# LEARNING TO DESIGN RNA

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

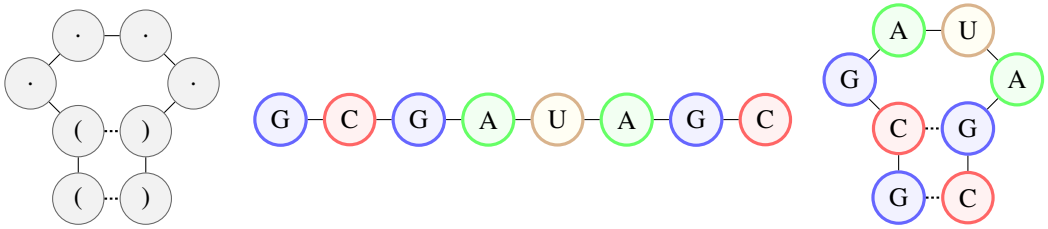
Designing RNA molecules has garnered recent interest in medicine, synthetic biology, biotechnology and bioinformatics since many functional RNA molecules were shown to be involved in regulatory processes for transcription, epigenetics and translation. Since an RNA's function depends on its structural properties, the *RNA Design* problem is to find an RNA sequence that folds into a specified secondary structure. Here, we propose a new algorithm for the *RNA Design* problem, dubbed *LEARNA*. *LEARNA* uses deep reinforcement learning to train a policy network to sequentially design an entire RNA sequence given a specified secondary target structure. By meta-learning across 8000 different RNA target structures for one hour on 20 cores, our extension *Meta-LEARNA* constructs an *RNA Design* policy that can be applied out of the box to solve novel RNA target structures. Methodologically, for what we believe to be the first time, we jointly optimize over a rich space of neural architectures for the policy network, the hyperparameters of the training procedure and the formulation of the decision process. Comprehensive empirical results on two widely-used RNA secondary structure design benchmarks, as well as a third one that we introduce, show that our approach achieves new state-of-the-art performance on all benchmarks while also being orders of magnitudes faster in reaching the previous state-of-the-art performance. In an ablation study, we analyze the importance of our method's different components.

## 1 INTRODUCTION

RNA is one of the major classes of information-carrying biopolymers in the cells of living organisms. Recent studies revealed a key role of functional non-protein-coding RNAs (ncRNAs) in regulatory processes and transcription control, which have also been connected to certain diseases like Parkinson's disease and Alzheimer's disease (ENCODE Project Consortium and others, 2004; Gstir et al., 2014; Kaushik et al., 2018). Functional ncRNAs are involved in the modulation of epigenetic marks, altering of messenger RNA (mRNA) stability, mRNA translation, alternative splicing, signal transduction and scaffolding of large macromolecular complexes (Vandivier et al., 2016). Therefore, engineering of ncRNA molecules is of growing importance with applications ranging from biotechnology and medicine to synthetic biology (Delebecque et al., 2011; 2012; Guo et al., 2010; Terns & Terns, 2014; Meyer et al., 2015). In fact, successful attempts to create functional RNA sequences *in vitro* and *in vivo* have been reported (Dotu et al., 2014; Wachsmuth et al., 2013).

At its most basic structural form, RNA is a sequence of the four nucleotides *Adenine (A)*, *Guanine (G)*, *Cytosine (C)* and *Uracile (U)*. This nucleotide sequence is called the *RNA sequence*, or primary structure. While the RNA sequence serves as the blueprint, the functional structure of the RNA molecule is determined by the folding translating the RNA sequence into its 3D tertiary structure. The intrinsic thermodynamic properties of the sequence dictate the resulting fold. The hydrogen bonds formed between two corresponding nucleotides constitute one of the driving forces in the thermodynamic model and influence the tertiary structure heavily. The structure that encompasses these hydrogen bonds is commonly referred to as the secondary structure of RNA. Many algorithms for RNA tertiary structure design directly work on RNA secondary structures (Kerpedjiev et al., 2015; Zhao et al., 2012; Reinharz et al., 2012). Therefore, fast and accurate algorithms for RNA secondary structure design could advance the current state of the art in RNA engineering.

The problem of finding an RNA sequence that folds into a desired secondary structure is known as the *RNA Design* problem or *RNA inverse folding* (Hofacker et al., 1994). Most algorithms for *RNA Design* focus on search strategies that start with an initial nucleotide sequence and modify it to find a




(a) Target structure $\omega$ (b) RNA sequence $\phi \in N^{|\omega|}$ (c) Fold $\mathcal{F}(\phi)$

Figure 1: Illustration of the *RNA Design* problem using a folding algorithm $\mathcal{F}$ and the dot-bracket notation. Given the desired RNA secondary structure and its dot-bracket notation (a), the task is to design an RNA sequence (b) that folds into the desired secondary structure (c).

solution for the given secondary structure (Hofacker et al., 1994; Andronescu et al., 2004; Taneda, 2011; Esmaili-Taheri et al., 2014; Eastman et al., 2018). In contrast, in this paper we describe a novel deep reinforcement learning (RL) approach to this problem. Our contributions are as follows:

- We describe *LEARNA*, a deep RL algorithm for *RNA Design*. *LEARNA* trains a policy network that, given a target secondary structure, can be rolled out to sequentially predict the entire RNA sequence. After generating an RNA sequence, our approach folds this sequence, locally adapts it, and uses the distance of the resulting structure to the target structure as an error signal for the RL agent.
- We describe *Meta-LEARNA*, a version of *LEARNA* that learns a single policy across many *RNA Design* problems directly applicable to new *RNA Design* problems. Specifically, it learns a conditional generative model from which we can sample candidate RNA sequences for a given RNA target structure and solves many problems with the first sample.
- Validation in *RNA Design* literature is often done using undisclosed data sources (Eastman et al., 2018; Yang et al., 2017) and previous benchmarks do not have a training split associated with them (Taneda, 2011; Anderson-Lee et al., 2016; Kleinkauf et al., 2015). Here, we introduce a new benchmark dataset with an explicit training, validation and test split.
- We jointly optimize the architecture of the policy network together with training hyper-parameters, and the state representation. By assessing the importance of these choices, we show that this is essential to achieve best results. To the best of our knowledge, this is the first application of neural architecture search (NAS) to RL, the first application of NAS to meta-learning, and the first time NAS is used to choose the best combination of convolutional and recurrent layers.
- A comprehensive empirical analysis shows that our approach achieves new state-of-the-art performance on the two most commonly used *RNA Design* benchmark datasets: Rfam-Taneda (following Taneda (2011)) and Eterna100 (following Anderson-Lee et al. (2016)), as well as on the test split of our new benchmark. Furthermore, *Meta-LEARNA* achieves the results of the previous state-of-the-art approaches $26\times$, $450\times$ and $4\times$ faster.

## 2 THE RNA DESIGN PROBLEM

The *RNA Design* problem aims to find an inverse mapping for a given RNA folding algorithm $\mathcal{F}$, which maps from an RNA sequence to a representation of its secondary structure:

**Definition 1** (RNA Design)**.** *Given a folding algorithm $\mathcal{F}$ and a target RNA secondary structure $\omega$, the RNA Design* problem *is to find an RNA sequence $\phi \in N^{|\omega|} = \{A, G, C, U\}^{|\omega|}$ that satisfies $\omega = \mathcal{F}(\phi)$.*

In this paper, we employ the most common folding algorithm: the *Zuker algorithm* (Zuker & Stiegler, 1981; Zuker & Sankoff, 1984), which uses a thermodynamic model to minimize the free energy to find the most stable conformation of the RNA secondary structure. We note, however, that our approach is not limited to it and would directly apply for any other RNA folding algorithm.

RNA secondary structures are often represented using the dot-bracket notation, where dots stand for unbound sites and nucleotides connected by a hydrogen bond are marked by opening and closing brackets.[1] Figure 1 illustrates the *RNA Design* problem and the dot-bracket notation.

Most algorithms for *RNA Design* employ a structural loss function $L_\omega(\mathcal{F}(\phi))$ to quantify the difference between the target structure $\omega$ and the structure resulting from folding an RNA sequence $\phi$. The minimizer of this loss corresponds to a solution to the *RNA Design* problem for a specific given target structure $\omega$:

$$\phi^* \in \underset{\phi \in N^{|\omega|}}{\arg\min}\, L_\omega(\mathcal{F}(\phi)) \qquad . \tag{1}$$

A common loss function, which we also employ in this work, is the Hamming distance (Hamming, 1950) between two structures.

We note that multiple RNA sequences may fold to the same secondary structure, such that the *RNA Design* problem does not generally have a unique solution; one could distinguish between solutions by preferring more stable folds, targeting a specific GC content, or satisfying other constraints; all of these could be incorporated into the loss function being optimized.

## 3 Related Work

Most algorithms targeting the *RNA Design* problem are either local or global algorithms. Local approaches commonly operate on a single sequence and try to find a solution by changing a small number of nucleotides at a time with the loss function guiding the search (*RNAInverse* (Hofacker et al., 1994), *RNA-SSD* (Andronescu et al., 2004), *INFO-RNA* (Busch & Backofen, 2006), *NUPACK* (Dirks & Pierce, 2004; Zadeh et al., 2010), *ERD* (Esmaili-Taheri et al., 2014) and the approach by Eastman et al. (2018)). Global methods, on the other hand, either have a large number of candidates being manipulated, or model a global distribution from which samples are generated (*MODENA* (Taneda, 2011), *antaRNA* (Kleinkauf et al., 2015) and *MCTS-RNA* (Yang et al., 2017)). A more detailed review can be found in Churkin et al. (2017).

Very recently, another, less general direction to RNA Design imposed a prior of human knowledge onto the agent (Shi et al., 2018). In this approach, a large ensemble of models is trained on human solutions of manually designed RNA Design problems, incorporating deep domain-knowledge guiding the agents behaviour, followed by an adaptive walk procedure using human strategies for refinement of the candidate solution. Totalized results over all models of the ensemble were reported on the Eterna100 (Anderson-Lee et al., 2016) dataset, solely consisting of manually designed RNA Design problems, that we also report on here. Although it showed good results in this one benchmark, human solutions and strategies were not available for our further benchmarks derived from natural RNA structures and due to exceeding computational costs we could not include this work in our comparison.

The work by Bello et al. (2016) heavily influenced our work. In it, the authors apply RL to combinatorial problems, namely the Traveling Salesman Problem. The agent proposes complete solutions rather than manipulating an existing one, and is trained using an episodic reward, in this case the negative tour length. Inspired by this work, we propose to frame the *RNA Design* problem as a RL problem where each candidate solution is designed from scratch. In our approach, the agent predicts which nucleotide to place next into the sequence, learning to design RNA end-to-end.

This is in stark contrast to the recent work by Eastman et al. (2018) carried out in parallel to and independently from ours. Eastman et al. used RL to perform a local search starting from a randomly initialized sequence. The RL agent applies local modifications to design a solution that folds into the desired target structure. The current sequence constitutes the state and each action represents changing an unpaired nucleotide or a pair of nucleotides. After each action the current sequence is evaluated utilizing the Zuker algorithm and the agent only receives a nonzero reward signal once it finds a correct sequence. The agent's policy is a convolutional neural network pre-trained on fixed-length, randomly generated sequences. In the remainder of the paper, we refer to this approach as RL-LS, since the RL agent performs a local search.

[1] There are also other notations (Shapiro, 1988; Fontana et al., 1993); our approach would also apply to these.

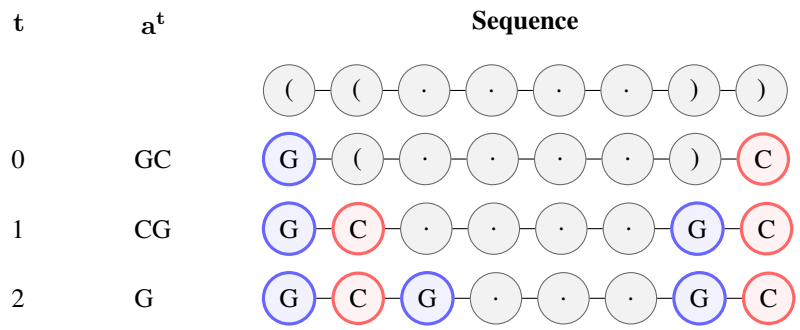



Figure 2: Illustration of the design of a candidate solution. The agent builds the candidate solution sequentially by choosing actions to place nucleotides.

