RTfold: RNA secondary structure prediction using deep learning with domain inductive bias

Andrew J. Jung [1 2] **Leo J. Lee** [1 2] **Alice J. Gao** [1 2] **Brendan J. Frey** [1 2]

Abstract

RNA secondary structure prediction with deep learning is challenging due to the limited training data and the inherent difficulties in predicting structured output, necessitating good inductive biases for generalization. In this work, we propose *RTfold* with the following key motivating ideas: 1) end-to-end training combined with constrained optimization, 2) neural architecture with layer-wise recurrent inductive bias, and 3) a larger training set augmented with synthetic data for pre-training. *RTfold* achieves good performance on our preliminary evaluations, and we show how the above three motivating ideas contribute to better generalization.

1. Introduction

An RNA molecule's function is often dependent on its structure. Experimentally determining RNA structures can be challenging so computational prediction methods are often used as alternatives or to augment the experiments. Many computational methods focus on RNA secondary structure[1] because this is easier to predict than three-dimensional structure. Furthermore, RNA functions can often be well understood based on secondary structures without the full structures (Sloma & Mathews, 2016).

Given an input RNA sequence, a classical computational approach is to use dynamic programming (DP) to predict the optimal secondary structure, consisting of substructures with the best total score. Methods such as *RNAfold* (Lorenz et al., 2011) use free-energy measurements for scoring, but experimental limitations restrict the types of free-energy parameters and how accurate they can be. Alternatively, the scoring parameters can be learned from datasets of known secondary structures. In *MXfold2* (Sato et al., 2021), a deep learning model is trained to predict the scoring parameters. Nevertheless, the inherent assumptions underlying DP-based algorithms — the substructure scores are additive and context independent — might not accurately reflect the true RNA folding and limit the types of predicted structures.

Attempts to overcome these limitations have motivated deep learning approaches which directly predict RNA secondary structure without DP. However, achieving good generalization has been challenging, partly due to limited amount of data. *SPOT-RNA* (Singh et al., 2019) addresses this by constructing a training set with 10,814 secondary structures from *bpRNA* dataset (Danaee et al., 2018). *SPOT-RNA* outperforms the existing DP-based methods on held-out test data, but it does not perform as well on new datasets. Furthermore, *SPOT-RNA* does not constrain its output to be a valid RNA secondary structure, but incorporating structural constraints could improve generalization (Amos & Kolter, 2017). *E2Efold* (Chen et al., 2020) attempts to achieve this by constraining its output with convex optimization. During training, the algorithm for solving the convex problem is unrolled for back-propagation. Unfortunately, often the output of *E2Efold* still significantly violates the structural constraints because of relaxations introduced to make this approach feasible (details in Appendix.A). Furthermore, it has been shown that *E2Efold* generalizes poorly outside of its training data (Sato et al., 2021; Fu et al., 2022). *UFold* (Fu et al., 2022) attempts to improve upon these two deep learning methods by using U-Net (Ronneberger et al., 2015) based architecture with *SPOT-RNA*'s training set and *E2Efold*'s approach for constraining the output.

DP-based methods can potentially generalize well from strong inductive biases but DP algorithms can be limiting, whereas deep learning models are universal approximators without such restrictive assumptions but have difficulties generalizing. In an attempt to bridge this gap, we propose *RTfold*. Our contributions can be summarized as below:

1. We propose a framework for training a deep learning model end-to-end with RNA secondary structure constraints. The constraints are formulated as a linear program (LP) and back-propagation through this LP is

[1]University of Toronto [2]Vector Institute. Correspondence to: A. Jung <andrewjung@psi.toronto.edu>, L. Lee <ljlee@psi.toronto.edu>.

The 2022 ICML Workshop on Computational Biology. Baltimore, Maryland, USA, 2022.

[1]RNA secondary structure, formed by hydrogen bonds between pairs of nucleotide bases, constitutes the scaffolding of RNA three-dimensional structure (Mathews, 2006).

accomplished by perturbation with Fenchel-Young loss (Berthet et al., 2020). Our approach guarantees that the constraints are always satisfied, and the inductive bias helps the model generalize.

