold-back TM-score for each family. (C) Comparison of RNA design outcomes for PZ23, contrasting LEARNA's predictions (blue structure) with ours (green), both folded using AlphaFold3 and overlaid with the ground-truth structure (red). (D) The same comparison for the RF01151 family. (Figures A and B are created by the student researcher using GraphPad Prism and Figures C and D are created by the student researcher using PyMOL)

3.2. Mask-family Validation

To further assess the performance of my RNA sequence design, I conducted a cross-family validation using Rfam annotations. In this validation, we split my dataset into training and test sets using a leave-one-out strategy. Specifically, for each family in the dataset, I trained the model on all families except for one, which was held out as the test set. This process was repeated for each family, ensuring that every family was tested independently. This approach provides a more robust evaluation of how well the model generalizes across different RNA families.

I used the curated PDB dataset from the previous experiment, along with family annotations provided by Rfam. The Rfam annotations categorize RNA families, allowing me to validate the model's ability to predict sequences and structures for RNA families it has not seen before. My method received an average Recovery Rate of 0.36 and average TM-score of 0.254 for cross-family results, showing robust results across diverse families (Figure 5B). I compared this to Learna's sequence generalizability of 0.155 and my model achieved higher results. Using this statistic, it can be calculated that my created model improved the sequence generalizability by 132%

3.3. RNA-puzzle Evaluation

In addition to cross-family validation, I also tested my model on open-wide challenges such as RNA-Puzzles, a renowned benchmark for RNA sequence design and structure prediction. For this evaluation, I filtered out sequences with a sequence similarity greater than 80% from my training set to ensure that the test sequences were sufficiently distinct from those used during training. This

approach is designed to test the model's generalization ability on new and diverse RNA structures. I trained and tested my model on the RNAPuzzles dataset and evaluated its performance using the recovery rate and TM-score metrics. The recovery rate measures the accuracy of the designed RNA sequences when compared to the natural sequences, while the TM-score evaluates the similarity between the predicted and experimental RNA structures. The results of this evaluation (Table 1, Figure 5A) demonstrate that my method outperforms other approaches, such as RDesign, Ribologic, and Learna, in both recovery rate and TM-score. Specifically, I method achieved a recovery rate of 0.481 and a TM-score of 0.332, indicating superior performance in both sequence recovery and structural prediction accuracy. The other models produced an average TM-score on RNA-puzzles dataset of 0.238. Using this statistic, it can be calculated that my created model improved the TM-score by

$$0.332/0.238-1\approx 40\% \tag{11}$$

4. Discussion

Compared to conventional RNA inverse folding methods, my model demonstrates substantial improvements across multiple evaluation benchmarks. On both the standard benchmark set and the RNA-Puzzles challenge, it consistently outperforms existing approaches in terms of sequence recovery and structural fidelity. For RNA-Puzzles, my model achieves a recovery rate of 0.481 and a TM-score of 0.332. Given that TMscores are calculated by refolding the generated sequences using AlphaFold3 and comparing the resulting structures with experimentally determined ones, it is important to note that AlphaFold3 itself typically yields

an average TM-score of approx. 0.5. Thus, a TM-score of 0.332, starting from de novo generated sequences, is high and demonstrates the structural validity of my designs. Additionally, the recovery rate, which evaluates exact sequence-level matches, surpasses the TM-score, further underscoring the precision of my model in generating structurally informed sequences. My method also excels in Mask-Family validation, achieving a sequence similarity score of 0.42 for RF01151. In this setting, scores above 0.4 are considered statistically significant, highlighting the model's strong generalization capacity across different RNA families.

When benchmarked against existing models, my approach demonstrates clear superiority. For RNA-Puzzles, Ribologic reports a recovery rate and TM-score of 0.242 and 0.250, respectively, while Learna performs even lower at 0.293 and 0.194. RDesign, despite achieving a relatively high recovery rate of 0.403, lags in structural accuracy with a TM-score of only 0.272. In contrast, my model achieves high performance across both metrics, reflecting its balanced capability in accurate sequence generation and structural realization.

Similarly, in the standard benchmark, my model attains a recovery rate of 0.413 and a TM-score of 0.281. These results outperform RDesign (recovery rate 0.328; TM score 0.252), Ribologic (recovery rate 0.142; TM-score 0.190), and Learna (recovery rate 0.213; TM-score 0.224). Notably, my model improves upon Ribologic's recovery rate by nearly fourfold and more than doubles that of Learna. These comprehensive gains across both sequence- and structure-level metrics firmly establish the superiority of my framework and mark an advancement toward reliable, high-accuracy RNA design.

Despite its strong performance, my model has several limitations. It struggles with highly complex RNA structures, such as those containing long-range interactions, pseudoknots, and multi-loop junctions, although geometric features help mitigate this challenge. The model is limited to single RNA design and does not account for co-folding or binding constraints in RNA complexes. Additionally, it predicts only the most stable conformation, ignoring alternative functional states common in riboswitches and ribozymes. Finally, the scarcity of high-quality RNA structural data hampers training, underscoring the need for expanded datasets to further enhance model performance.