Variational autoencoders, generative adversarial networks and reinforcement learning have recently shown promising results in protein design and other related problems in matter engineering (Gupta & Zou, 2018; Greener et al., 2018; Olivecrona et al., 2017). For a detailed review on machine learning approaches in the field of matter engineering, we refer to Sanchez-Lengeling & Aspuru-Guzik (2018). In recent work related to RNA design, a convolutional neural network based auto-encoder with additional supervised fine tuning was proposed to score on-target and off-target efficacy of the guide RNAs for the genome editing technique CRISPR/CAS9 (Chuai et al., 2018). This automated efficacy scoring could inform future guide RNA design endeavours for the CRISPR/CAS9 technique. Our work adds evidence for the competitiveness of machine learning methods in this general problem domain.

## 4 Learning to Design RNA

In this section we describe our novel generative approach for the *RNA Design* problem based on reinforcement learning. We first formulate *RNA Design* as a decision process and then propose several strategies to yield agents that learn to design RNA end-to-end.

### 4.1 A Decision Process Modelling RNA Design

We propose to model the *RNA Design* problem with respect to a given target structure $\omega$ as the undiscounted decision process $D_\omega := (\mathcal{S},\, \mathcal{A},\, \mathcal{R}_\omega,\, \mathcal{P}_\omega)$; its components (the state space $\mathcal{S}$, the action space $\mathcal{A}$, the reward function $\mathcal{R}_\omega$ and the transition function $\mathcal{P}_\omega$) are specified in the paragraphs below. The *RNA Design* problem is defined with respect to a folding algorithm, which we denote as $\mathcal{F}(\cdot)$; we also denote the set of dot-bracket encoded RNA secondary structures with $\Omega$ .

**Action space** In each episode an agent has the task to design an RNA sequence that folds into the given $\omega \in \Omega$. To design a candidate solution $\phi \in N^{|\omega|}$ the agent places nucleotides by choosing an action $a^t$ at each time step $t$. For unpaired sites, $a^t$ corresponds to one of the four RNA nucleotides (G, C, A or U); for paired sites, two nucleotides are placed simultaneously at time step $t$. In our formulation, these two nucleotides correspond to one of the Watson-Crick base pairs (GC, CG, AU, or UA). The action space at time step $t$ then is

$$\mathcal{A} := \{0,\, 1,\, 2,\, 3\} \equiv \begin{cases} \{A,\, G,\, C,\, U\} & \text{for } \mathcal{C}_\omega(t) = . \quad \text{["dot"]} \\ \{GC,\, CG,\, AU,\, UA\} & \text{for } \mathcal{C}_\omega(t) = ( \quad \text{["opening bracket"]} \end{cases}, \tag{2}$$

where $\mathcal{C}_\omega(t)$ is the t-th character of the target structure $\omega$, ignoring closing brackets as these sites are assigned nucleotides when encountering the corresponding opening bracket. See Figure 2 for an illustration.

**State space** The agent chooses an action $a^t$ based on the state $s^t$ provided by the environment. We formulated states to provide local information to the agent. For this we set $s^t$ to the $(2\kappa + 1)$-gram centered around the t-th site of the target structure $\omega$, where $\kappa$ is a hyperparameter we dub the *state*

*radius*. To be able to construct this centered n-gram at all sites we introduced $\kappa$ padding characters at the start and the end of the target structure. Formally, the state space can then be written as

$$\mathcal{S} := \{0,\, 1,\, 2,\, 3\}^{2\kappa+1} \equiv (\mathcal{B} \cup \{\text{padding}\})^{2\kappa+1} \qquad , \tag{3}$$

where $\mathcal{B}$ is the set of symbols in the dot-bracket notation: a dot, an opening and a closing bracket.

**Transition Function** Since at each time step $t$ the state $s^t$ is set to a fixed $(2\kappa + 1)$-gram, the transition function $\mathcal{P}_\omega$ is deterministic and defined accordingly.

**Reward Function** At the terminal time step $T$ the agent has assigned nucleotides to all sites of the candidate solution $\phi$ and the environment generates the (only non-zero) reward $\mathcal{R}_\omega^T$. This reward is based on the loss function

$$L_\omega(\mathcal{F}(\phi)) := \left( \frac{\mathrm{d_H}(\mathcal{F}(\phi),\, \omega)}{|\omega|} \right)^\alpha \qquad , \tag{4}$$

where $\mathrm{d_H}(\cdot, \cdot)$ is the *Hamming distance* and $\alpha > 1$ is a hyperparameter to shape the objective function. We set $\mathcal{R}_\omega^T = -L_\omega(\mathcal{F}(\phi))$ to solve the optimization problem in Equation 1. To increase sample efficiency and boost performance of the stochastic RL agent, we include a local improvement step as follows: If $\mathrm{d_H}(\mathcal{F}(\phi),\, \omega) < \xi$, where $\xi$ is a hyperparameter, we search through *neighboring* primary sequences by exhaustively trying all combinations for the mismatched sites, returning the minimum hamming distance observed. Pseudocode for computing $\mathcal{R}_\omega^T$ can be found in Appendix A. We set $\xi = 5$, which corresponds to at most 256 neighboring sequences.

### 4.2 Obtaining Policies for RNA Design

We use deep reinforcement learning to optimize policy networks $\pi^\theta$ with parameters $\theta$, that define a posterior distribution on the action space $\mathcal{A}$ conditional on the state space $\mathcal{S}$. Our policy networks consist of an embedding layer for the input state and a deep neural network; this neural network optionally contains convolutional, recurrent and fully-connected layers, and its precise architecture is jointly optimized together with the hyperparameters as described in Section 5. We propose several strategies to learn the parameters $\theta$ of a given policy network as detailed below.

**LEARNA** The LEARNA strategy learns to design a sequence for the target structure $\omega$ in an online fashion, from scratch. The parameters $\theta$ are randomly initialized before the agent episodically interacts with the decision process $\mathcal{D}_\omega$. For updating the parameters we use the policy gradient method PPO (Schulman et al., 2017), which was recently successfully applied to several other applications of machine learning (Heess et al., 2017; Bansal et al., 2018; Zoph et al., 2018).

**Meta-LEARNA** Meta-LEARNA uses a meta-learning approach (Lemke et al., 2015) that views the RNA design of the target structures in the training set $\Omega_{train}$ as tasks and learns to transfer knowledge across them. Each of the target structures $\omega_i \in \Omega_{train}$ defines a different decision process $\mathcal{D}_{\omega_i}$, and we train a single policy network on them using asynchronous parallel PPO updates. Once the training is finished, the parameters $\theta$ are fixed and $\pi^\theta$ can be applied to the decision process $\mathcal{D}_\omega$ by sampling from the learned generative model.

**Meta-LEARNA-Adapt** Meta-LEARNA-Adapt combines the previous two strategies: First, we run Meta-LEARNA to train parameters $\theta$ in an offline training phase on $\Omega_{train}$. However, to work on target structure $\omega$, the policy is not fixed but is only used to initialize LEARNA running on the decision process $\mathcal{D}_\omega$.

## 5 Joint Architecture and Hyperparameter Search

One problem of current reinforcement learning methods is that their performance can be very sensitive to choices of neural architectures, training hyperparameters, and the formulation of the problem as a decision process (Henderson et al., 2017). Therefore, we propose to use techniques from the field of automatic machine learning, in particular an efficient Bayesian optimization method (Falkner et al., 2018) to address the problems of neural architecture search

(NAS) (Zoph & Le, 2017; Elsken et al., 2018) and hyperparameter optimization as a joint optimization problem. To automatically select the best neural architecture based on data, we define a search space that includes both elements of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) and let the optimizer choose the best combination of the two. We believe this to be novel, as – to the best of our knowledge – NAS has not previously been applied to RL nor meta-learning.

In this section, we present our representation of the search space and describe our approach to optimizing performance.

### 5.1 Search Space

Our search space has three components described in the following: choices about the policy network's neural architecture, environment parameters (including the representation of the state and the reward), and training hyperparameters.

**Neural Architecture** We construct our architecture as follows: (1) the dot bracket representation of the state is either binary encoded (distinguishing between paired and unpaired sites) or processed by an embedding layer that converts the symbol-based representation into a learnable numerical one for each site. Then, (2) an optional CNN with at most two layers can be applied to the state, followed by (3) an optional LSTM with at most two layers. As the final stage, we always add (4) a shallow fully-connected network with one or two layers, which outputs the distribution over actions. This parameterization covers a broad range of possible architectures while keeping the dimensionality of the search space relatively modest.

**Environment Parameters** Since our ultimate goal is not to solve a specific decision process (DP), but to use the best DP for solving our problem, we also optimize parameters concerning the state representation and the reward: We optimize the number of sites symmetrically centered around the current one, called the state space radius $\kappa$ (see Section 4.1) and the shape of the reward via the parameter $\alpha$ (see Equation 4).

**Training Hyperparameters** Since the performance of neural networks strongly depends on the training hyperparameters governing optimization and regularization, we optimized some of the parameters of PPO, which we employ for training the network: the learning rate, the batch size, and strength of the entropy regularization.

Overall, these design choices yield a 14-dimensional search space comprising mostly integer variables. The complete list of parameters, their types, ranges, and the priors we used over them can be found in Appendix E. We used almost identical search spaces for *LEARNA* and *Meta-LEARNA*, but adapted the ranges for the learning rate and the entropy regularization slightly based on preliminary experiments. Please refer to Tables 3 and 4 in Appendix E for more details.

### 5.2 Search Procedure

We now have to address the problem of optimizing in a search space consisting of continuous and integer parameters, where a single function evaluation is rather costly. We chose the recently-proposed optimizer BOHB (Falkner et al., 2018) to find good configurations, because it can handle these mixed spaces, utilize parallel resources, and additionally can exploit cheap approximations of the objective functions to speed up the optimization. These so-called low-fidelity approximations can be achieved in numerous ways, e.g., limiting the training time, the number of independent repetitions of the evaluations, or using only fractions of the data. In our setting, we decided to limit the wall-clock time for training (*Meta-LEARNA*) or the evaluations (*LEARNA*). For a detailed description of the limits, we refer to Appendix E.

**Datasets** Since our approach involves training (*Meta-LEARNA*) and to properly optimize the listed design choices without overfitting, we needed a designated training and validation dataset. However, previous benchmarks used in the RNA Design literature do not provide a train/validation/test split. This led us to introduce a new benchmark with an explicit train/validation/test split based on the Rfam database version 13.0 (Kalvari et al., 2017). We employed the protocol described in Appendix

C to yield our training set Rfam-Learn-Train, our validation set Rfam-Learn-Validation and our test set Rfam-Learn-Test. All datasets we used for this paper are listed in detail in Appendix D.

**Budgets** We optimize each of our approaches using only our own training and validation sets for Rfam-Learn, but due to the very different standard evaluation timeouts of the different benchmarks (10 minutes for Rfam and 1 day for Eterna, see Appendix D), we created two versions of *LEARNA*: (1) *LEARNA-10min*, which is optimized for achieving strong performance in 10 minutes (on one core per sequence) and (2) *LEARNA-30min*, which is optimized for achieving strong performance within 30 minutes (on one core per sequence; we chose 30 minutes, rather than 1 day, to keep the optimization manageable within our compute resources). *LEARNA-10min* is used on the Rfam-Taneda dataset and *LEARNA-30min* is applied to the other two datasets with the following modification: every 30 minutes, the policy network and all internal variables of PPO are reinitialized, i.e., we perform a restart of the algorithm to overcome occasional stagnation of PPO. Finally, our meta learning approach *Meta-LEARNA* is optimized to achieve strong performance when run for one hour on twenty cores (with an internal budget of 3 minutes on one core per sequence).

**Objective** Despite the fact that RL is known to often yield noisy or unreliable outcomes in single optimization runs (Henderson et al., 2017), we actively decided to only use a single optimization run and a single validation set for each configuration to keep the optimization manageable. To counteract the problems associated with single (potentially) noisy observations, we studied three different loss functions for the hyperparameter optimization: (a) The number of unsolved sequences, (b) the sum of mean distances, and (c) the sum of minimum distances to the target structure. While we ultimately seek to minimize (a), this is a rather noisy and discrete quantity. In preliminary experiments, optimizing (b) turned out to be inferior to (c), presumably because the former punishes exploration by the agent more, while the latter rewards ultimately getting close to the solution. Therefore, we used (c).

# 6 EXPERIMENTS

We evaluate our approaches against state-of-the-art methods and perform an ablation study to assess the importance of our method's components. We report results on two established benchmarks from the literature and on our own benchmark. Full information on the three benchmarks is given in Appendix C. For each benchmark, we followed its standard evaluation protocol, performing multiple attempts (in the following referred to as evaluation runs) with a fixed time limit for each target structure. For each benchmark, we report the accumulated number of solved targets across all evaluation runs and provide means and confidence bounds for all experiments. All methods were compared on the same hardware, each allowed one core per evaluation of a single target structure. Implementation details are listed in Appendix B.

