Towards Automated Design of Riboswitches

Anonymous Authors[1]

Abstract

Experimental screening and selection pipelines for the discovery of novel Riboswitches are expensive, time-consuming, and inefficient. Using computational methods to reduce the number of candidates for the screen could drastically decrease these costs. However, existing computational approaches do not fully satisfy all requirements for the design of such initial screening libraries. In this work, we present a new method, *libLEARNA*, capable of providing RNA focus libraries of diverse variable-length qualified candidates. Our novel structure-based design approach considers global properties as well as desired sequence and structure features. We demonstrate the benefits of our method by designing theophylline riboswitch libraries, following a previously published protocol, and yielding 30% more unique high-quality candidates.

1. Introduction

Riboswitches (Mironov et al., 2002) are regulatory RNA elements, typically located in the 5′ untranslated region of messenger RNAs (mRNAs), that can specifically bind certain metabolites (ligand) to alter gene expression. They consist of an *aptamer* for recognizing the ligand, closely connected to an *expression platform* that couples ligand binding with gene regulation. The binding signal is transduced by a conformational change of the switch. Riboswitches are typically found in bacteria but can also be artificially constructed. Since aptamers are capable of binding nearly every molecular and supramolecular target (Vorobyeva et al., 2018), riboswitches have become interesting tools for different applications including modulation of cellular functions (Hallberg et al., 2017) or the development of biosensors (Findeiß et al., 2017).

However, the discovery of new functional riboswitches and aptamers is a challenging process. Current experimental screening procedures, e.g. systematic evolution of ligands by exponential enrichment (SELEX) (Tuerk & Gold, 1990), use RNA libraries of up to 10^{16} fixed-length random sequences (Vorobyeva et al., 2018) for a single target, resulting in expensive, time-consuming, and inefficient screens. It is common knowledge that the length, the secondary structures, their diversity, and the nucleotide composition of the sequences in the initial library are crucial for a successful SELEX (Vorobyeva et al., 2018; Kohlberger & Gadermaier, 2022). However, random libraries of fixed-length sequences are still most widely used. Focused design can help to decrease the size of the initial library and therefore drastically reduce the costs of these SELEX pipelines.

To tackle the problem of library generation, there exists different computational approaches. While earlier work focused on candidate selection from fixed-length random libraries (Kim et al., 2010; Zhou et al., 2015) based on different features, more recent approaches seek to tackle the problem with generative algorithms, employing recurrent neural networks (Im et al., 2019), Restricted Boltzmann Machines (Di Gioacchino et al., 2022), variational autoencoders (VAE) (Iwano et al., 2022), Monte-Carlo Tree Search(Lee et al., 2021), or evolutionary algorithms (Andress et al., 2022). However, these methods require initial libraries either from previous SELEX data (Di Gioacchino et al., 2022; Iwano et al., 2022), of known good binders (Andress et al., 2022), or even known interaction scores (Im et al., 2019; Lee et al., 2021), which is often unavailable for the discovery of novel aptamers. Further, the design process works with sequence information only, ignoring nucleotide compositions or using secondary structure only to enable docking simulations for scoring.

As an alternative, rational structure-based RNA design algorithms have previously been employed to successfully design riboswitches (Wachsmuth et al., 2012; Ender et al., 2021). The general goal of these algorithms is to find an RNA sequence that folds into a desired secondary structure. However, all structure-based RNA design algorithms are either limited to the design of fixed-length candidates (Hofacker et al., 1994; Andronescu et al., 2004; Taneda, 2010; Kleinkauf et al., 2015; Eastman et al., 2018; Runge et al., 2019) or require a fully defined pairing scheme of the input structure which is a strong restriction on the structural di-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by *the 2023 ICML Workshop on Computational Biology*. Do not distribute.

versity and hinders their applicability to larger scale library designs. This is also true for tools that were specifically developed for the task of riboswitch design, like the sampling approach proposed by Hammer et al. (2017).