2. We propose a deep learning architecture based on layer-wise recurrence and self-attention. Our architecture enhances the model's expressiveness without the corresponding increase in the number of model parameters.

3. We construct a new training set larger than the one proposed in *SPOT-RNA*, and augment this with pre-training on a synthetic dataset.

2. Methods

2.1. End-to-end training with RNA secondary structure constraints

For an input RNA sequence x of length L, a deep learning model F can be trained to predict the most likely secondary structure, represented as a symmetric base-pairing matrix $M \in \{0,1\}^{L \times L}$. However, the model output $F(x) \in \mathbb{R}^{L \times L}$ is normally unconstrained, so $F(x)$ will often violate the following three structural constraints, conventionally enforced for RNA secondary structures (Steeg, 1993):

1. Canonical pairs: $M_{ij} = 1$ only if $x_i x_j$ is AU, UA, GC, CG, GU, or UG pairs
2. No pairing within 3 bases: $M_{ij} = 0$ if $|i - j| < 3$
3. At most a single pair for each base: $\forall i, \sum_{j=1}^{L} M_{ij} \leq 1$

2.1.1. Constrained optimization for secondary structure

Here, we describe a linear program T for constraining $F(x)$ into a valid secondary structure prediction $\hat{M} \triangleq T(F(x))$. T is formulated as[2]:

$$\begin{aligned} \hat{M} = T(F(x)) = \arg\max_{M} \quad & \langle F(x), M \rangle \\ \text{s.t.} \quad & 0 \leq M \leq I(x) \\ & M\mathbb{1} \leq \mathbb{1} \\ & M = M^T; \end{aligned} \tag{1}$$

where $M \in \mathbb{R}^{L \times L}$, $I(x) \in \{0,1\}^{L \times L}$ with $I(x)_{ij} = 0$ for base-pairs $x_i x_j$ violating the constraints 1 and 2, and $\mathbb{1}$ being an $L \times 1$ vector of ones. The inequality involving $\mathbb{1}$ encodes the constraint 3. The solution $\hat{M}$ is always binary even though the optimization variable M is not constrained to be so because the fundamental theorem of linear programming (Thm. 6 (Dantzig et al., 1955)) states that its solution is always on a vertex of the domain polytope. Any generic linear programming solver can be used for T.

Training of the neural network can be done separately without T. For instance, F can first be trained using the element-wise cross-entropy loss between $F(x)$ and M [3], after which, T can be applied to constrain the output of the already trained model.

However, end-to-end learning with T allows F to be 'aware' of the constraints during training. This inductive bias can improve generalization, especially given the limited amount of data available for RNA secondary structures. The biggest challenge in the end-to-end training is back-propagating through T.

2.1.2. Differentiating discrete optimization with perturbation and Fenchel-Yonge loss

The predicted base-pairing matrix $\hat{M} = T(F(x))$ is discrete, so T is not differentiable w.r.t. its input $F(x)$[4]. Applying a convex regularizer to the objective of Eq.1 can make T differentiable, and its gradient can be computed from the implicit differentiation of T's optimality conditions (Amos & Kolter, 2017). Unfortunately, this approach incurs prohibitive $O(L^6)$ computational cost w.r.t. the input sequence length L. A solution proposed in *E2Efold* is back-propagation through the unrolled algorithm, but in order to prevent unbounded computation graph, the number of iterations is capped by a hyper-parameter. This can result in violation of the structural constraints because the unrolled constrained optimization might not fully converge (details in Appendix A).