## 6.1 COMPARATIVE STUDY

The results of our comparative study are given in Table 1 and Figure 3; we discuss them in the following.

**Eterna100** Solving up to 65% (*Meta-LEARNA*) of the target structures, all variants of *LEARNA* achieve new state-of-the-art results on the Eterna100 benchmark. Additionally, *Meta-LEARNA* only needs about 90 seconds to be on par with the final performance of any other method and achieves new state-of-the-art results in less than 3 minutes. This performance is stable through all of the 5 evaluation runs performed. *Meta-LEARNA* further achieves new state-of-the-art performance in each single evaluation run (see Appendix G). *Meta-LEARNA-Adapt* also performs very well, but actually performs somewhat worse than *Meta-LEARNA*; this seems counter-intuitive at first, but a closer analysis showed that this performance loss is simply caused by the overhead associated with the weight updates (required in the adaptive policy of *Meta-LEARNA-Adapt*, but not in the static policy of *Meta-LEARNA-Adapt*), which result in *Meta-LEARNA-Adapt* only being able to perform 7-10 times fewer evaluations than *Meta-LEARNA* performs in the same time.

**Rfam-Taneda** Concerning the Rfam-Taneda benchmark, *Meta-LEARNA* and *LEARNA* are on par with the current state-of-the-art results of *MCTS-RNA*, but remarkably, *Meta-LEARNA* only needs

Table 1: Summary of the final results for *RNAInverse*, *MCTS-RNA*, *RL-LS*, *antaRNA*, *LEARNA*, and *Meta-LEARNA* regarding the total number of solved target structures on the three benchmarks in percent. A target structure counts as solved if a solution was found in any of the evaluation runs on the specific dataset.

| METHOD | SOLVED SEQUENCES [%] | | |
|---|---|---|---|
| | ETERNA100 | RFAM-TANEDA | RFAM-LEARN-TEST |
| MCTS-RNA | 57 | 79 | 97 |
| ANTARNA | 58 | 66 | **100** |
| RL-LS | 59 | 62 | 62 |
| RNAINVERSE | 60 | 59 | 95 |
| LEARNA-10MIN | - | 79 | 95 |
| LEARNA-30MIN | 63 | - | 97 |
| META-LEARNA-ADAPT | 64 | **83** | 98 |
| META-LEARNA | **65** | 79 | **100** |

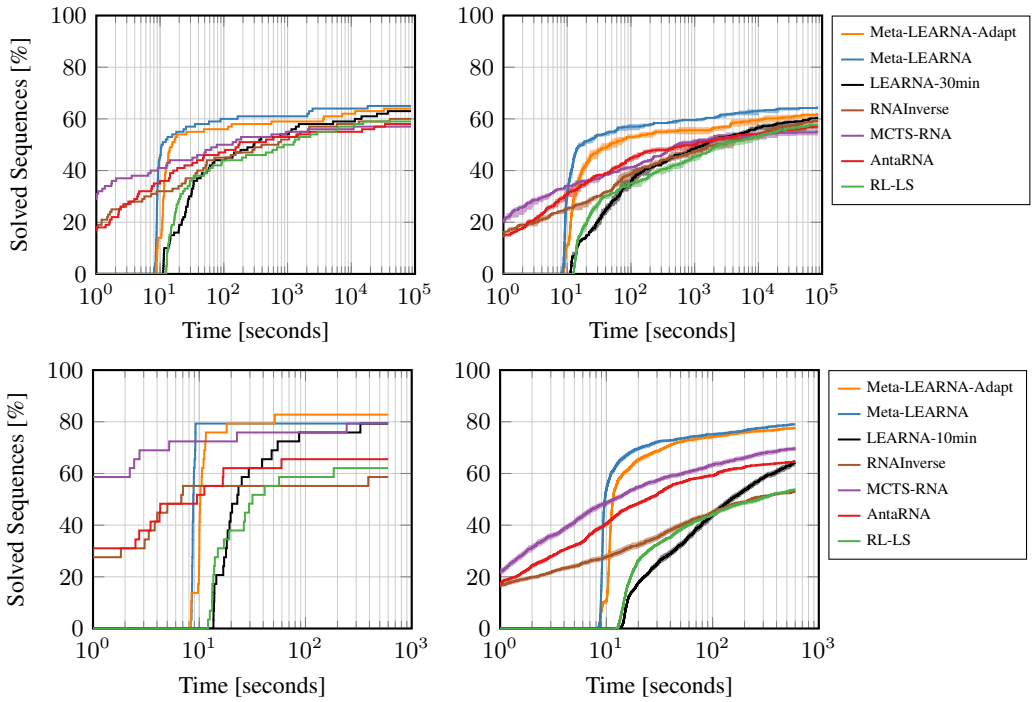



Figure 3: Comparison of all methods on the Eterna100 (top) and the Rfam-Taneda benchmark (bottom). On the left we show the total number of target structures that were solved in at least one evaluation run across the time spent on each particular target structure, while the right panels show the average number of solved target structures and the confidence interval.

10 seconds to achieve this performance. *Meta-LEARNA-Adapt* achieves new state-of-the-art results after 1 minute (see also Appendix G), solving 83% of the target structures.

**Rfam-Learn** The results on our new Rfam-Learn benchmark are shown in Figure 8 in Appendix F. Only *Meta-LEARNA* and *antaRNA* were able to solve all of the target structures in 1 hour; 5 minutes and 20 minutes respectively. Except for *RL-LS*, all algorithms could solve at least 95% of the target structures within the one hour time limit.

In summary, our novel deep reinforcement learning algorithm achieved the best performance on all of the three benchmarks while being much faster than all other algorithms. Our meta-learning approach *Meta-LEARNA* learned a representation of the dynamics underlying *RNA Design* and is capable of transferring this knowledge for solving new RNA Design problems, independent of the length of the

target structure. As our additional analysis in Appendix J shows, it also scales better with sequence length than existing approaches. For a detailed list of the performance of all algorithms on specific target structures, we refer to the detail tables in Appendix K.

### 6.2 Ablation study and Parameter Importance

To study the influence of the different components and parameters on the performance of our approach, we performed an ablation study and a functional analysis of variance (fANOVA) (Hooker, 2007; Hutter et al., 2014). For the ablation, we excluded either the adaptation option, the local improvement step or the restart option. We observed a boost in performance for all variants of our approach from the local improvement step, while the restart option stabilizes the single evaluation runs (see Figure 9 in Appendix H). The fANOVA results highlight the importance of a few parameters from all three components of the search space mentioned in Section 5. This emphasizes the importance of the joint optimization of the policy network's architecture, the environment parameters and the training hyperparameters. All results and a more detailed discussion of our ablation study and the fANOVA results can be found in Appendix H and I, respectively.

## 7 Conclusion

We proposed the deep reinforcement learning algorithm *LEARNA* for the *RNA Design* problem to sequentially construct candidate solutions in an end-to-end fashion. By pre-training on a large corpus of biological sequences, a local improvement step to aid the agent, and extensive architecture and hyperparameter optimization, we arrived at *Meta-LEARNA*, a ready-to-use agent that achieves state-of-the-art results on the Eterna100 benchmark (Anderson-Lee et al., 2016) and our own new dataset. By continuing training, dubbed *Meta-LEARNA-Adapt*, we can also improve over all previous results on the Rfam benchmark proposed by Taneda (2011). Our ablation study shows the importance of all components, suggesting that RL with an additional local improvement step can solve the *RNA Design* problem efficiently. Code and data to reproduce our results will be made publicly available after acceptance of this paper.

## References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Jeff Anderson-Lee, Eli Fisker, Vineet Kosaraju, Michelle Wu, Justin Kong, Jeehyung Lee, Minjae Lee, Mathew Zada, Adrien Treuille, and Rhiju Das. Principles for predicting RNA secondary structure design difficulty. *Journal of molecular biology*, 428(5):748–757, 2016.

Mirela Andronescu, Anthony P. Fejes, Frank Hutter, Holger H. Hoos, and Anne Condon. A New Algorithm for RNA Secondary Structure Design. *Journal of Molecular Biology*, 336(3): 607–624, 2004. ISSN 0022-2836. doi: https://doi.org/10.1016/j.jmb.2003.12.041. URL `http://www.sciencedirect.com/science/article/pii/S0022283603015596`.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*, 2018. Published online: `iclr.cc`.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Anke Busch and Rolf Backofen. INFO-RNA—a fast approach to inverse RNA folding. *Bioinformatics*, 22(15):1823–1831, 2006. doi: 10.1093/bioinformatics/btl194. URL http://dx.doi.org/10.1093/bioinformatics/btl194.

Guohui Chuai, Hanhui Ma, Jifang Yan, Ming Chen, Nanfang Hong, Dongyu Xue, Chi Zhou, Chenyu Zhu, Ke Chen, Bin Duan, Feng Gu, Sheng Qu, Deshuang Huang, Jia Wei, and Qi Liu. Deepcrispr: optimized crispr guide rna design by deep learning. *Genome Biology*, 19(1):80, Jun 2018. ISSN 1474-760X. doi: 10.1186/s13059-018-1459-4. URL https://doi.org/10.1186/s13059-018-1459-4.

Alexander Churkin, Matan Drory Retwitzer, Vladimir Reinharz, Yann Ponty, Jérôme Waldispühl, and Danny Barash. Design of rnas: comparing programs for inverse rna folding. *Briefings in bioinformatics*, 19(2):350–358, 2017.

Camille J. Delebecque, Ariel B. Lindner, Pamela A. Silver, and Faisal A. Aldaye. Organization of Intracellular Reactions with Rationally Designed RNA Assemblies. *Science*, 333(6041):470–474, 2011. ISSN 0036-8075. doi: 10.1126/science.1206938. URL http://science.sciencemag.org/content/333/6041/470.

CJ Delebecque, PA Silver, and AB Lindner. Designing and using RNA scaffolds to assemble proteins in vivo. *Nature Protocols*, 7(10):1797–807, 2012. doi: 10.1038/nprot.2012.102.

Robert M. Dirks and Niles A. Pierce. An Algorithm for Computing Nucleic Acid Base-Pairing Probabilities Including Pseudoknots. *Journal of Computational Chemistry*, 25(10):295—-1304, 2004. doi: 10.1002/jcc.20057.

Ivan Dotu, Juan Antonio Garcia-Martin, Betty L. Slinger, Vinodh Mechery, Michelle M. Meyer, and Peter Clote. Complete RNA inverse folding: computational design of functional hammerhead ribozymes. *Nucleic Acids Research*, 42(18):11752–11762, 2014. doi: 10.1093/nar/gku740. URL http://dx.doi.org/10.1093/nar/gku740.

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

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *ArXiv e-prints*, August 2018.

ENCODE Project Consortium and others. The ENCODE (encyclopedia of DNA elements) project. *Science*, 306(5696):636–640, 2004.

Ali Esmaili-Taheri, Mohammad Ganjtabesh, and Morteza Mohammad-Noori. Evolutionary solution for the RNA design problem. *Bioinformatics*, 30(9):1250–1258, 2014. doi: 10.1093/bioinformatics/btu001. URL http://dx.doi.org/10.1093/bioinformatics/btu001.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pp. 1436–1445, July 2018.

Walter Fontana, Peter F. Stadler, Erich G. Bornberg-Bauer, Thomas Griesmacher, Ivo L. Hofacker, Manfred Tacker, Pedro Tarazona, Edward D. Weinberger, and Peter Schuster. Rna folding and combinatory landscapes. *Phys. Rev. E*, 47:2083–2099, Mar 1993. doi: 10.1103/PhysRevE.47.2083. URL https://link.aps.org/doi/10.1103/PhysRevE.47.2083.

Joe G Greener, Lewis Moffat, and David T Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific reports*, 8(1):16189, 2018.

Ronald Gstir, Simon Schafferer, Marcel Scheideler, Matthias Misslinger, Matthias Griehl, Nina Daschil, Christian Humpel, Gerald J. Obermair, Claudia Schmuckermair, Joerg Striessnig, Bernhard E. Flucher, and Alexander Hüttenhofer. Generation of a neuro-specific microarray reveals novel differentially expressed noncoding RNAs in mouse models for neurodegenerative diseases. *RNA*, 20(12):1929–1943, 2014.