In this work, we present a novel structure-based RNA design algorithm, *libLEARNA*, that is capable of designing large amounts of diverse candidate sequences with different lengths, while considering desired sequence and structural constraints, as well as nucleotide distributions, during the design process. In particular, our contributions are as follows:

- We introduce a novel *RNA design paradigm* that enables the design of variable-length RNA sequences from desired arbitrary sequence and structure parts.
- We improve an existing *RNA design algorithm*, *LEARNA* (Runge et al., 2019), with a masked training objective to enable efficient RNA library generation, including global sequence properties due to the robustness of our approach.
- We show the benefits of our approach by exemplarily designing theophylline riboswitch libraries, following a previously proposed protocol by Wachsmuth et al. (2012), yielding 30% more unique candidates compared to the original approach and up to 47% when designing sequences with desired G and C nucleotide ratios, while providing greater structural diversity across nearly uniformly distributed sequence lengths.

2. Method

We first recap the foundation of our method, the *LEARNA* algorithm, followed by the changes we apply to the state representations, the training procedure, and the meta-optimization process which leads to its successor*libLEARNA*.

2.1. Background

LEARNA is a generative automated deep reinforcement learning (AutoRL) (Parker-Holder et al., 2022) algorithm for the inverse RNA folding problem (Hofacker et al., 1994). The algorithm samples RNA sequences from a learned policy given a target secondary structure. In an inner-loop, a deep reinforcement learning (RL) algorithm meta-learns an RNA design policy across thousands of different inverse RNA folding tasks. The validation loss is communicated to an efficient Bayesian Optimization method, BOHB (Falkner et al., 2018), which jointly optimizes the configuration of the RL system in the outer-loop. The actions of the RL agent correspond to placing a nucleotide (A, C, G, or U) for each position, or directly placing Watson-Crick pairs (A-U, U-A, G-C, or C-G) in case the structure indicates that the given position is paired to another nucleotide. States are defined as local representations of the input structure in dot-bracket format (Hofacker et al., 1994) using an n-gram centered around the current position. After applying a folding algorithm to the designed sequence, the reward function is based on the Hamming distance between the folded candidate and the desired structure. Finally, the best-performing configuration on the validation set is evaluated at test time.

2.2. libLEARNA

We improve *LEARNA* to design RNA candidates from arbitrary sequence and structure constraints due to an extended state representation, changes in the training procedure, and additional dimensions, as well as changes to the general objective of the meta-optimization process.

State Representations To inform *libLEARNA* about constraints in the sequence and the structure, we extend the state representations of *LEARNA*. In particular, we numerically encode pairs of a sequence symbol and its corresponding structure symbol for each position of the design task. This enlarges the state space of *libLEARNA* compared to *LEARNA* but allows *libLEARNA* to be more flexible in terms of design tasks it can be applied to.

Training Instead of training on tasks of the inverse RNA folding problem, we use a masked training objective similar to the masked language model training in BERT (Devlin et al., 2018). In particular, we follow Runge et al. (2019) to generate three training datasets with different length distributions ($\leq$ 200 nucleotides (nt), $\geq$ 200 nt, random length) of 100000 samples each, and a non-overlapping validation set of 100 samples from the Rfam (Griffiths-Jones et al., 2003) database version 14.1 using *RNAfold* (Lorenz et al., 2011) for folding the sequences. However, in contrast to Runge et al. (2019) we do not mask the entire sequences but derive tasks that correspond to the design of RNAs from desired sequence and structure parts by applying a random masking procedure to the sequences and the structures detailed in Appendix A. The task of *libLEARNA* during training then is to fill the masked parts of the sequence such that, after applying a folding algorithm to the designed sequence, all positions of the resulting folding satisfy the given positional constraints of the masked structure. Similar to Runge et al. (2019), the reward is based on the Hamming distance, while masked positions in the structure are ignored. Our design procedure describes a completely new structure-based RNA design paradigm since the design algorithm is not informed about the pairing conditions *a priori*. This is in contrast to previous work in the field of structure-based RNA design and enables the design of RNAs from arbitrary sequence and structure parts while creating larger structural diversity and further allowing us to extend the task at any given point since we are no longer bound to explicit pairing positions. However, we allow indexing pairs to explicitly indicate pairing positions if desired. More details and examples of tasks

Table 1. **Originally proposed theophylline riboswitch constructs and partial RNA design space formulation.** (Top) The sequence parts of the six originally proposed riboswitch constructs by Wachsmuth et al. (2012) and the sequence part of the design space. (Bottom) The corresponding structure parts of the constructs and the structural part of the design space. Red: TCT8-4 theophylline aptamer; green: variable length spacer domain; blue: domain ought to pair with the aptamer (complementary to the 3′-end of the aptamer sequence in the original design); black: 8-U-stretch. Masked positions are indicated with ?, positions for extensions are indicated with $\hat{?}$.