Our proposed approach, inspired by Berthet et al. (2020), overcomes these limitations. First, $F(x)$ is perturbed with random variable $Z \in \mathbb{R}^{L \times L}$ where each elements are i.i.d. standard Gaussian. Random perturbation with Z can be considered as an implicitly way of applying a regularizer to relax a discrete problem (Hazan et al., 2016). Monte-Carlo estimate of the perturbed solution $\mathbb{E}_Z[T(F(x) + \varepsilon Z)]$ is computed as:

$$\hat{M}_\varepsilon(F(x)) = \frac{1}{N} \sum_{n=1}^{N} T(F(x) + \varepsilon Z^{(n)}) \tag{2}$$

where $\varepsilon > 0$ is a hyperparameter controlling the scale of the perturbation.

Then, Fenchel-Yonge (FY) loss (Blondel et al., 2020) is applied to Eq.2, which allows easy gradient computation

[2] $\langle \mathcal{F}_\theta(x), M \rangle$ is the Frobenius inner product and $\leq$ denotes element-wise inequality

[3] This is how SPOT-RNA is trained

[4] $\partial \hat{M}^* / \partial \mathcal{F}_\theta(x)$ is zero almost everywhere and infinite at the transition point from one solution to another

without any explicit differentiation[5]. As shown in Section 4 of Berthet et al. (2020), the gradient can be computed as:

$$r_{F_\theta} L\ (F\ (x); M\ ) = \hat{M}\ (F\ (x)) \quad M$$
$$= \frac{1}{N} \sum_{n=1}^{N} T\ \ F\ (x) + Z^{(n)} \quad M \tag{3}$$

Intuitively, the gradient of FY-loss reduces the gap between the expectation of the perturbed output and the target, minimizing the duality gap (Blondel et al., 2020; Berthet et al., 2020).

As shown above, random perturbation with FY-loss allows easy gradient computation without any explicit differentiation, but only if repeatably solving T can be done efficiently. Fortunately, T can be reformulated to leverage the sparsity of $F\ (x) + Z$. Combined with trivial parallelization over N random samples, computing Eq.3 can be done efficiently.

2.2. Layer-wise recursion with self-attention

The end-to-end training proposed in 2.1 is a general method that can be applied to any type of neural architecture. In this section, we describe our proposed architecture (Figure 3) which orthogonally improves generalization.

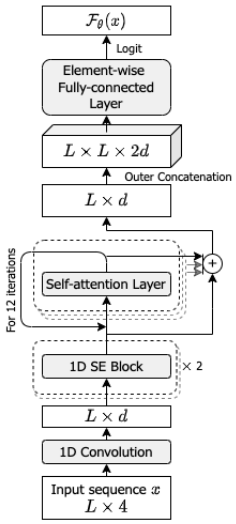

Figure 1: Neural architecture of RTfold

A key highlight of the proposed architecture is layer-wise recursion which repeatedly applies a weight-tied layer for N number of times. Without increasing the number of trainable parameters, layer-wise recursion can increase the expressiveness of the model (Schwarzschild et al., 2021; Bai et al., 2019). This recurrent inductive bias has been shown to be important for algorithmic tasks (Kaiser & Sutskever, 2015; Dehghani et al., 2018) and recently in protein folding (Jumper et al., 2021). Intuitively, RNA secondary structure prediction can be regarded as an algorithmic task of parsing RNA folding grammar. For instance, most DP-based methods implicitly model RNA folding as context-free grammars and use DP for parsing (Do et al., 2006). In this regard, layer-wise recursion can provide a useful inductive-bias for RNA secondary structure problem, in addition to being parameter efficient compared to the conventional architectures.

The layer-wise recursion block is based on transformer self-attention layer (Vaswani et al., 2017) with relative positional encoding (Huang et al., 2018). Preceding it are the Squeeze-and-Excitation blocks (Hu et al., 2018) which uses self-attention to exploit global channel-wise information. Self-attentions in these two types of blocks help capturing the long-distance interactions between bases.

Most of the blocks operate on a 1D sequence. Outer concatenation is used to project a 1D sequence into a 2D map, followed by element-wise fully-connected layer to compute the final output $F\ (x)$. This approach reduces the computational and memory complexity of *RTfold* compared to the existing deep learning models, which are primarily based on convolutional layers on 2D maps. For instance, *RTfold* can infer upto $2800nt$ sequence length on a Nvidia Titan Xp GPU, compared to $500nt$ for *SPOT-RNA* on the same GPU.

