PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

EternaBrain: Automated RNA design through move sets and strategies from an Internet-scale RNA videogame

Rohan V. Koodli[1¤], Benjamin Keep[2], Katherine R. Coppess[3], Fernando Portela[1], Eterna participants[¶], Rhiju Das[1,3*]

1 Department of Biochemistry, Stanford University School of Medicine, Stanford, CA, United States of America, **2** Department of Education, Stanford University, Stanford, CA, United States of America, **3** Department of Physics, Stanford University, Stanford, CA, United States of America

¤ Current address: University of California, Berkeley, CA.
¶ Membership of Eterna participants is provided in Acknowledgments and Supplemental S1 File.
* rhiju@stanford.edu

OPEN ACCESS

Citation: Koodli RV, Keep B, Coppess KR, Portela F, Eterna participants, Das R (2019) EternaBrain: Automated RNA design through move sets and strategies from an Internet-scale RNA videogame. PLoS Comput Biol 15(6): e1007059. https://doi.org/10.1371/journal.pcbi.1007059

Editor: Shi-Jie Chen, University of Missouri, UNITED STATES

Received: March 18, 2019

Accepted: April 30, 2019

Published: June 27, 2019

Data Availability Statement: Data are available in a Github repository: https://github.com/EteRNAgame/EternaBrain.

Funding: We acknowledge funding from a Stanford Graduate Fellowship (to B.K.), the U.S. National Institutes of Health (R01 GM100953 and R35 GM122579 to R.D.), and a Discovery Innovation Award (Stanford University School of Medicine to R.D.). The funder websites are: https://vpge.stanford.edu/fellowships-funding/sgf https://www.nigms.nih.gov/ http://medicalgiving.stanford.edu/

Abstract

Emerging RNA-based approaches to disease detection and gene therapy require RNA sequences that fold into specific base-pairing patterns, but computational algorithms generally remain inadequate for these secondary structure design tasks. The Eterna project has crowdsourced RNA design to human video game players in the form of puzzles that reach extraordinary difficulty. Here, we demonstrate that Eterna participants' moves and strategies can be leveraged to improve automated computational RNA design. We present an eternamoves-large repository consisting of 1.8 million of player moves on 12 of the most-played Eterna puzzles as well as an eternamoves-select repository of 30,477 moves from the top 72 players on a select set of more advanced puzzles. On eternamoves-select, we present a multilayer convolutional neural network (CNN) EternaBrain that achieves test accuracies of 51% and 34% in base prediction and location prediction, respectively, suggesting that top players' moves are partially stereotyped. Pipelining this CNN's move predictions with single-action-playout (SAP) of six strategies compiled by human players solves 61 out of 100 independent puzzles in the Eterna100 benchmark. EternaBrain-SAP outperforms previously published RNA design algorithms and achieves similar or better performance than a newer generation of deep learning methods, while being largely orthogonal to these other methods. Our study provides useful lessons for future efforts to achieve human-competitive performance with automated RNA design algorithms.

Author summary

The design of RNA sequences that fold into target structures is a computationally difficult task whose importance continues to grow with the advent of RNA-based therapeutics and diagnostics. This paper reports a new approach stemming from the Eterna massive open laboratory, a project that crowdsources RNA design to >250,000 'players' on the internet.

events/discovery-innovation-awards.html/ The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