CONSTRUCT	APTAMER	SPACER	COMPLEMENTARY REGION	8-U-STRETCH
RS1 SEQUENCE	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	UUACAUC	UGAAGUGCUGCC	UUUUUUUU
RS2 SEQUENCE	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	UGAUCUCGCU	UGAAGUGCUGC	UUUUUUUU
RS3 SEQUENCE	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	UUUACAUACUCGGUAAAC	UGAAGUGCUGCCA	UUUUUUUU
RS4 SEQUENCE	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	AACCGAAAUUUGCGCU	UGAAGUGCUGC	UUUUUUUU
RS8 SEQUENCE	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	CUCCUAGUGGGAG	UGAAGUGCUG	UUUUUUUU
RS10 SEQUENCE	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	GAAAUCUC	UGAAGUGCUG	UUUUUUUU
TASK SEQUENCE PARTS	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	$\hat{?}$??????$\hat{?}$	UGAAGUGCUG$\hat{?}$	UUUUUUUU
RS1 STRUCTURE	..........(((((.....)))))....(((((((((((	.....	))))))))))	.....
RS2 STRUCTURE	..........(((((.....)))))....(((((((((((	.....	))))))))))	.....
RS3 STRUCTURE	..........(((((.....)))))...(((((((((((	((((((.....)))))).	)))))))))))	.....
RS4 STRUCTURE	..........(((((.....)))))....(((((((((((	(..((.....)).)	))))))))))	.....
RS8 STRUCTURE	........(((((((.....)))))...)))((((((((((	((((....))))	))))))))))	.....
RS10 STRUCTURE	........(((((((.....)))))...)))((((((((((	((....)	))))))))))	.....
TASK STRUCTURE PARTS	........???(((((.....)))))...???((((((((((	$\hat{?}$??....$\hat{?}$	)))))))))$\hat{?}$	?.......

for the datasets can be found in Appendix A.

Meta-Optimization For *libLEARNA*, we mainly adopt the configuration space proposed by Runge et al. (2019) but introduce four new dimensions. (1) While algorithms for the inverse RNA folding problem typically benefit from directly predicting Watson-Crick base pairs at sites that are known to be paired with another nucleotide, the pairing partners for many paired sites are not trivially distinguishable as a result of our masking procedure during training. However, we include the choice to make use of the direct prediction of pairs if the pairing partner can be identified, i.e. there is no masking between the pairing positions. (2) We add a choice to the configuration space to dynamically adapt the states based on the design progress to inform the agent about its own decisions. (3) The choice of training data and (4) the schedule of the tasks during training can have a strong impact on the final performance of the learning algorithm. We, therefore, include two dimensions to choose from three different training data distributions and curricula (unsorted tasks or sorted by length). We further allowed searching over an additional LSTM layer. The result is an 18-dimensional search space to jointly optimize over the network architecture, all parts of the MDP, as well as training hyperparameters, task distributions, and schedule, using BOHB (Falkner et al., 2018). We use the exact same setup during meta-optimization as Runge et al. (2019), with the same training budgets and validation protocols. However, while Runge et al. (2019) optimized for an RL algorithm without any policy updates at test time, we directly optimize for an algorithm with policy updates at evaluation time to increase the adaptation capabilities of our approach. The meta-optimization procedure, the configuration space, as well as the final configuration of *libLEARNA* are detailed in Appendix B.

3. Riboswitch Library Generation

After describing the procedure of the original design and evaluation of candidates for synthetic riboswitches for theophylline-dependent regulation of transcription proposed by Wachsmuth et al. (2012) we detail our approach for the construction of an RNA design space and the design of RNA libraries and evaluate both methods.

3.1. Original Setup

Initial Setting Originally, Wachsmuth et al. (2012) constructed riboswitch candidates from (1) the TCT8-4 theophylline aptamer sequence and structure, (2) a spacer sequence of 6 to 20 nucleotides (nt), (3) a sequence of 10nt to 21nt complementary to the 3′-end of the aptamer, and (4) a U-stretch of 8nt at the 3′-end of the construct.

Design Procedure To generate candidates, Wachsmuth et al. (2012) designed a large library of random sequences for the spacer region (6-20nt) and a library of sequences complementary to the 3′-end of the aptamer (10-21nt). From these sets, randomly sampled sequences were combined with the aptamer and the 8-U-stretch.