2.3. Larger training set augmented with synthetic data for pre-training

For fair evaluation, training and evaluation sets should not contain any RNA sequences that are too similar. *SPOT-RNA* removes redundant sequences between its training set (*TR0*) and validation/test datasets (*VL0*/*TS0*) using the *CD-HIT-EST* program (Fu et al., 2012) with 80% sequence similarity cut-off. However, this type of filtering is also applied within its training dataset (*TR0*), which might result in loss of valuable training data because it might be beneficial for models to learn similar sequences which could still result in different structures[6]. Instead, we construct a new training set, *bpRNA-train*, by relaxing the similarity cut-off within the training data to 99%, which expands the number of sequences to 27,671 compared to 10,814 in TR0 (details in Appendix.B).

Additionally, a large number of sequences in *bpRNA* are from only a few RNA families, so we construct a synthetic dataset to pre-train *RTfold* on more diverse data. The synthetic dataset consists of 65,000 randomly generated sequences and structure labels generated with *RNAfold*.

[5]Although FY loss is not actually computed since we only really need the gradient.

[6]Analogous to how data augmentations in computer vision can be helpful.

3. Experiments

We compare *RTFold* with three other RNA structure prediction methods: *RNAfold*, *MXfold2*, and *SPOT-RNA*[7]. *RTfold* is trained on *bpRNA-train* and *MXfold2*, *SPOT-RNA* are trained on *TR0*.

Evaluation on bpRNA *TS0*

The first evaluation set is *TS0*, as prepared in Singh et al. (2019), and table 1 summarizes average F1-score, precision, and recall[8].

Table 1: Evaluation on bpRNA *TS0*.

	F1	PRECISION	RECALL
RTFOLD	**0.687**	**0.806**	**0.693**
SPOT-RNA	0.629	0.709	0.560
MXFOLD2	0.575	0.520	0.682
RNAFOLD	0.508	0.446	0.631

RTfold outperforms the other three methods. Unlike *SPOT-RNA* which only outperforms *MXFold2* and *RNAfold* on F1-score and precision, RTfold achieves the best performance over all three metrics.

Evaluation on ArchieveII

To further evaluate the performance on an independent dataset, we test the models on *ArchieveII* (Sloma & Mathews, 2016), which contains 3,188 secondary structures carefully curated by domain experts. Both *bpRNA-train* and *TR0* contain sequences similar to those in *ArchieveII*, and these redundant sequences in *ArchieveII* are removed using *CD-HIT-EST* with 80% similarity cutoff. The filtered *ArchieveII* subset contains 787 sequences out of the original 3,188 sequences, and table 2 summarizes the results.

Table 2: Performance on ArchieveII test set

	F1	PRECISION	RECALL
RTFOLD	**0.814**	**0.891**	**0.789**
SPOT-RNA	0.608	0.572	0.680
MXFOLD2	0.643	0.634	0.675
RNAFOLD	0.517	0.495	0.566

RTfold achieves the best performance, demonstrating it can generalize well outside *bpRNA* dataset. *RTfold*, *MXfold2*, and *RNAfold* achieves higher F1-score on *ArchiveII* than on *TS0*, which could be because *bpRNA* contains more noisy data. This result could also mean that *RTfold* is more robust to training on a noisy dataset because of its inductive biases. On the other hand, F1-score for *SPOT-RNA* is lower on *ArchieveII*.

Ablation Studies

We perform ablation studies to demonstrate how the three key ideas of *RTfold* contribute to better generalization. On the *ArchieveII* subset, we compare *RTfold* with 5 setups: 1) *RTfold-1*, a neural network only model, 2) *RTfold-2*, output of a trained *RTfold-1* processed with T, 3) *RTfold-layer1*, *RTfold* without layer-wise recursion, 4) *RTfold-layer2*, *RTfold* like architecture but with 12 self-attention blocks, each with its own trainable parameters, and 5) *RTfold-TR0*, *RTfold* trained only on *TR0*.