My new method of computational RNA inverse-folding has a wide range of applications. Most notably, its ability to encode precise protein-binding RNA architectures will allow me to engineer

RNAs that recruit specific RNA-binding proteins to targeted transcripts, enabling post-transcriptional regulation with high molecular specificity. This same framework can be used to create therapeutic aptamers—structured RNAs that tightly bind and neutralize disease-associated proteins—through controlled presentation of binding loops and tertiary contacts. By starting from the desired structure and going backward to the nucleotide sequence, I can efficiently explore a much broader region of the design space, identifying sequence variants that preserve the functional fold, thus designing its interaction capabilities. This work highlights the transformative potential of machine learning in RNA research and positions AI-driven methods as foundational tools for the next generation of molecular design.

5. Conclusion

In this research, I identified a significant problem in the field of RNA sequence discovery and created and applied a new framework to provide a solution for the problem. My created framework generates results that surpass the capabilities of all compared RNA prediction models with higher recovery rates and TMscores, demonstrating superior sequence reconstruction capabilities and structural prediction accuracy. I also made my code open-source so that the public has free access to my created model.1 Prior to the rise of machine learning, RNA design relied heavily on dynamic programming tools such as ViennaRNA and RNAfold, which struggle with scalability due to the exponential growth of possible structures with RNA length. For each RNA of length n, there are n $-3/2 \times 1.8$ n possible structures, each taking a large amount of time to design. Experimental design was often a slow, trial-and-error process, taking weeks to months. This work establishes a "3D-aware" paradigm for RNA inverse folding, enabling the rapid generation of structurally valid RNA sequences in hours rather than months. By combining geometric learning with sequence modeling, the framework offers a scalable foundation for accelerating mRNA vaccine development, RNA-based therapeutics, and synthetic biology, and positions geometric deep learning as a key technology for next-generation biomolecular design.

References

[1] Dauparas, J. et al. Robust deep learning–based protein sequence design using proteinmpnn. Science 378, 49–56 (2022).

[2] Shen, T. et al. Accurate rna 3d structure prediction using a language model-based deep learning approach. Nature Methods 1–12 (2024).

[3] Wang, W. et al. trrosettarna: automated prediction of rna 3d structure with transformer network. Nature Communications 14, 7266 (2023).

[4] Watkins, A. M., Rangan, R. & Das, R. Farfar2: improved de novo rosetta prediction of complex global rna folds. Structure 28, 963–976 (2020).

[5] Boniecki, M. J. et al. Simrna: a coarse-grained method for rna folding simulations and 3d structure prediction. Nucleic acids research 44, e63–e63 (2016).

[6] Zadeh, J. N. et al. Nupack: Analysis and design of nucleic acid systems. Journal of computational chemistry 32, 170–173 (2011).

[7] Lorenz, R. et al. Viennarna package 2.0. Algorithms for molecular biology 6, 1–14 (2011).

[8] Yang, X., Yoshizoe, K., Taneda, A. & Tsuda, K. Rna inverse folding using monte carlo tree search. BMC bioinformatics 18, 1–12 (2017).

[9] Huang, H., Lin, Z., He, D., Hong, L. & Li, Y. Ribodiffusion: tertiary structure-based rna inverse folding with generative diffusion models. Bioinformatics 40, i347–i356 (2024).

[10] Yao, A., Su, O. & Kumar, S. Enhancing real-time vision systems: Integrating dynamic vision sensors with vision transformers to increase computational efficiency. Applied and Computational Engineering107, 73–82 (2024).

[11] Jing, B., Eismann, S., Suriana, P., Townshend, R. J. & Dror, R. Learning from protein structure with geometric vector perceptrons. arXiv preprint arXiv:2009.01411 (2020).

[12] Perry, Z. R., Pyle, A. M. & Zhang, C. Arena: rapid and accurate reconstruction of full atomic rna structures from coarse-grained models. Journal of Molecular Biology 435, 168210 (2023).

[13] Yao, A. A novel interpretative deep neural network with grad-cam's heatmap for the early diagnosis of alzheimer's disease. Theoretical and Natural Science 49, 106–119 (2024).

[14] Zhang, Y. & Skolnick, J. Scoring function for automated assessment of protein structure template quality. Proteins: Structure, Function, and Bioinformatics 57, 702–710 (2004).

[15] Wu, M. J., Andreasson, J. O., Kladwang, W., Greenleaf, W. & Das, R. Automated design of highly diverse riboswitches. bioRxiv 603001 (2019).

[16] Runge, F. & Hutter, F. Machine learning for rna design: Learna. In RNA Design: Methods and Protocols, 63–93 (Springer, 2024).