3.2. libLEARNA Setup

Design Space Formulation We start the formulation of our design space from the entire sequence and the structure part of the unbound aptamer, a spacer region of unknown sequence, a region complementary to the 3′-end of the aptamer of at least 10nt and the unpaired 8-U-stretch, similar to Wachsmuth et al. (2012). To ensure a length of 6nt of the spacer region, we use the shared structure constraints of the six final constructs proposed by Wachsmuth et al. (2012), which were also used to restrict the unknown parts

Table 2. **Overview of candidates that satisfy the design criteria.** All numbers are averages across five runs with different random seeds.

METHOD	VALID CANDIDATES [%]	UNIQUE STRUCTURES
WACHSMUTH ET AL. (2012)	41,7	6316.6
libLEARNA	**70,9**	**8269.6**

of the aptamer. We then introduce three positions where the task can be extended (indicated by ? in Table 1) to fit the requirements and fix the length of the design space to 66nt-91nt according to Wachsmuth et al. (2012). The final definition of the design space and the proposed constructs of Wachsmuth et al. (2012) are shown in Table 1.

Design Procedure The input to *libLEARNA* is the design space shown in Table 1. To generate candidates, *libLEARNA* internally first samples a masked task from the design space by uniformly sampling a sequence of masking tokens for each extension position considering the length constraints and then tries to solve the task with a single shot.

3.3. Experiments

We assess the performance of *libLEARNA* against the originally proposed library generation procedure proposed by Wachsmuth et al. (2012) in two experiments: (1) The design of a library based on sequence and structure constraints only, and (2) the design of candidates when additionally querying *libLEARNA* to design candidates with a specific G and C nucleotide ratio (GC-content), given a tolerance of 0.01. We note that *libLEARNA* was never trained for predictions with desired GC-contents but that a GC-content loss-term (the absolute deviation of the GC-content of the current candidate sequence from the desired GC-content) was simply added to the reward function of *libLEARNA*. The reward function then is a weighted sum (without any tuning, setting all weights to 1) of the structure- and the GC-loss. However, similar to the structural improvement step described by Runge et al. (2019), we implement a GC-improvement step to guide the agent for this challenging task, detailed in Appendix C. For each experiment, we generate 50000 candidates with the approach of Wachsmuth et al. (2012) and *libLEARNA* and evaluate them as follows.

Evaluation The evaluation of the designed candidates was performed following Wachsmuth et al. (2012): after dropping duplicates, the designed sequences were folded using *RNAfold* and verified for (1) the existence of two hairpin structure elements, the aptamer hairpin for binding the ligand and the terminator hairpin formed between the 3′-end of the aptamer, the spacer and the region complementary to the aptamer sequence, (2) no pairing within the last seven nucleotides of the 8-U-stretch, (3) no pairing between the spacer and the aptamer in a sequence of folding steps, simulating co-transcriptional folding with a fixed elongation speed of five. A candidate that did not pass all criteria was rejected.

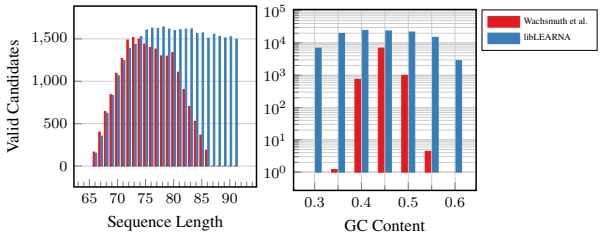

Figure 1. **Length and GC-content distribution in the generated libraries.** (Left) The plot displays the distribution of the candidates across different lengths. (Right) The plot shows the number of unique candidates that satisfy all design criteria and have the specified G and C nucleotide distribution when running *libLEARNA* with a desired GC-content. All numbers are averages across five runs with different random seeds.

Results We observe that *libLEARNA* generates considerably more candidates that pass the design criteria compared to the original procedure proposed by Wachsmuth et al. (2012), yielding 30% more satisfying candidates on average (Table 2). Further, the candidates are nearly uniformly distributed across the lengths of the design space, especially for the longer sequences (Figure 1 left), and the structure diversity generated by *libLEARNA* is around 23% higher (Table 2). When designing candidates with desired GC-contents, *libLEARNA* provides up to 47% more candidates that satisfy the design criteria and contain the specific G and C nucleotide distribution (Figure 1 right). Remarkably, *libLEARNA* can also design such candidates on the margins of possible GC-contents (0.3 and 0.6) which lay between 0.29 and 0.63 for the given riboswitch design space.