Peixuan Guo, Oana Coban, Nicholas M. Snead, Joe Trebley, Steve Hoeprich, Songchuan Guo, and Yi Shu. Engineering RNA for Targeted siRNA Delivery and Medical Application. *Advanced Drug Delivery Reviews*, 62(6):650 – 666, 2010. ISSN 0169-409X. doi: https://doi.org/10.1016/j.addr.2010.03.008. URL http://www.sciencedirect.com/science/article/pii/S0169409X10000773. From Biology to Materials: Engineering DNA and RNA for Drug Delivery and Nanomedicine.

Anvita Gupta and James Zou. Feedback gan (fbgan) for dna: a novel feedback-loop architecture for optimizing protein functions. *arXiv preprint arXiv:1804.01694*, 2018.

R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29 (2):147–160, April 1950. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1950.tb00463.x.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017. URL http://arxiv.org/abs/1707.02286.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Ivo Hofacker, Walter Fontana, Peter Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast Folding and Comparison of RNA Secondary Structures. *Monatshefte fuer Chemie/Chemical Monthly*, 125:167–188, 02 1994.

Giles Hooker. Generalized functional ANOVA diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 16(3):709–732, 2007.

F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In E. Xing and T. Jebara (eds.), *Proceedings of the 31th International Conference on Machine Learning, (ICML'14)*, pp. 754–762. Omnipress, 2014.

Ioanna Kalvari, Joanna Argasinska, Natalia Quinones-Olvera, Eric P Nawrocki, Elena Rivas, Sean R Eddy, Alex Bateman, Robert D Finn, and Anton I Petrov. Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. *Nucleic acids research*, 46(D1):D335–D342, 2017.

Kriti Kaushik, Ambily Sivadas, Shamsudheen Karuthedath Vellarikkal, Ankit Verma, Rijith Jayarajan, Satyaprakash Pandey, Tavprithesh Sethi, Souvik Maiti, Vinod Scaria, and Sridhar Sivasubbu. RNA secondary structure profiling in zebrafish reveals unique regulatory features. *BMC Genomics*, 20:147, 2018.

P Kerpedjiev, C Höner zu Siederdissen, and IL Hofacker. Predicting RNA 3D structure using a coarse-grain helix-centered model. *RNA*, 21(6):1110–1121, 2015. doi: 10.1261/rna.047522.114.

Robert Kleinkauf, Torsten Houwaart, Rolf Backofen, and Martin Mann. antaRNA–Multi-objective inverse folding of pseudoknot RNA using ant-colony optimization. *BMC bioinformatics*, 16(1): 389, 2015.

Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. 44(1):117–130, Jun 2015.

Ronny Lorenz, Stephan H. Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, Nov 2011a. ISSN 1748-7188. doi: 10.1186/1748-7188-6-26. URL https://doi.org/10.1186/1748-7188-6-26.

Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011b.

Sarai Meyer, James Chappell, Sitara Sankar, Rebecca Chew, and Julius B. Lucks. Improving fold activation of small transcription activating RNAs (STARs) with rational RNA engineering strategies. *Biotechnology and Bioengineering*, 113(1):216–225, 2015. doi: 10.1002/bit.25693. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.25693.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

Vladimir Reinharz, François Major, and Jérôme Waldispühl. Towards 3d structure prediction of large rna molecules: an integer programming framework to insert local 3d motifs in rna secondary structure. *Bioinformatics*, 28(12):i207–i214, 2012. doi: 10.1093/bioinformatics/bts226. URL http://dx.doi.org/10.1093/bioinformatics/bts226.

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018. ISSN 0036-8075. doi: 10.1126/science.aat2663. URL http://science.sciencemag.org/content/361/6400/360.

Michael Schaarschmidt, Alexander Kuhnle, and Kai Fricke. Tensorforce: A tensorflow library for applied reinforcement learning. Web page, 2017. URL https://github.com/reinforceio/tensorforce.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Bruce A Shapiro. An algorithm for comparing multiple rna secondary structures. *Bioinformatics*, 4 (3):387–393, 1988.

Jade Shi, Rhiju Das, and Vijay S Pande. Sentrna: Improving computational rna design by incorporating a prior of human design strategies. *arXiv preprint arXiv:1803.03146*, 2018.

Akito Taneda. MODENA: a multi-objective RNA inverse folding. *Advances and applications in bioinformatics and chemistry: AABC*, 4:1, 2011.

Rebecca M. Terns and Michael P. Terns. CRISPR-based technologies: prokaryotic defense weapons repurposed. *Trends in Genetics*, 30(3):111–118, 2014. doi: https://doi.org/10.1016/j.tig.2014.01.003.

Lee E. Vandivier, Stephen J. Anderson, Shawn W. Foley, and Brian D. Gregory. The conservation and function of RNA secondary structure in plants. *Annual review of plant biology*, 67:463–488, 2016.

Manja Wachsmuth, Sven Findeiß, Nadine Weissheimer, Peter F. Stadler, and Mario Mörl. De novo design of a synthetic riboswitch that regulates transcription termination. *Nucleic Acids Research*, 41(4):2541–2551, 2013. doi: 10.1093/nar/gks1330. URL http://dx.doi.org/10.1093/nar/gks1330.

Xiufeng Yang, Kazuki Yoshizoe, Akito Taneda, and Koji Tsuda. RNA inverse folding using Monte Carlo tree search. *BMC bioinformatics*, 18(1):468, 2017.

Joseph N. Zadeh, Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. Pierce, Asif R. Khan, Robert M. Dirks, and Niles A. Pierce. Nupack: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173, 2010. doi: 10.1002/jcc.21596. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21596.

Y Zhao, Y Huang, Z Gong, Y Wang, J Man, and Y Xiao. Automated and fast building of three-dimensional RNA structures. *Scientific Reports*, 2(734), 2012. doi: 10.1038/srep00734.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*, 2017. Published online: iclr.cc.

B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018.

M Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.

Michael Zuker and David Sankoff. RNA secondary structures and their prediction. *Bulletin of Mathematical Biology*, 46(4):591 – 621, 1984. ISSN 0092-8240. doi: https://doi.org/10.1016/S0092-8240(84)80062-2. URL http://www.sciencedirect.com/science/article/pii/S0092824084800622.

## A PSEUDOCODE FOR REWARD FUNCTION

**Algorithm 1:** Pseudo code for the local improvement step (LIS) using hamming distance $\mathrm{d_H}(\cdot,\cdot)$ and folding algorithm $\mathcal{F}(\cdot)$.

```
input  : designed solution φ, target structure ω, initial hamming distance δ
output : locally improved distance
Δ ← ∅
nucleotide_combinations ← {A, G, U, C}^δ
candidate_solutions ← replaceMismatchedSites(φ, ω, nucleotide_combinations)
foreach ψ ∈ candidate_solutions do
    δ ← d_H(F(ψ), ω)
    if δ = 0 then
        return δ
    end
    Δ ← Δ ∪ {δ}
end
return min Δ
```

**Algorithm 2:** Pseudo code for computing reward $\mathcal{R}_\omega^T$ using LIS (Algorithm 1), hamming distance $\mathrm{d_H}(\cdot,\cdot)$ and folding algorithm $\mathcal{F}(\cdot)$.

```
input  : designed solution φ, target structure ω, LIS cut-off parameter ξ
output : reward R_ω^T
δ ← d_H(F(φ), ω)
if δ = 0 then
    return δ
else if δ < ξ then
    δ ← LIS(φ, ω, δ)
end
L_ω ← (δ/|ω|)^α
return −L_ω
```

## B TECHNICAL DETAILS

We used the implementation of the *Zuker algorithm* provided by *ViennaRNA* (Lorenz et al., 2011b) versions 2.4.8 (*MCTS-RNA*, *RL-LS* and *LEARNA*), 2.1.9 (*antaRNA*) and 2.4.9 (*RNAInverse*). Our implementation uses the Reinforcement Learning library *tensorforce*, version 0.3.3 (Schaarschmidt et al., 2017) working with *TensorFlow* version 1.4.0 (Abadi et al., 2015). All computations were done on Broadwell E5-2630v4 2.2 GHz CPUs with a limitation of 5 GByte RAM per core. For the training phase of Meta-LEARNA, we used all 20 cores of these machines, but at evaluation time, all methods were only allowed a single core (using core binding).

## C RFAM-LEARN DATASET

To ensure a large enough and interesting dataset, we downloaded all families of the Rfam database version 13.0 (Kalvari et al., 2017) and folded them using the ViennaRNA package (Lorenz et al., 2011a). We removed all secondary structure with multiple known solutions, and only kept structures with lengths between 50 and 450 to match the existing datasets. To focus on the harder sequences, we only kept the ones that a single run of *MCTS-RNA* could not solve within 30 seconds. We chose *MCTS-RNA* for filtering as it was the fastest algorithm from the literature. The remaining secondary structures were split into a training set of 65000, a validation set of 100, and a test set of 100 secondary structures.

## D BENCHMARKS

Table 2: Overview on the three benchmarks Eterna100 (Anderson-Lee et al., 2016), Rfam-Taneda (Taneda, 2011) and Rfam-Learn we used for our experiments. The table displays the timeout, the number of evaluations for each target structure, the number of sequences and the range of sequence lengths for the corresponding benchmark.

| DATASET | TIMEOUT | EVALUATIONS | SEQUENCES | LENGTH |
|---|---|---|---|---|
| ETERNA100 | 24H | 5 | 100 | 12–400 |
| RFAM-TANEDA | 10MIN | 50 | 29 | 54–451 |
| RFAM-LEARN-TRAIN | – | – | 65000 | 50–450 |
| RFAM-LEARN-VAL | – | – | 100 | 50–444 |
| RFAM-LEARN-TEST | 1H | 5 | 100 | 50–446 |

## E JOINT ARCHITECTURE AND HYPERPARAMETER SEARCH

Here, we provide a detailed description of the search space, more details on the different computational budgets used for optimization, and the final configurations found by the optimizer. The search spaces for LEARNA and Meta-LEARNA can be found in Tables 3 and 4.

Table 3: Search space for the agent's architecture and the hyperparameters used for optimizing *LEARNA* for both the 10 and 30 minutes budget.

| Parameter Name | Type | Range | Prior |
|---|---|---|---|
| filter size in $1^{st}$ conv layer | integer | [0, 8] | uniform |
| filter size in $2^{nd}$ conv layer | integer | [0, 4] | uniform |
| # filter in $1^{st}$ conv layer | integer | [1, 32] | log-uniform |
| # filter in $2^{nd}$ conv layer | integer | [1, 32] | log-uniform |
| # fully connected layers | integer | [1, 2] | uniform |
| # units in fully connected layer(s) | integer | [8, 64] | log-uniform |
| # LSTM layers | integer | [0, 2] | uniform |
| # units in every LSTM layer | integer | [1, 64] | log-uniform |
| # state space radius | integer | [1, 32] | uniform |
| embedding dimensionality | integer | [0, 4] | uniform |
| batch size | integer | [32, 128] | log-uniform |
| entropy regularization | float | $[1 \cdot 10^{-5}, 1 \cdot 10^{-2}]$ | log-uniform |
| learning rate for PPO | float | $[1 \cdot 10^{-5}, 1 \cdot 10^{-3}]$ | log-uniform |
| reward exponent | float | [1, 10] | uniform |

We now specify the budgets mentioned in the main text used to speed up the optimization. For LEARNA (both the 10min and 30min variant), we directly optimize the performance on the validation set. By using small timeout, we can eliminate bad configurations quickly and focus most of the resources on the promising ones. In BOHB, these budgets (here the timeout) are geometrically distributed with a factor of three between them. This leads to the budgets shown in the legend of Figs. 4 and 5.

For Meta-LEARNA, we limit the training time on the Rfam-Learn-train dataset directly and keep the evaluation timeout on the validation set fixed at 60 seconds. The maximum timeout of 1 hour on 20 CPUs was chosen to almost match the timeout of the Eterna100 dataset for a single sequence. The minimum timeout was set to 400 seconds, chosen by preliminary runs and inspecting the achieved performance. The validation timeout of one minute was chosen such that the training time on the smallest budget of 400 seconds is still larger than the evaluation time for the 100 validation sequences. Additionally, this encourages the agent to find a solution quickly.