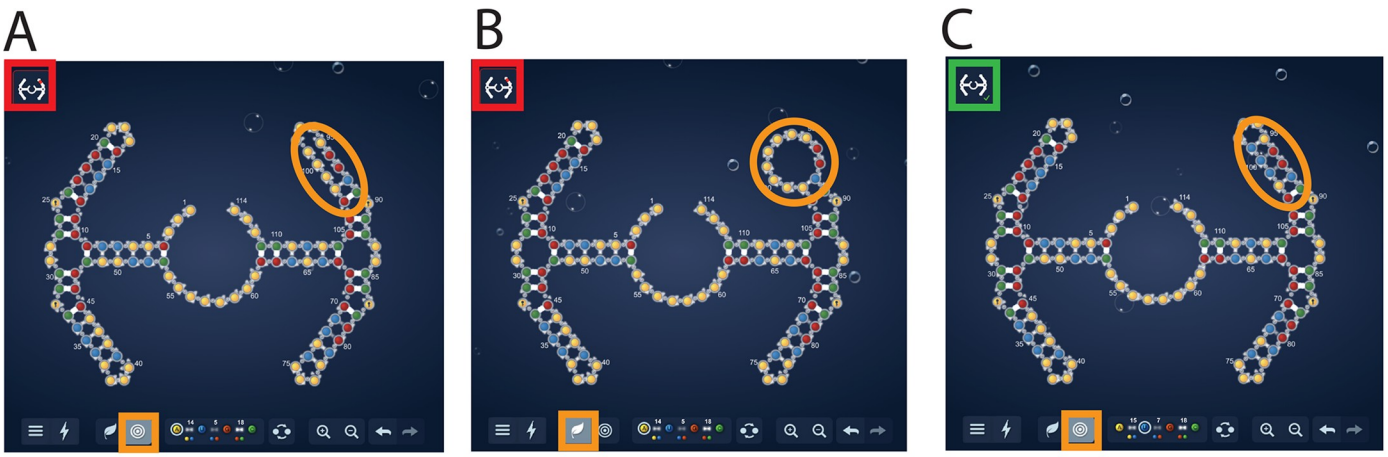
The efforts of Eterna participants have led to the accumulation of nearly 2 million moves that lead to successful *in silico* solutions on difficult puzzles, many of which are only solvable by humans. Inspired by recent advances in automated game playing, we discovered that these moves are sufficiently stereotyped so that a neural network can be trained to predict moves with accuracy significantly higher than random guessing. The resulting method EternaBrain allows solution of new RNA design problems when used to predict complete series of moves rather than just single moves. Further improvement comes from heuristic strategies that are well known amongst the Eterna community but not described in prior publications on automated RNA design. EternaBrain appears highly complementary to other emerging next-generation RNA design methods based on neural-network and game playing approaches, suggesting new routes for automated methods to emulate human experts in RNA design.