4. Conclusion

We propose a new RNA design paradigm based on a masked prediction task that enables the design of RNA libraries with large structure diversity from arbitrary sequence and structure constraints. We use the paradigm and develop a new algorithm, *libLEARNA*, capable of generating these libraries. We exemplarily demonstrate the benefits of our method on the design of variable-length riboswitches, showing the efficiency of our approach. Our results for library generation with desired G and C nucleotide distributions show that *libLEARNA* can handle global constraints, while the robustness to reward changes suggests potential for handling more constraints, e.g. a desired energy threshold of the structures, which we will assess in future work. Overall, our approach bears great potential to support experimental pipelines.

References

Andress, C., Kappel, K., Cuperlovic-Culf, M., Yan, H., and Li, Y. Daptev: Deep aptamer evolutionary modelling for covid-19 drug design. *bioRxiv*, pp. 2022–11, 2022.

Andronescu, M., Fejes, A. P., Hutter, F., Hoos, H. H., and Condon, A. A new algorithm for rna secondary structure design. *Journal of molecular biology*, 336(3):607–624, 2004.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Di Gioacchino, A., Procyk, J., Molari, M., Schreck, J. S., Zhou, Y., Liu, Y., Monasson, R., Cocco, S., and Šulc, P. Generative and interpretable machine learning for aptamer design and analysis of in vitro sequence selection. *PLoS computational biology*, 18(9):e1010561, 2022.

Eastman, P., Shi, J., Ramsundar, B., and Pande, V. S. Solving the rna design problem with reinforcement learning. *PLoS computational biology*, 14(6):e1006176, 2018.

Ender, A., Etzel, M., Hammer, S., Findeiß, S., Stadler, P., and Mörl, M. Ligand-dependent trna processing by a rationally designed rnase p riboswitch. *Nucleic acids research*, 49(3):1784–1800, 2021.

Falkner, S., Klein, A., and Hutter, F. BOHB: Robust and efficient hyperparameter optimization at scale. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1437–1446, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Findeiß, S., Etzel, M., Will, S., Mörl, M., and Stadler, P. F. Design of artificial riboswitches as biosensors. *Sensors*, 17(9):1990, 2017.

Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., and Eddy, S. R. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439–441, 01 2003. ISSN 0305-1048.

Hallberg, Z. F., Su, Y., Kitto, R. Z., and Hammond, M. C. Engineering and in vivo applications of riboswitches. *Annual review of biochemistry*, 86:515–539, 2017.

Hammer, S., Tschiatschek, B., Flamm, C., Hofacker, I. L., and Findeiß, S. Rnablueprint: flexible multiple target nucleic acid sequence design. *Bioinformatics*, 33(18): 2850–2858, 2017.

Hofacker, I., Fontana, W., Stadler, P., Bonhoeffer, S., Tacker, M., and Schuster, P. Fast Folding and Comparison of RNA Secondary Structures. *Monatshefte fuer Chemie/-Chemical Monthly*, 125:167–188, 02 1994.

Im, J., Park, B., and Han, K. A generative model for constructing nucleic acid sequences binding to a protein. *BMC genomics*, 20(13):1–13, 2019.

Iwano, N., Adachi, T., Aoki, K., Nakamura, Y., and Hamada, M. Generative aptamer discovery using raptgen. *Nature Computational Science*, 2(6):378–386, 2022.

Kim, N., Izzo, J. A., Elmetwaly, S., Gan, H. H., and Schlick, T. Computational generation and screening of rna motifs in large nucleotide sequence pools. *Nucleic acids research*, 38(13):e139–e139, 2010.

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

Kohlberger, M. and Gadermaier, G. Selex: Critical factors and optimization strategies for successful aptamer selection. *Biotechnology and Applied Biochemistry*, 69(5): 1771–1792, 2022.

Lee, G., Jang, G. H., Kang, H. Y., and Song, G. Predicting aptamer sequences that interact with target proteins using an aptamer-protein interaction classifier and a monte carlo tree search approach. *PloS one*, 16(6):e0253760, 2021.