Table 4: Search space for the agent's architecture and the hyperparameters used for optimizing *Meta-Learna*.

| Parameter Name | Type | Range | Prior |
|---|---|---|---|
| filter size in 1st conv layer | integer | [0, 8] | uniform |
| filter size in 2nd conv layer | integer | [0, 4] | uniform |
| # filters in 1st conv layer | integer | [1, 32] | uniform |
| # filters in 2nd conv layer | integer | [1, 32] | uniform |
| # fully connected layers | integer | [1, 2] | uniform |
| # units in fully connected layer(s) | integer | [8, 64] | log-uniform |
| # LSTM layers | integer | [0, 2] | uniform |
| # units in every LSTM layer | integer | [1, 64] | log-uniform |
| # state space radius | integer | [0, 32] | uniform |
| embedding dimensionality | integer | [0, 4] | uniform |
| batch size | integer | [32, 128] | log-uniform |
| entropy regularization | float | $[5 \cdot 10^{-5}, 5 \cdot 10^{-3}]$ | log-uniform |
| learning rate for PPO | float | $[1 \cdot 10^{-6}, 1 \cdot 10^{-4}]$ | log-uniform |
| reward exponent | float | [1, 10] | uniform |

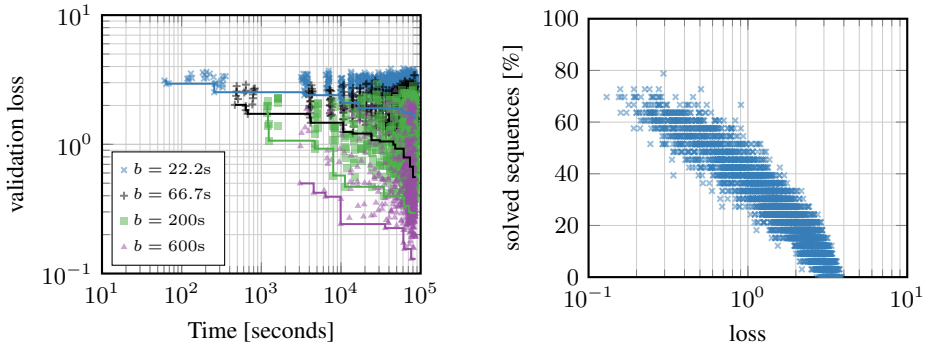



Figure 4: Left: Observed validation loss during the BOHB run for *LEARNA-10min*. The different budgets $b$ corrensponds to the timeout for each of the 33 validation sequences. Right: Relationship between the observed validation loss (sum of minimal, normalized Hamming distances) and the fraction of solved sequences.

We asses the robustness of a found configuration/agent by evaluating it multiple times on our validation set (Figure 7). This way, we can ensure that the chosen solution was not simply an outlier. Ideally, we want to achieve stable performance across different random seeds.

Finally, Table 5 summarizes the evaluated configurations. The biggest differences between LEARNA-10min and LEARNA-30min are the bigger batch size, the larger reward exponent and the much smaller state space radius for the latter. Inspecting configurations that achieved a slightly worse performance, reveals that those vary quite substantially indicating that many different configurations lead to almost the same performance.

In contrast, the configuration for Meta-LEARNA a very large state space dimensionality, a large CNN part with many filters and no LSTM component. This is not to say that LSTMs are worse for this task, but is more likely due to the relatively short trainings budget of 1 hour on 20 CPUs.

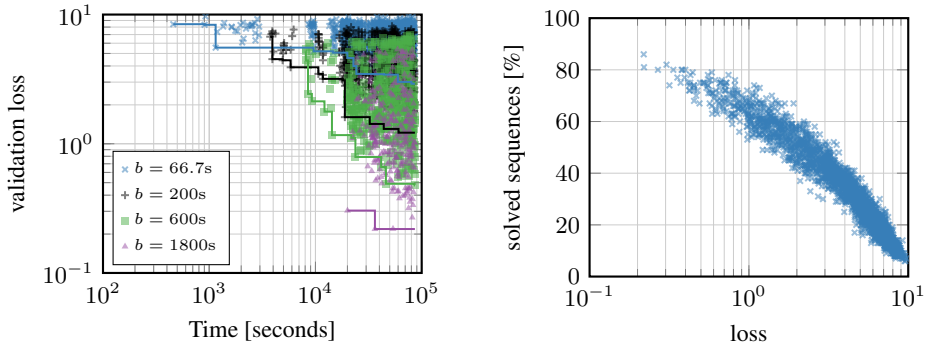



Figure 5: Left: Observed validation loss during the BOHB run for *LEARNA-30min*. The different budgets $b$ corrensponds to the timeout for each of the 100 validation sequences. Right: Relationship between the observed validation loss (sum of minimal, normalized Hamming distances) and the fraction of solved sequences.

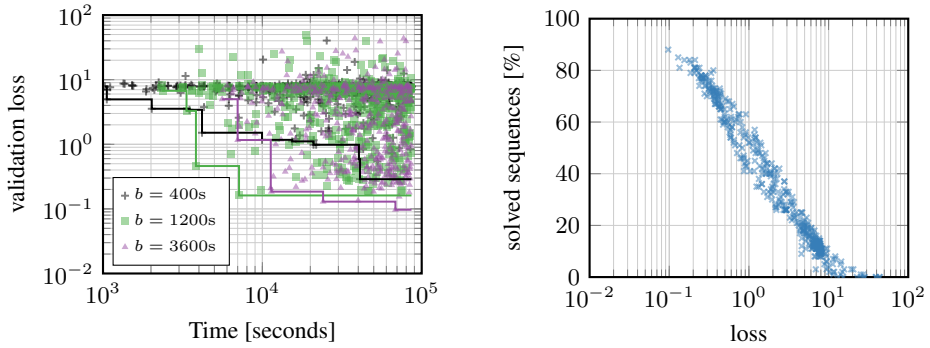



Figure 6: Left: Observed validation loss during the BOHB run for *Meta-LEARNA*. The different budgets $b$ corrensponds to the training time on 20 CPUs in parallel before evaluating on the 100 validation sequences for 3 minutes each. The results seem to suggest that one can achieve a very similar performance with only 20 minutes of training, which could imply that much longer training of the agent might be required for substantially better performance. Right: Relationship between the observed validation loss (sum of minimal, normalized Hamming distances) and the fraction of solved sequences during validation. The plot suggests that our loss metric correlates strongly with the number of successfully found primary sequences.

Table 5: The selected configurations for each scenario and budget.

| Parameter Name | LEARNA-10min | LEARNA-30min | Meta-LEARNA |
|---|---|---|---|
| filter size in 1st conv layer | 5 | 0 | 5 |
| filter size in 2nd conv layer | 3 | 3 | 7 |
| # filters in 1st conv layer | 8 | 10 | 32 |
| # filters in 2nd conv layer | 1 | 3 | 14 |
| # fully connected layers | 1 | 1 | 1 |
| # units in fully connected layer(s) | 52 | 32 | 9 |
| # LSTM layers | 2 | 2 | 0 |
| # units in every LSTM layer | 4 | 7 | 53 |
| # state space radius | 16 | 2 | 26 |
| embedding dimensionality | 0 | 0 | 1 |
| batch size | 32 | 79 | 80 |
| entropy regularization | $4.44 \cdot 10^{-4}$ | $1.63 \cdot 10^{-4}$ | $1.98 \cdot 10^{-4}$ |
| learning rate for PPO | $5.49 \cdot 10^{-4}$ | $3.38 \cdot 10^{-4}$ | $6.37 \cdot 10^{-5}$ |
| reward exponent | 5.72 | 9.43 | 9.22 |

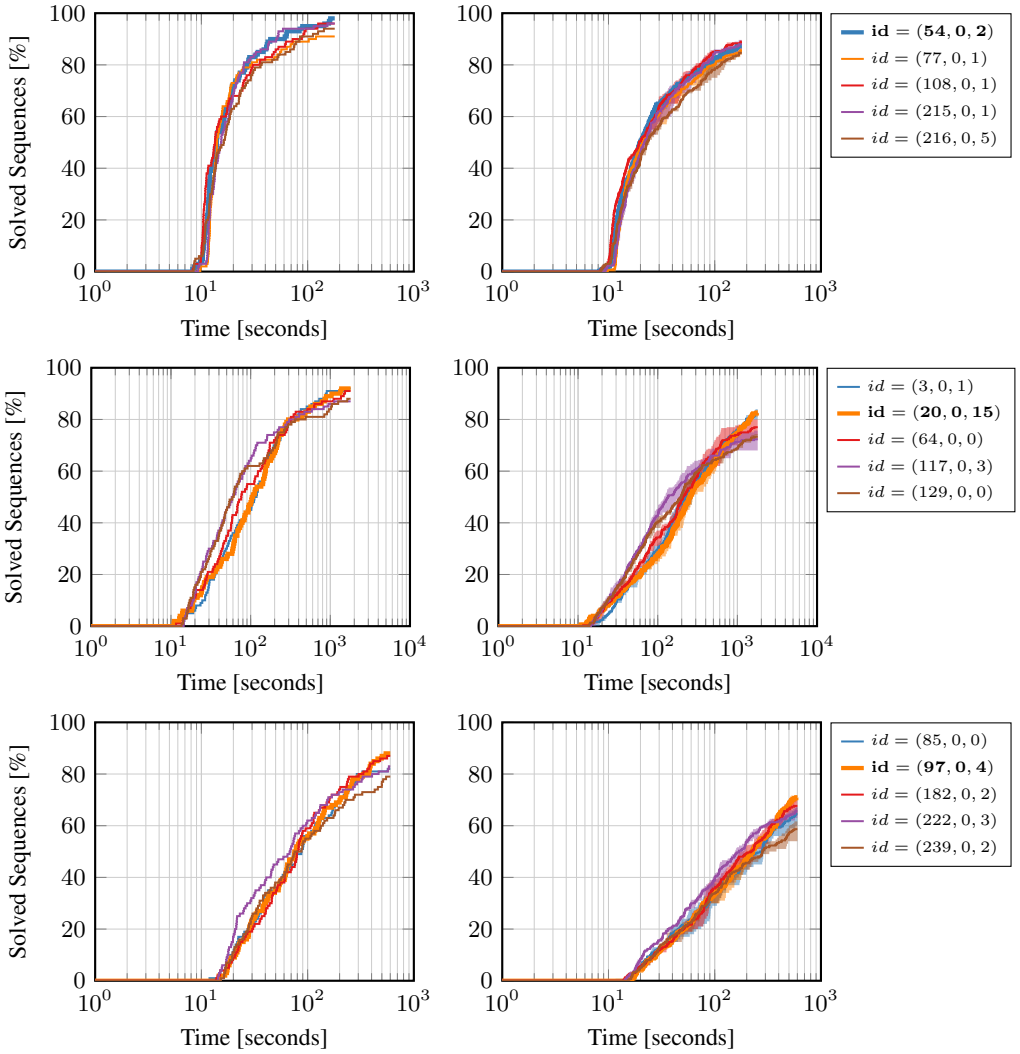



Figure 7: 5 independent validation runs of the best 5 configurations for *Meta-LEARNA* (top), *LEARNA* with a 30 minute budget (middle), *LEARNA* with a 10 minute budget (bottom) on the Rfam-Learn-Validation set to asses the robustness. Chosen configurations are highlighted in bold.

## F COMPARISON PLOTS

Here, we show the performance of all methods tested on all three benchmarks. In particular, we present the fraction of solved sequences for Eterna100, Rfam-Taneda, and Rfam-Learn-Test accumulated and averaged over independent evaluation runs.