Table 3: Ablation studies on ArchieveII test subset

	F1	PRECISION	RECALL
RTFOLD	**0.814**	0.891	0.789
RTFOLD-1	0.761	0.835	0.743
RTFOLD-2	0.792	**0.915**	0.761
RTFOLD-LAYER1	0.740	0.814	0.720
RTFOLD-LAYER2	0.789	0.830	**0.794**
RTFOLD-TR0	0.661	0.764	0.652

RTfold achieves higher F1-score than *RTfold-2* even though they both used T, demonstrating the benefit of the end-to-end training. *RTfold-layer1* has the same number of parameters as *RTfold* but lower performance without the layer-wise recursion. *RTfold-layer2*, with the same depth as *RTfold* but significantly more parameters, is outperformed by *RTfold*, demonstrating how layer-wise recursion can improve the expressiveness without additional parameters. *RTfold-TR0* achieves the worst performance without the improved training set and pre-training.

4. Conclusion and Discussions

In this work, we proposed a new framework, *RTfold*, for predicting RNA secondary structure. *RTfold* achieves the best performance on our evaluation sets, by taking advantage of: 1) end-to-end training with constrained optimization, which is a general approach that can be applied to many other structured prediction problems, 2) recurrent inductive bias, and 3) larger training set and pre-training on synthetic data. These improvements contribute to the performance, as demonstrated in the ablation studies. Additionally, *RTfold* is one of the first to move beyond the fully convolutional paradigm of the previous deep learning approaches.

Future work includes more comprehensive evaluation and comparisons, and more detailed analysis of *RTfold*. Our ablation study shows the importance of larger, good quality training set, so more efforts will also be made to construct even more diverse and informative training data.

[7]We ignore *E2Efold* due to its poor performance (Sato et al., 2021; Fu et al., 2022). We ignore *UFold* because its training data contains significant overlap with ArchieveII test set and this requires retraining the model.

[8]Threshold for SPOT-RNA is set to 0.3 (Singh et al., 2019)

References

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pp. 136–145. PMLR, 2017.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. Learning with differentiable perturbed optimizers. *arXiv preprint arXiv:2002.08676*, 2020.

Blondel, M., Martins, A. F., and Niculae, V. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

Chen, X., Li, Y., Umarov, R., Gao, X., and Song, L. Rna secondary structure prediction by learning unrolled algorithms. *arXiv preprint arXiv:2002.05810*, 2020.

Danaee, P., Rouches, M., Wiley, M., Deng, D., Huang, L., and Hendrix, D. bprna: large-scale automated annotation and analysis of rna secondary structure. *Nucleic acids research*, 46(11):5381–5394, 2018.

Dantzig, G. B., Orden, A., Wolfe, P., et al. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5 (2):183–195, 1955.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Do, C. B., Woods, D. A., and Batzoglou, S. Contrafold: Rna secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–e98, 2006.

Fu, L., Niu, B., Zhu, Z., Wu, S., and Li, W. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

Fu, L., Cao, Y., Wu, J., Peng, Q., Nie, Q., and Xie, X. Ufold: fast and accurate rna secondary structure prediction with deep learning. *Nucleic acids research*, 50(3):e14–e14, 2022.

Hazan, T., Papandreou, G., and Tarlow, D. *Perturbations, Optimization, and Statistics*. MIT Press, 2016.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kaiser, Ł. and Sutskever, I. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

Lorenz, R., Bernhart, S. H., Zu Siederdissen, C. H., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. Viennarna package 2.0. *Algorithms for molecular biology*, 6 (1):1–14, 2011.

Mathews, D. H. Revolutions in rna secondary structure prediction. *Journal of molecular biology*, 359(3):526–532, 2006.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Sato, K., Akiyama, M., and Sakakibara, Y. Rna secondary structure prediction using deep learning with thermodynamic integration. *Nature communications*, 12(1):1–9, 2021.

Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., and Goldstein, T. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Singh, J., Hanson, J., Paliwal, K., and Zhou, Y. Rna secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning. *Nature communications*, 10(1):1–13, 2019.

Sloma, M. F. and Mathews, D. H. Exact calculation of loop formation probability identifies folding motifs in rna secondary structures. *RNA*, 22(12):1808–1818, 2016.

Steeg, E. W. Neural networks, adaptive optimization, and rna secondary structure prediction. *Artificial intelligence and molecular biology*, pp. 121–160, 1993.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

A. Analysis of E2Efold predictions

We investigate how well the RNA secondary structure constraints are satisfied in the output of E2Efold. Unrolling its primal-dual method until convergence can result in arbitrarily long computational graph. Instead, E2Efold restricts the number of iterations by a hyper-parameter T, but this means the structural constraints are not guaranteed to be satisfied.

We run E2Efold on ArchiveII dataset using the trained model parameter supplied by the authors (`https://github.com/ml4bio/e2efold`). On 739 out of 3,911 samples in the dataset, E2Efold violated the RNA secondary structure constraints. All of the violated constraints are constraint 3, which restricts each base to have at most a single pair. This is expected as constraints 1 and 2 can trivially be satisfied with masking. Figure 2 shows some examples of the predicted base-pairing matrices.

B. bpRNA-train dataset

We motivate our proposed **bpRNA-train** training set by first describing the construction and limitations of *bpRNA-TR0* training set.

bpRNA-TR0 consist of 10,814 RNA structures from *bpRNA-1m* (Danaee et al., 2018) and it is used to train SPOT-RNA and MXfold. *bpRNA-1m* has 102,318 structures aggregated from seven RNA structure datasets and many of the RNA sequences for these structures are highly similar. To filter out redundant sequences, CD-HIT-EST program (Fu et al., 2012) with 80% or higher sequence similarity cutoff is used. Then, the remaining sequences are randomly split into training (*TR0*), validation (*VL0*), and testing (*TS0*) subsets.

In order to properly evaluate how well the trained models generalize, it is important that training set (e.g. *TR0*) do not contain any samples similar to the ones in evaluation sets (e.g. *VL0*, *TS0*). Removing redundant sequences between the two can be achieved by running a sequence alignment program and filtering out sequences above a similarity score threshold. However, when this is applied on training set, valuable data might be discarded because: 1) sometimes similar sequences have very different structures, and 2) similar sequences have subtle differences in structures which might provide valuable information during training. For instance, when clustering similar sequences using CD-HIT-EST using the 80% similarity cut-off, we often observe clusters with very distinct structures, like shown in Figure 3 (the left three vs. right two). Even for similar structures, it might be important to train deep learning model on how differences in sequences result in subtle changes in structures, which essentially acts like data augmentation commonly used in training deep learning models. Additionally, structures in bpRNA-1m are considered to be noisy as many of them are obtained through comparative sequence analysis, so without any knowledge on which one being the closest to the true structures, it might be best to keep all of them.

To mitigate the aforementioned problem, we construct a new training set, which we call **bpRNA-train**, by relaxing the similarity cut-offs to 99%. In this work, we keep the same VL0 and TS0 for the validation and testing to carry out fair comparison to the existing methods. To ensure no sequences in the new training set is similar to ones in the evaluation sets, we use CD-HIT-EST to filter out sequences with 80% or more similarity to sequences in VL0 and TS0. Then, we use CD-HIT-EST on the remaining set with 99% similarity threshold. With the relaxed threshold, a few clusters of many similar sequences can heavily bias the dataset: roughly a quarter of the sequences in the dataset come from one percent of clusters. Thus, we limit the size of each cluster to 10. The resulting **bpRNA-train** consists of 27,671 sequences.