Lorenz, R., Bernhart, S. H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6 (1):26, Nov 2011. ISSN 1748-7188.

Mironov, A. S., Gusarov, I., Rafikov, R., Lopez, L. E., Shatalin, K., Kreneva, R. A., Perumov, D. A., and Nudler, E. Sensing small molecules by nascent rna: a mechanism to control transcription in bacteria. *cell*, 111(5):747–756, 2002.

Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen, V., Calandra, R., Faust, A., Hutter, F., and Lindauer, M. Automated reinforcement learning (autorl): A survey and open problems. *Journal of Artificial Intelligence Research (JAIR)*, 74:517–568, 2022.

Runge, F., Stoll, D., Falkner, S., and Hutter, F. Learning to design RNA. In *International Conference on Learning Representations*, 2019.

Taneda, A. Modena: a multi-objective rna inverse folding. *Advances and Applications in Bioinformatics and Chemistry*, pp. 1–12, 2010.

Tuerk, C. and Gold, L. Systematic evolution of ligands by exponential enrichment: Rna ligands to bacteriophage t4 dna polymerase. *science*, 249(4968):505–510, 1990.

Vorobyeva, M. A., Davydova, A. S., Vorobjev, P. E., Pyshnyi, D. V., and Venyaminova, A. G. Key aspects of nucleic acid library design for in vitro selection. *International journal of molecular sciences*, 19(2):470, 2018.

Wachsmuth, M., Findeiß, S., Weissheimer, N., Stadler, P. F., and Mörl, M. De novo design of a synthetic riboswitch that regulates transcription termination . *Nucleic Acids Research*, 41(4):2541–2551, 12 2012. ISSN 0305-1048.

Zhou, Q., Xia, X., Luo, Z., Liang, H., and Shakhnovich, E. Searching the sequence space for potent aptamers using selex in silico. *Journal of chemical theory and computation*, 11(12):5939–5946, 2015.

A. Data

Table 3. **Overview of the training and validation sets.**

DATA SET	TASKS	MEAN LENGTH	MEDIAN LENGTH	MIN/MAX LENGTH
TRAINING1 ("LONG")	100000	470.9	300	200–8033
TRAINING2 ("SHORT")	100000	103.9	99	23–200
TRAINING3 ("RANDOM")	100000	142.7	106	23–6361
VALIDATION	100	135.8	94	46–1802

Following Runge et al. (2019), we generate training data from the Rfam database version 14.1 by folding all sequences using *RNAfold*. We build a total of three training sets of 100000 samples each with different length distributions, and a non-overlapping validation set of 100 samples. The datasets are described in Table 3. Instead of training for the task of inverse RNA folding only by masking the sequences of the samples only, we first mask up to five parts of the structures, each covering up to 20% of the total length, while the positions, the lengths, and the number of parts are sampled uniformly at random. In the second step, we mask corresponding parts of the sequences that remain unmasked in the structures to derive tasks of alternating sequences and structure constraints. Finally, we randomly mask the sequences of ∼20% of the samples to derive tasks that correspond to RNA design from arbitrary sequence and structure parts. Examples of the resulting training tasks are shown in Table 4.

Table 4. **Examples of different tasks in the training data.**

TASK DESCRIPTION	TASK SPACE	EXAMPLE	PERCENTAGE OF DATA
INVERSE RNA FOLDING	STRUCTURE	`...........((((....))))......`	11,5
	SEQUENCE	`??????????????????????????????`	
ALTERNATING CONSTRAINTS	STRUCTURE	`?....???.?.((((...??????????`	66,7
	SEQUENCE	`C????UAU?C????????UCAGAUAAAAC`	
RANDOM MASKING	STRUCTURE	`?....???.?.((((...??????????`	21,8
	SEQUENCE	`C????U?U?C????????UCAGA????AC`	

B. Meta-Optimization

We use an automated deep reinforcement learning (AutoRL) approach that automatically selects the best RL setting for *libLEARNA*, given a rich configuration space as proposed by Runge et al. (2019). In particular, we use a meta-optimization process to jointly optimize the formulation of our RL algorithm: in the outer loop, the meta-learner iteratively samples a configuration that defines an RL algorithm, which is then used to learn an RNA design policy in the inner loop. The resulting policy is evaluated on a validation data set and the meta-learner observes the validation loss to update its own model accordingly. The meta-optimization loop for *libLEARNA* is illustrated in Figure 2.