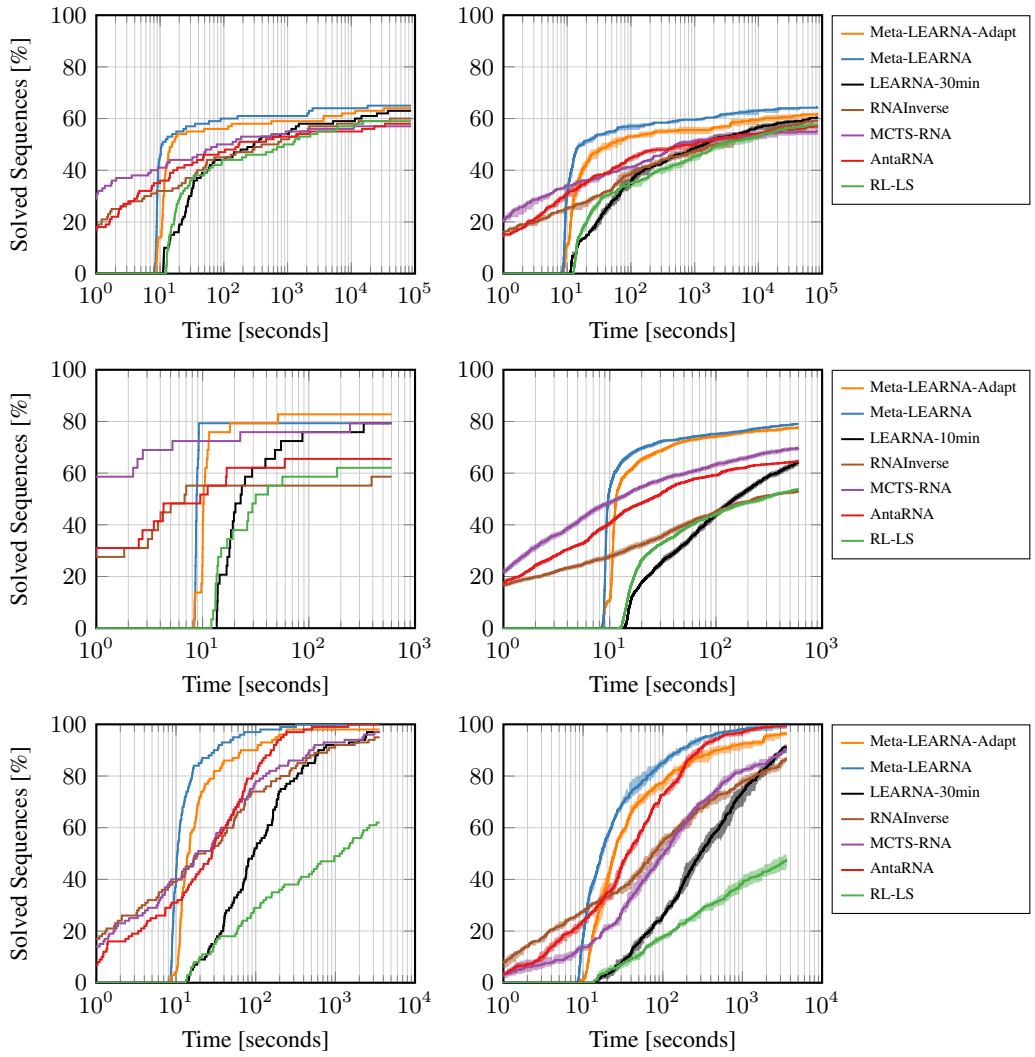



Figure 8: Comparison of all methods on the Eterna100 (top) Rfam-Taneda (middle) and Rfam-Learn-Test (bottom) benchmark. The left side shows the fraction of solved target structures accumulated over independent evaluation runs, while the right side shows the mean of that fraction with confident intervals using 5, 50, and 5 independent evaluation runs respectively. On all three benchmarks, Meta-LEARNA outperforms all other methods in terms of number of solved sequences and/or time to achieve this performance. For Eterna100, all three strategies of our novel approach achieve new state-of-the-art results, while being orders of magnitudes faster. On our benchmark, all algorithms except *RL-LS* solve at least 95 % of the target structures, but Meta-LEARNA performs best (after a short lag due to computational overhead).

## G COMPARISON SUMMARY TABLES

Table 6: Comparison of all methods on Eterna100. Results list the number of solved target structures in at least 1, 2, 3, 4, or all of the evaluation runs in percent. *Random* describes a random agent.

| METHOD | SOLVED SEQUENCES [%] | | | | |
|---|---|---|---|---|---|
| | TOTAL | 2 RUNS | 3 RUNS | 4 RUNS | ALL RUNS |
| RANDOM | 39 | 36 | 35 | 34 | 32 |
| MCTS-RNA | 57 | 57 | 56 | 54 | 51 |
| ANTARNA | 58 | 58 | 58 | 56 | 55 |
| RL-LS | 59 | 59 | 58 | 57 | 55 |
| RNAINVERSE | 60 | 60 | 59 | 59 | 58 |
| LEARNA-30MIN | 63 | 61 | 60 | 60 | 58 |
| META-LEARNA-ADAPT | 64 | 63 | 62 | 60 | 60 |
| META-LEARNA | **65** | **65** | **64** | **64** | **64** |

Table 7: Comparison of all methods on Rfam-Taneda. Results list the number of solved target structures in at least 1, 5, 10, 25, or all of the evaluation runs in percent. *Random* describes a random agent.

| METHOD | SOLVED SEQUENCES [%] | | | | |
|---|---|---|---|---|---|
| | TOTAL | 5 RUNS | 10 RUNS | 25 RUNS | ALL RUNS |
| RANDOM | 35 | 28 | 28 | 24 | 21 |
| RNAINVERSE | 59 | 55 | 55 | 52 | 48 |
| RL-LS | 62 | 62 | 55 | 52 | 48 |
| ANTARNA | 66 | 66 | 66 | 66 | 62 |
| MCTS-RNA | 79 | 76 | 72 | 72 | 59 |
| LEARNA-10MIN | 79 | 76 | 76 | 66 | 28 |
| META-LEARNA-ADAPT | **83** | **83** | **79** | **79** | 69 |
| META-LEARNA | 79 | 79 | 79 | **79** | **76** |

Table 8: Comparison of all methods on Rfam-Learn-Test. Results list the number of solved target structures in at least 1, 2, 3, 4, or all of the evaluation runs in percent. *Random* describes a random agent.

| METHOD | SOLVED SEQUENCES [%] | | | | |
|---|---|---|---|---|---|
| | TOTAL | 2 RUNS | 3 RUNS | 4 RUNS | ALL RUNS |
| RANDOM | 31 | 26 | 25 | 22 | 21 |
| RL-LS | 62 | 53 | 45 | 41 | 37 |
| RNAINVERSE | 95 | 90 | 87 | 83 | 78 |
| MCTS-RNA | 97 | 94 | 91 | 89 | 82 |
| ANTARNA | **100** | 99 | 99 | **99** | **99** |
| LEARNA-10MIN | 95 | 94 | 92 | 84 | 74 |
| LEARNA-30MIN | 97 | 94 | 93 | 90 | 82 |
| META-LEARNA-ADAPT | 98 | 98 | 98 | 96 | 92 |
| META-LEARNA | **100** | **100** | **100** | **99** | 98 |

Table 9: Comparison of all methods on Eterna100. Results list the number of solved target structures at different time points in percent. *Random* describes a random agent.

| METHOD | SOLVED SEQUENCES [%] | | | | | | |
|---|---|---|---|---|---|---|---|
| | 10S | 1MIN | 30MIN | 1H | 4H | 12H | 24H |
| RANDOM | 19 | 22 | 29 | 33 | 37 | 39 | 39 |
| MCTS-RNA | 41 | 48 | 55 | 56 | 57 | 57 | 57 |
| ANTARNA | 36 | 46 | 54 | 55 | 55 | 58 | 58 |
| RL-LS | 0 | 40 | 53 | 55 | 58 | 59 | 59 |
| RNAINVERSE | 32 | 44 | 55 | 57 | 59 | 60 | 60 |
| LEARNA-30MIN | 0 | 41 | 58 | 58 | 60 | 63 | 63 |
| META-LEARNA-ADAPT | 14 | 56 | 59 | 59 | 63 | 64 | 64 |
| META-LEARNA | **45** | **59** | **61** | **64** | **64** | **65** | **65** |

Table 10: Comparison of all methods on Rfam-Taneda. Results list the number of solved target structures at different time points in percent. *Random* describes a random agent.

| METHOD | SOLVED SEQUENCES [%] | | | | |
|---|---|---|---|---|---|
| | 10S | 30S | 1MIN | 5MIN | 10MIN |
| RANDOM | 28 | 28 | 31 | 31 | 35 |
| ANTARNA | 52 | 62 | 66 | 66 | 66 |
| RNAINVERSE | 55 | 55 | 55 | 55 | 59 |
| RL-LS | 0 | 48 | 59 | 62 | 62 |
| MCTS-RNA | 72 | 76 | 76 | 79 | 79 |
| LEARNA-10MIN | 0 | 62 | 72 | 76 | 79 |
| META-LEARNA-ADAPT | 28 | **79** | **83** | **83** | **83** |
| META-LEARNA | **79** | **79** | 79 | 79 | 79 |

Table 11: Comparison of all methods on Rfam-Learn-Test. Results list the number of solved target structures at different time points in percent. *Random* describes a random agent.

| METHOD | SOLVED SEQUENCES [%] | | | | | | |
|---|---|---|---|---|---|---|---|
| | 10S | 30S | 1MIN | 5MIN | 10MIN | 30MIN | 1H |
| RANDOM | 11 | 14 | 16 | 23 | 25 | 30 | 31 |
| RL-LS | 0 | 14 | 21 | 38 | 45 | 56 | 62 |
| RNAINVERSE | 39 | 53 | 66 | 83 | 89 | 93 | 95 |
| MCTS-RNA | 40 | 55 | 68 | 86 | 92 | 94 | 97 |
| ANTARNA | 36 | 58 | 73 | 97 | 99 | **100** | **100** |
| LEARNA-10MIN | 0 | 24 | 39 | 78 | 85 | 93 | 95 |
| LEARNA-30MIN | 0 | 15 | 34 | 78 | 88 | 93 | 97 |
| META-LEARNA-ADAPT | 4 | 81 | 87 | 98 | 98 | 98 | 98 |
| META-LEARNA | **41** | **90** | **95** | **100** | **100** | **100** | **100** |

## H ABLATION

Here, we study the contribution of different components of our approaches with an ablation. By removing one component at a time, we can see the impact it has on the final performance.

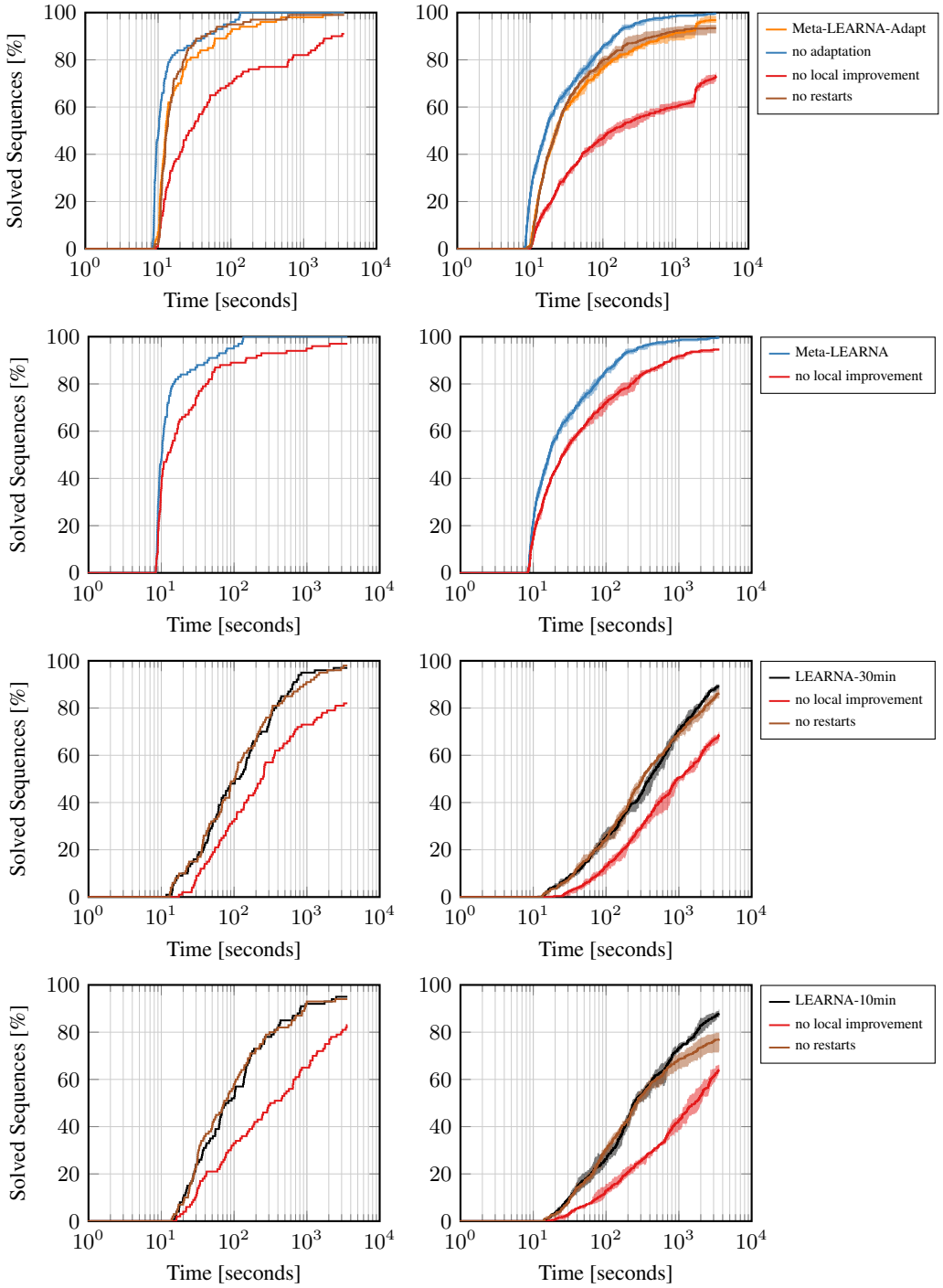



Figure 9: Ablation study of *Meta-LEARNA-Adapt* (first row), *Meta-LEARNA* (second row), *LEARNA-30min* (third row) and *LEARNA-10min* (fourth row) on Rfam-Learn-Test. The left side shows the accumulated number of solved target structures over 5 independent runs, while the right side shows the mean with confident interval.