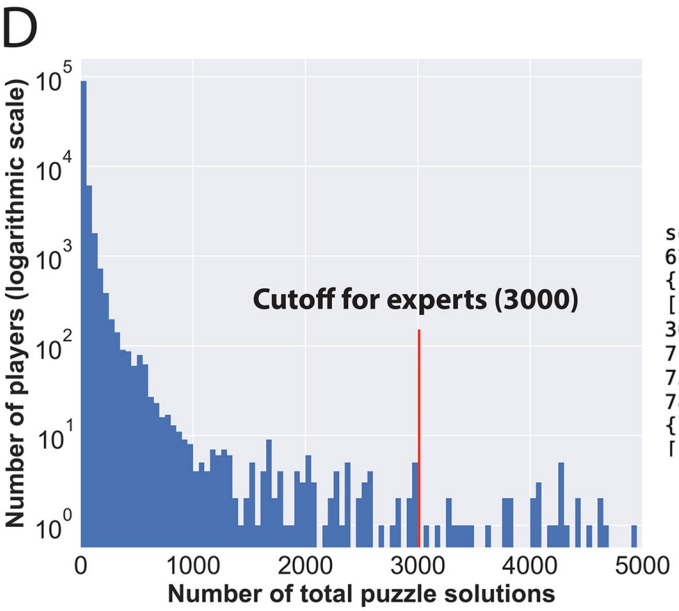
Introduction

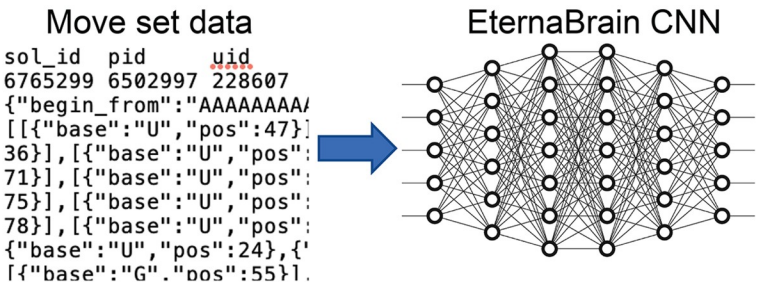
Due to its versatility and important roles throughout biology, there is strong interest in designing RNA-guided machines for disease detection, virus defense, and gene therapy, e.g., for gene silencing and CRISPR/Cas9 gene editing [1][2]. Much of RNA's functionality is dependent on well-defined structures, and so these and future RNA technologies require computational methods to effectively design sequences that fold into a target structure or set of structures suited to a desired task. The simplest problem involves designing RNA sequences that energetically favor one specific secondary structure–a target pattern of Watson-Crick base pairs–over alternative secondary structures. Even this most basic problem is computationally difficult [3]. While exact solutions can be determined through exhaustive calculation [4], such computational enumeration generally takes an impractically long time for solve complex target structures.

Numerous groups have developed RNA secondary structure design algorithms, including MODENA [5], RNAinverse [6], INFO-RNA [7], RNA-SSD [8], and NUPACK [9]. In the original studies presenting these methods, tests typically involved simple structures that do not capture the symmetries, duplex lengths, and sizes needed for biotechnology applications [10]. The incompleteness of these prior methods and tests became clear with the release of the *Eterna* game [11] in 2011, which crowdsources RNA design in the form of puzzles through an internet-scale videogame (Fig 1A–1C). Eterna participants learn the basics of RNA design through examples that are initially tested *in silico* through a computational model of secondary structure folding. Advanced participants can submit their RNA designs in lab challenges and receive wet-lab feedback on how their molecules fold during *in vitro* experiments performed on a weekly time scale. These efforts expose the 'reality gap' in RNA design–the mismatch between current computational folding models and experiment. In preparation for these experimental challenges, participants also challenge fellow participants through *in silico* puzzles that require learning or developing sophisticated puzzle-solving strategies; these separate 'games within the game' are useful for guiding the development of computational RNA design methods [10]. Since Eterna's inception, the community has grown to over 250,000 registered participants as of 2019, with over 17,000 player-created puzzles. This community has been successful in designing RNAs that consistently outperform RNA design algorithms in both *in silico* and *in vitro* tests [10].

Since Eterna's inception, participants have discovered classes of RNA secondary structures for which prior algorithms cannot find sequence solutions even *in silico*, i.e., when folded with