For *libLEARNA*, we mainly adopt the configuration space proposed by Runge et al. (2019) but introduce four novel dimensions, described in the following.

Action Semantics Inverse RNA folding algorithms typically benefit from directly predicting Watson-Crick base pairs at paired sites. For *libLEARNA*, however, many paired sites are not trivially distinguishable and we included a dimension via the *action semantics* parameter to either use direct prediction of pairs or not, to potentially exploit predictions of non-Watson-Crick base pairs.

Training Data The choice of training data and the schedule of the tasks during training can have a strong impact on the final performance of the learning algorithm. We, therefore, include two dimensions to choose from different training data distributions (*training data* parameter) and curricula (*curriculum* parameter).

Individual State Composition The predictions of the RL agent are based on the presented states. While the formulation of individual states in Runge et al. (2019) was based only on the provided target structure, we decide to add a choice to

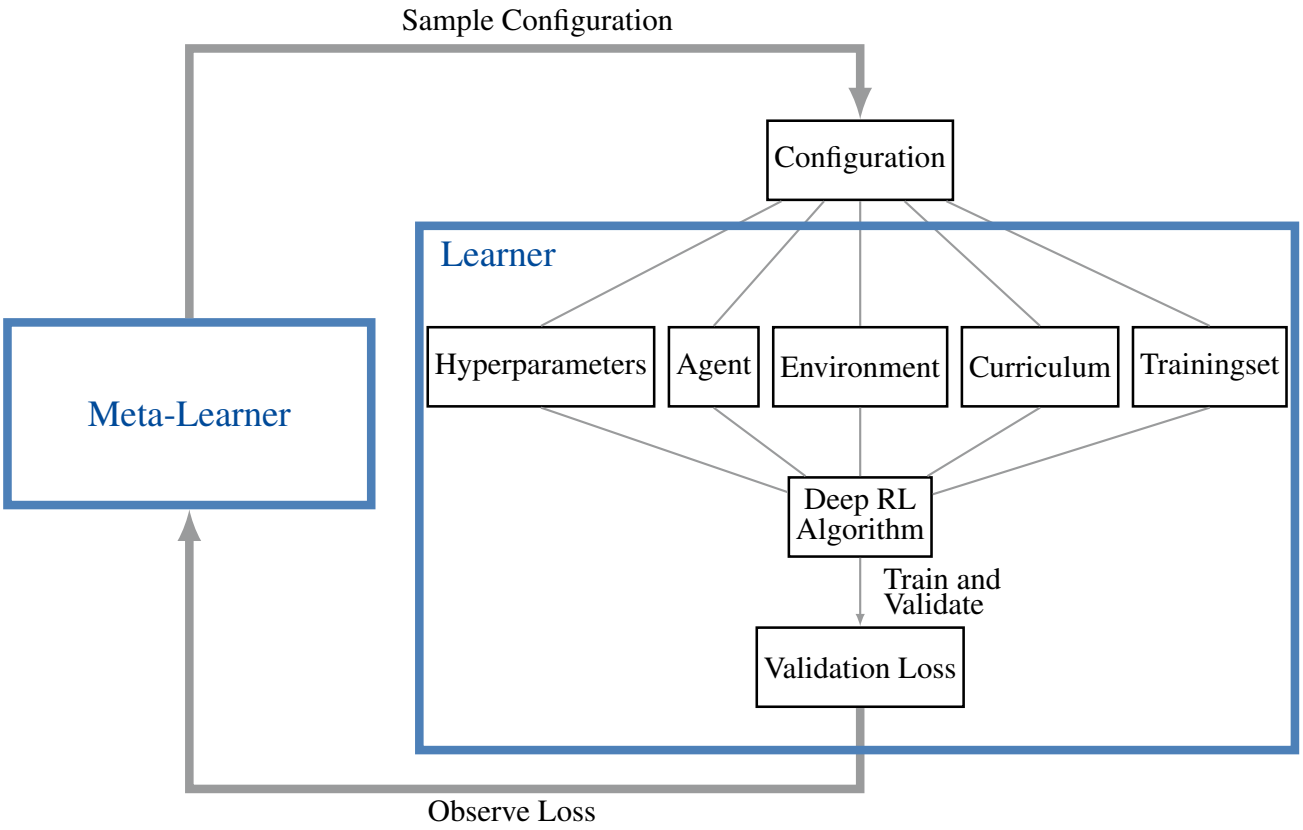

Figure 2. **Meta-optimization loop.** In each iteration, the meta-learner samples a configuration from a rich configuration space. The sampled configuration defines the learning algorithm's hyperparameters, a specific environment, a particular network architecture of the agent, a training curriculum, and a training data set. These components together formulate the learner; a deep reinforcement learning algorithm, which is trained on the selected training set and evaluated on a validation set. The resulting validation loss is communicated to the meta-learner to update its model, seeking to learn to sample better configurations with each iteration.