# I FUNCTIONAL ANALYSIS OF VARIANCE FOR META-LEARNA

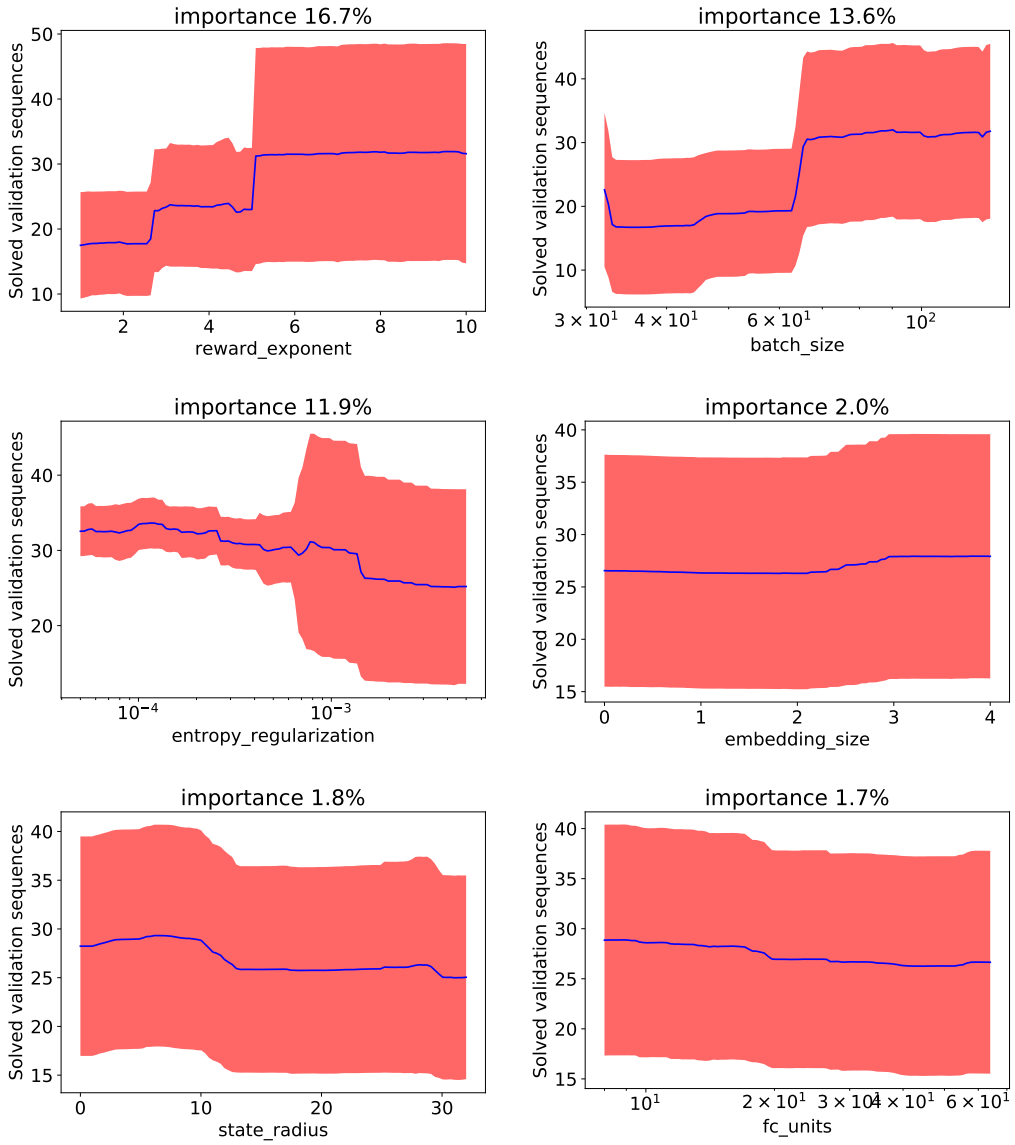



Figure 10: fANOVA plots for the most important individual parameters. The importance measures the fraction of the total variance explained by a single parameter. All others are marginalized out based on random forest regression model. The line shows the resulting mean as a function a single parameter, while the shaded interval corresponds to the remaining variance due to the other parameters.

In addition to the ablation study, we can also perform an analysis of variance (ANOVA) that quantifies the global importance of a parameter by the fraction of the total variance it explains. Because our parameter space is rather high dimensional, and we collected a limited (relative to the dimensionality) and highly biased (because we optimized performance) set of evaluations, we use the functional ANOVA (fANOVA) (Hooker, 2007). In particular, we use fANOVA based on random forests as introduced by Hutter et al. (2014). The results are shown in Figures 10 and 11.

Among the six most important individual hyperparameters, we find the reward representation (the reward exponent), training and regularization hyperparameters (batch size and entropy regularization in PPO), the state space representation (state radius (number of sites presented to the agent) and embedding size (dimensionality of the embedding vector learned for the dot-bracket notation)), and

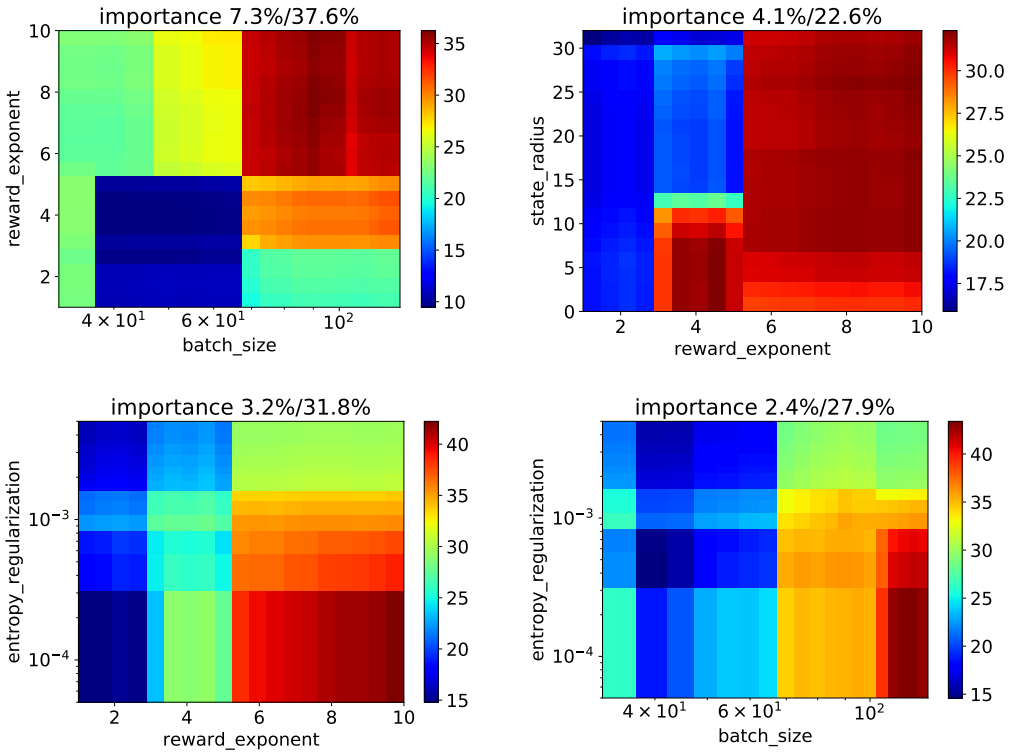



Figure 11: fANOVA plots for the most important pairs of parameters. The importances shown are the individual one (first) and the total (second). The former concerns only the true pairwise contribution while the latter quantifies the importance including the individual contributions shown in Fig. 10 of both parameters. We only show the four most important once explain a negligible fraction of the variance. Here the plot shows the (predicted) mean number of solved sequences in the validation set when marginalizing over all other parameters.

lastly an architectural hyperparameter (number of units in the fully connected layer(s)). The global analysis performed by fANOVA highlights hyperparameters that impact performance most across the hyperparameter space. As a result, the shown number of solved validation sequences is rather low in the plots ($\lesssim 50$, where the best found configurations achieved almost 90, see Figure 6). It is important to note that the quantitative behavior predicted by the fANOVA analysis does not have to be representative for the best configurations, especially if the *good part* of the space is rather small. This also means other hyperparameters, e.g., the architecture and type of the network, can be more important than indicated by the fANOVA in order to reach peak performance.

From the plots, we see larger batch sizes perform better, which might be specific for the time constraint on the training and might change with a much larger budget. The reward exponent should also be set quite high, although a smaller one also seems to work well in combination with a rather small state radius (see top right panel of Fig. 11).

## J PERFORMANCE ACROSS SEQUENCE LENGTHS

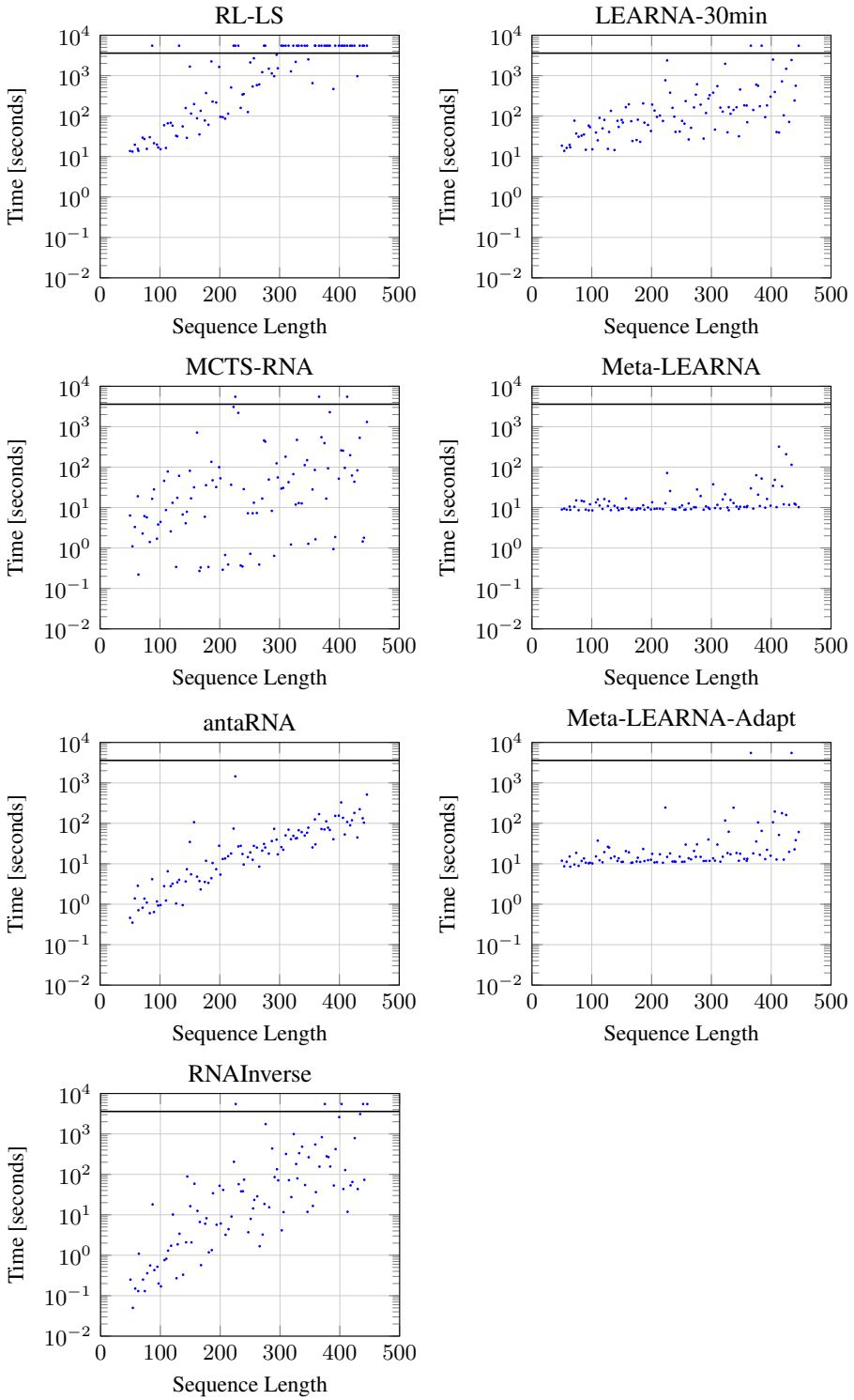



Figure 12: Minimum solution times across sequence lengths on the Rfam-Learn-Test benchmark. The solid line represents the timeout of 1 hour for the Rfam-Learn-Test benchmark and points drawn above this line were not solved.