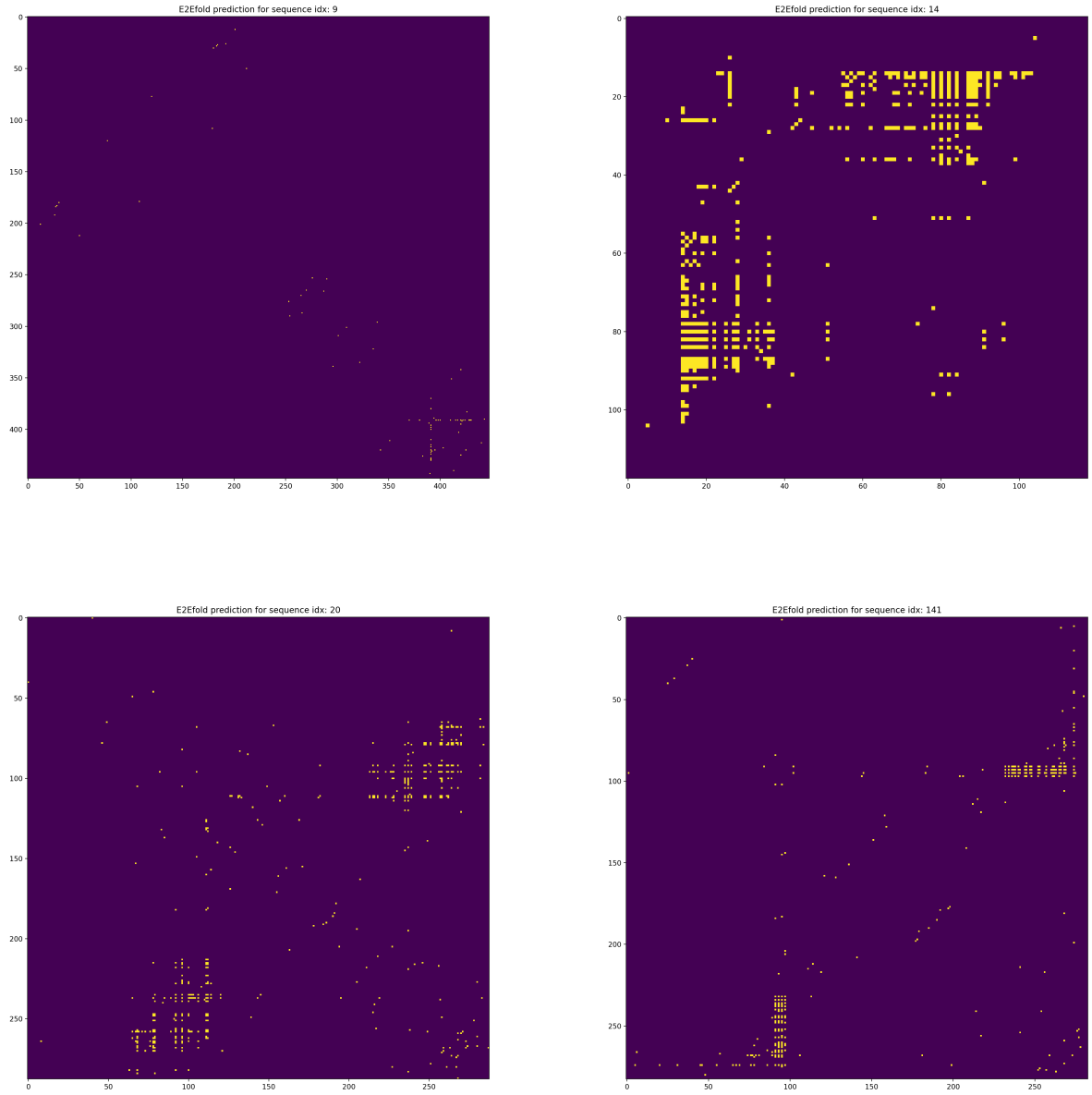

Figure 2: Examples of E2Efold predictions on ArchieveII violating the structural constraints.