formulate states that provide the agent with information about its own decisions. We use the *state composition* parameter σ to choose either to define states based on the target task or to define states using the design process of the agent (updating the representation at each time step).

Overall, our design choices yield an 18-dimensional configuration space, encompassing a broad range of neural architectures to formulate the agent (including elements of recurrent neural networks (RNNs) and convolutional neural networks (CNNs)), a variety of different environment formulations, three distinct training data distributions, two training curricula, and training hyperparameters. The complete list of parameters, their types, ranges, and the priors we used over them as well as the final selected configuration of *libLEARNA* are listed in Table 5.

C. Improvement Steps

LEARNA employs a local improvement step (LIS) to exhaustively search over neighboring sequences if the agent is close to a solution for a given task to account for the stochasticity of the agent. We adopt this procedure to improve the structure loss for a given task. During evaluations, we add a desired GC-content to the task. We, therefore, develop a GC-improvement step (GIS) to guide the agent, either increasing or decreasing the GC content of the designed sequence by replacing nucleotides at random positions with their respective counterparts (either A or U by G or C if the GC-content is too low, or vice-versa). The GIS becomes active whenever the LIS becomes active and changes to the sequence are only applied if the structure loss is at least as low as before the GIS.

Table 5. **Configuration space and finally selected parameters of *libLEARNA*.**

PARAMETER NAME	TYPE	RANGE	PRIOR	zsLEARNA
STATE RADIUS κ	INTEGER	$[0, 32]$	UNIFORM	10
INDIVIDUAL STATE COMPOSITION σ	CATEGORICAL	[“TARGET”, “DESIGN”]	UNIFORM	“TARGET”
ACTION SEMANTICS	CATEGORICAL	[“PAIR”, “SINGLE”]	UNIFORM	“PAIR”
REWARD EXPONENT α	FLOAT	$[1, 12]$	UNIFORM	10.76
FILTER SIZE IN 1ST CONV LAYER	INTEGER	$\{0\} \cup \{3, 5, \ldots, 17\}$	UNIFORM	0
FILTER SIZE IN 2ND CONV LAYER	INTEGER	$\{0, 3, 5, 7, 9\}$	UNIFORM	0
# FILTER IN 1ST CONV LAYER	INTEGER	$[1, 32]$	LOG-UNIFORM	17
# FILTER IN 2ND CONV LAYER	INTEGER	$[1, 32]$	LOG-UNIFORM	24
# LSTM LAYERS	INTEGER	$[0, 3]$	UNIFORM	0
# UNITS IN EVERY LSTM LAYER	INTEGER	$[1, 64]$	LOG-UNIFORM	20
# FULLY CONNECTED LAYERS	INTEGER	$[1, 2]$	UNIFORM	2
# UNITS IN FULLY CONNECTED LAYERS	INTEGER	$[8, 64]$	LOG-UNIFORM	12
EMBEDDING DIMENSIONALITY	INTEGER	$[0, 8]$	UNIFORM	17
BATCH SIZE	INTEGER	$[32, 256]$	LOG-UNIFORM	247
ENTROPY REGULARIZATION	FLOAT	$[1 \cdot 10^{-7}, 1 \cdot 10^{-2}]$	LOG-UNIFORM	$4.46 \cdot 10^{-7}$
LEARNING RATE FOR PPO	FLOAT	$[1 \cdot 10^{-6}, 1 \cdot 10^{-3}]$	LOG-UNIFORM	$5.9 \cdot 10^{-4}$
TRAINING DATA	CATEGORICAL	[“RANDOM”, “SHORT”, “LONG”]	UNIFORM	“SHORT”
TRAINING CURRICULUM	CATEGORICAL	[“RANDOM”, “SORTED”]	UNIFORM	“SORTED”