# K COMPARISON DETAIL TABLES

Table 12: Results for 5 independent attempts on the first half of the 100 target structures of the Rfam-Learn-Test benchmark. The timeout for each attempt was set to 1 hour.

| ID | LEARNA-30MIN | META-LEARNA | MCTS-RNA | RL-LS | RNAINVERSE | ANTARNA |
|---|---|---|---|---|---|---|
| 1 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 2 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 3 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 4 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 6 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 7 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 8 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 9 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 10 | 5/5 | 5/5 | 4/5 | - | 5/5 | 5/5 |
| 11 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 12 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 13 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 14 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 15 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 16 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 17 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 18 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 19 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 20 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 21 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 22 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 23 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 24 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 25 | 5/5 | 5/5 | 5/5 | 4/5 | 5/5 | 5/5 |
| 26 | 5/5 | 5/5 | 4/5 | 2/5 | 5/5 | 5/5 |
| 27 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 28 | 2/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 29 | 5/5 | 5/5 | 3/5 | 3/5 | 5/5 | 5/5 |
| 30 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 31 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 32 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 33 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 34 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 35 | 5/5 | 5/5 | 5/5 | 4/5 | 5/5 | 5/5 |
| 36 | 5/5 | 5/5 | 4/5 | 3/5 | 5/5 | 5/5 |
| 37 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 38 | 5/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 39 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 40 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 41 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 42 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 43 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 44 | 5/5 | 5/5 | 1/5 | - | 4/5 | 5/5 |
| 45 | 1/5 | 5/5 | - | - | - | 1/5 |
| 46 | 5/5 | 5/5 | 1/5 | - | 5/5 | 5/5 |
| 47 | 5/5 | 5/5 | 5/5 | 3/5 | 5/5 | 5/5 |
| 48 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 49 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 50 | 5/5 | 5/5 | 5/5 | 4/5 | 5/5 | 5/5 |
| TOTAL | 243/250 | 250/250 | 232/250 | 205/250 | 244/250 | 246/250 |
| SOLVED | 50/50 | 50/50 | 49/50 | 45/50 | 49/50 | 50/50 |

Table 13: Results for 5 independent attempts on the second half of the 100 target structures of the Rfam-Learn-Test benchmark. The timeout for each attempt was set to 1 hour.

| ID | LEARNA | META-LEARNA | MCTS-RNA | RL-LS | RNAINVERSE | ANTARNA |
|---|---|---|---|---|---|---|
| 51 | 5/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 52 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 53 | 5/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 54 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 |
| 55 | 5/5 | 5/5 | 5/5 | 3/5 | 5/5 | 5/5 |
| 56 | 5/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 57 | 5/5 | 5/5 | 5/5 | - | 4/5 | 5/5 |
| 58 | 5/5 | 5/5 | 5/5 | - | 3/5 | 5/5 |
| 59 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 |
| 60 | 5/5 | 5/5 | 5/5 | 2/5 | 3/5 | 5/5 |
| 61 | 5/5 | 5/5 | 4/5 | 2/5 | 1/5 | 5/5 |
| 62 | 5/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 63 | 5/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 64 | 3/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 65 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 66 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 67 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 68 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 |
| 69 | 4/5 | 5/5 | 5/5 | - | 3/5 | 5/5 |
| 70 | 5/5 | 5/5 | 5/5 | 1/5 | 5/5 | 5/5 |
| 71 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 72 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 73 | 4/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 74 | 5/5 | 5/5 | 5/5 | - | 2/5 | 5/5 |
| 75 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 76 | 5/5 | 5/5 | 5/5 | 1/5 | 2/5 | 5/5 |
| 77 | 5/5 | 5/5 | 5/5 | 4/5 | 5/5 | 5/5 |
| 78 | 5/5 | 5/5 | 5/5 | - | 4/5 | 5/5 |
| 79 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 80 | - | 4/5 | - | - | 1/5 | 5/5 |
| 81 | 4/5 | 5/5 | 5/5 | - | 2/5 | 5/5 |
| 82 | 4/5 | 5/5 | 4/5 | - | - | 5/5 |
| 83 | 3/5 | 5/5 | 5/5 | - | 4/5 | 5/5 |
| 84 | 5/5 | 5/5 | 4/5 | - | 3/5 | 5/5 |
| 85 | - | 5/5 | 1/5 | - | 5/5 | 5/5 |
| 86 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 |
| 87 | 4/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 88 | 3/5 | 5/5 | 5/5 | - | 1/5 | 5/5 |
| 89 | 1/5 | 5/5 | 2/5 | - | - | 5/5 |
| 90 | 4/5 | 5/5 | 2/5 | - | 5/5 | 5/5 |
| 91 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 92 | 5/5 | 5/5 | - | - | 5/5 | 5/5 |
| 93 | 4/5 | 5/5 | 4/5 | - | 4/5 | 5/5 |
| 94 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 95 | 5/5 | 5/5 | 5/5 | - | 1/5 | 5/5 |
| 96 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 |
| 97 | 1/5 | 3/5 | 2/5 | - | 1/5 | 5/5 |
| 98 | 5/5 | 5/5 | 5/5 | - | - | 5/5 |
| 99 | 4/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 100 | - | 5/5 | 3/5 | - | - | 5/5 |
| TOTAL | 218/250 | 247/250 | 221/250 | 33/250 | 189/250 | 250/250 |
| SOLVED | 47/50 | 50/50 | 48/50 | 17/50 | 46/50 | 50/50 |

Table 14: Results for 5 independent attempts on the first half of the 100 target structures of the Eterna100 benchmark. The timeout for each attempt was set to 24 hours.

| ID | LEARNA-30MIN | META-LEARNA | MCTS-RNA | RL-LS | RNAINVERSE | ANTARNA |
|---|---|---|---|---|---|---|
| 1 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 2 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 3 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 4 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 6 | 5/5 | 5/5 | 5/5 | - | 5/5 | 5/5 |
| 7 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 8 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 9 | - | 5/5 | - | 4/5 | - | 3/5 |
| 10 | 5/5 | 5/5 | - | 5/5 | 5/5 | 5/5 |
| 11 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 12 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 13 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 14 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 15 | 4/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 16 | 5/5 | 5/5 | - | 3/5 | - | - |
| 17 | 5/5 | 5/5 | 4/5 | 5/5 | 2/5 | - |
| 18 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 19 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 20 | 5/5 | 5/5 | 4/5 | 5/5 | 5/5 | 5/5 |
| 21 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 22 | - | 5/5 | 5/5 | 5/5 | - | 5/5 |
| 23 | 5/5 | 5/5 | - | 5/5 | 5/5 | 5/5 |
| 24 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 25 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 26 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 27 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 28 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 29 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 30 | 5/5 | 5/5 | - | 5/5 | 5/5 | 5/5 |
| 31 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 32 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 33 | 5/5 | 5/5 | - | - | 5/5 | 5/5 |
| 34 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 35 | - | 5/5 | - | - | - | - |
| 36 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 37 | 5/5 | 5/5 | 5/5 | 2/5 | 4/5 | - |
| 38 | 5/5 | 5/5 | 5/5 | - | - | - |
| 39 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 40 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 41 | 5/5 | 5/5 | - | 5/5 | 5/5 | 5/5 |
| 42 | 5/5 | 5/5 | 4/5 | 5/5 | 5/5 | 5/5 |
| 43 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 44 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 45 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 46 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 47 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 48 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 49 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 50 | - | - | - | - | - | - |
| TOTAL | 228/250 | 245/250 | 202/250 | 219/250 | 216/250 | 218/250 |
| SOLVED | 46/50 | 49/50 | 41/50 | 45/50 | 44/50 | 44/50 |

Table 15: Results for 5 independent attempts on the second half of the 100 target structures of the Eterna100 benchmark. The timeout for each attempt was set to 24 hours.

| ID | LEARNA-30MIN | META-LEARNA | MCTS-RNA | RL-LS | RNAINVERSE | ANTARNA |
|---|---|---|---|---|---|---|
| 51 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 52 | - | - | - | - | - | - |
| 53 | - | 5/5 | - | - | - | - |
| 54 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 55 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 56 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 57 | - | - | - | - | - | - |
| 58 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 59 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | - |
| 60 | - | - | - | - | - | - |
| 61 | - | - | - | - | - | - |
| 62 | - | - | - | - | 5/5 | 3/5 |
| 63 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 64 | - | - | - | - | - | - |
| 65 | - | - | 2/5 | - | - | - |
| 66 | 1/5 | - | - | - | 5/5 | 5/5 |
| 67 | - | - | - | - | - | - |
| 68 | - | - | - | - | - | - |
| 69 | 4/5 | 5/5 | 5/5 | - | - | - |
| 70 | 5/5 | 5/5 | 3/5 | - | - | 4/5 |
| 71 | - | - | - | - | - | - |
| 72 | - | - | - | 5/5 | 5/5 | - |
| 73 | - | - | - | - | - | - |
| 74 | - | 2/5 | - | - | - | - |
| 75 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 76 | - | - | - | - | - | - |
| 77 | 2/5 | - | 3/5 | 4/5 | 5/5 | - |
| 78 | - | - | - | - | - | - |
| 79 | - | - | - | - | - | - |
| 80 | - | - | - | - | - | - |
| 81 | - | - | - | - | - | - |
| 82 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 83 | - | | - | - | - | - |
| 84 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 85 | - | - | - | - | - | - |
| 86 | - | - | - | - | - | - |
| 87 | - | - | - | - | - | - |
| 88 | - | - | - | - | - | - |
| 89 | - | - | - | - | - | - |
| 90 | - | - | - | - | - | - |
| 91 | - | - | - | - | - | - |
| 92 | - | - | - | - | - | - |
| 93 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 94 | - | - | - | - | - | - |
| 95 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| 96 | - | - | - | - | - | - |
| 97 | - | - | - | - | - | - |
| 98 | 1/5 | - | - | - | - | - |
| 99 | - | - | - | - | - | - |
| 100 | - | - | - | - | - | - |
| TOTAL | 73/250 | 77/250 | 73/250 | 69/250 | 80/250 | 67/250 |
| SOLVED | 17/50 | 16/50 | 16/50 | 14/50 | 16/50 | 14/50 |

Table 16: Results for 50 independent attempts on each of the 29 target structures of the Rfam-Taneda benchmark. The timeout for each attempt was set to 10 minutes.

| ID | LEARNA-10MIN | META-LEARNA-ADAPT | MCTS-RNA | RL-LS | RNAINVERSE | ANTARNA |
|---|---|---|---|---|---|---|
| 1 | 24/50 | 49/50 | 32/50 | 7/50 | 20/50 | 50/50 |
| 2 | 40/50 | 50/50 | 28/50 | 5/50 | - | - |
| 3 | 1/50 | 28/50 | 4/50 | - | - | - |
| 4 | 48/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 5 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 6 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 7 | 46/50 | 50/50 | 50/50 | 48/50 | 50/50 | 50/50 |
| 8 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 9 | 21/50 | 50/50 | 44/50 | - | - | - |
| 10 | - | - | - | - | - | - |
| 11 | - | - | - | - | - | - |
| 12 | 41/50 | 50/50 | 50/50 | 50/50 | 3/50 | 50/50 |
| 13 | 47/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 14 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 15 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 16 | - | - | - | - | - | - |
| 17 | 29/50 | 50/50 | 47/50 | - | 50/50 | 50/50 |
| 18 | - | 6/50 | - | - | - | - |
| 19 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 20 | - | - | - | - | - | - |
| 21 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 22 | 48/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 23 | - | - | - | - | - | - |
| 24 | 43/50 | 50/50 | 50/50 | 19/50 | - | 50/50 |
| 25 | 49/50 | 50/50 | 50/50 | 50/50 | 46/50 | 50/50 |
| 26 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 27 | 10/50 | 43/50 | 6/50 | - | - | - |
| 28 | 48/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| 29 | 35/50 | 50/50 | 50/50 | - | - | 36/50 |
| TOTAL | 930/1450 | 1126/1450 | 1011/1450 | 779/1450 | 769/1450 | 936/1450 |
| SOLVED | 23/29 | 24/29 | 23/29 | 18/29 | 17/29 | 19/29 |