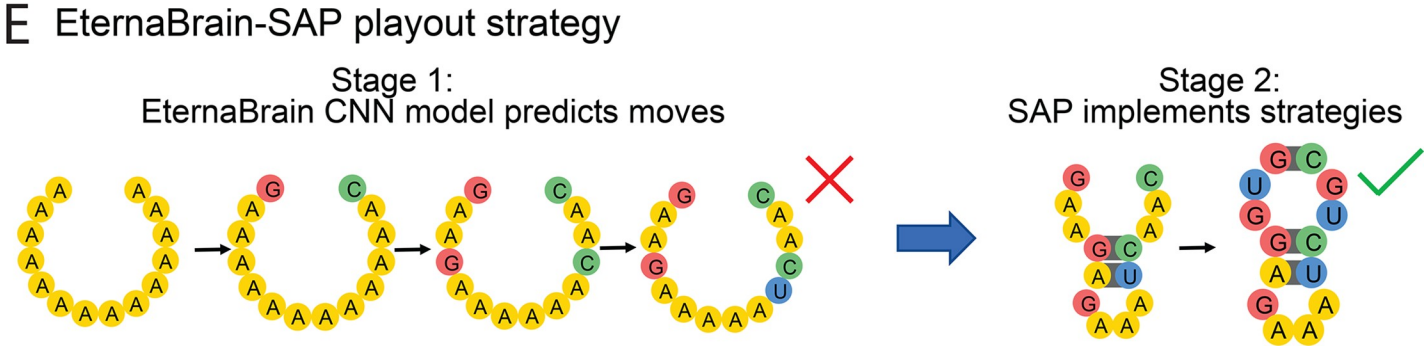


Fig 1. Eterna and EternaBrain. (A-C) Puzzle-solving interface presented to human players of Eterna including the state of the puzzle (whether it is solved or not) in the top left corner (red/green outline), the puzzle itself (in the middle), and the toolbar (bottom) with which the players can mutate the RNA sequence to make it fold into the desired state; yellow, blue, red, and green symbols represent A, U, G, and C nucleotides. (A) The desired target structure for the RNA molecule, as indicated by the bullseye in the bottom left (orange highlight). (B) Nature mode, as indicated by the leaf in the bottom left (orange highlight), gives the predicted minimum free energy structure for the current sequence. Since the bases in the top right should be paired with each other (orange circle), this puzzle is not yet folding correctly; this status is shown by the red indicator in the top left corner. (C) The solved puzzle. The nature-mode structure matches the target structure, and the indicator in the top left corner

turns green, meaning the puzzle has been solved. (D) (left) Wide distribution of contributed Eterna solutions across different players. For preparing the *eternamoves-select* data set, we selected any player who had solved more than 3000 distinct puzzles, which left us with 72 players. (right) In EternaBrain, we tested whether information on players' moves could be used to train a convolutional neural network. (E) For solving new puzzles, the final EternaBrain-SAP framework first uses the EternaBrain convolutional neural net model to predict sequence changes ('moves') for new RNA puzzles. In a second stage, the Single Action Playout (SAP), six additional hand-coded strategies are applied to complete the solution.

https://doi.org/10.1371/journal.pcbi.1007059.g001

computational energy models that can be rapidly evaluated [10]. However, under the same computational energy models, solutions to these design problems can be discovered by experienced human participants. Thus, there remains a gap between algorithms and humans even for the purely computational problem of *in silico* design. Fortunately, Eterna has produced rich resources that might allow for this gap to be closed. First, in 2016, several participants curated a benchmark of 100 problems of increasing difficulty, termed the 'Eterna100' [10]. By offering a wider spectrum of difficulty than prior benchmark sets based on inferred structures of random or biological RNAs [5][6][7][8][9], these secondary structures allow stringent tests of advances in computational design methods while still guaranteeing that solutions can be found. Second, player-created tutorial puzzles have 'canonized' new strategies for *in silico* RNA design. In 2016, Eterna developers installed these puzzles as the standard progression of problems for new participants. At the same time, Eterna participants have agreed to share the sequences of moves that lead to successful solutions to scientific research, resulting in a data repository including nearly 2 million player moves (Fig 1D).

The availability of such a large collection of move sets as well as canonical player strategies suggests new approaches to solving the RNA secondary structure design problem. While previous RNA design algorithms have used hierarchical decomposition of target structures, genetic algorithms, and probabilistic sampling of sequences [5][7][8], a recent generation of classification, translation, and game-playing algorithms make powerful use of statistical pattern recognition through multi-layer artificial neural networks [12]. One striking example is Google DeepMind's AlphaGo [13], which outcompeted expert human participants in the complex game of *Go*. This approach was inspired by the discovery that expert participant moves in the game Go were sufficiently stereotyped that they could be predicted with better than 50% accuracy by a convolutional neural network (CNN), despite the large space of possible moves in this game (up to 19 x 19 = 361 board positions for pieces, giving a baseline random guess accuracy of less than 0.5%) [13]. AlphaGo and its successor AlphaGo Zero have further improved their performance using reinforcement learning [14]. While reinforcement learning has shown promise for RNA design, it is not yet human competitive [15][16]. This observation suggests that there remain lessons to be learned from human moves and strategies. Nevertheless, it remains unclear whether single base changes during human RNA design are sufficiently stereotyped as to inform neural network or other machine learning methods– strategies that involve back-and-forth flipping of base changes, sets of multiple sequence moves, or sequence explorations outside the game browser may be important for the success of the best Eterna participants.

To test whether new strategies for automated RNA design might be gleaned from successful single base moves, we present a data set of 1.8 million moves on Eterna's most played puzzles, called *eternamoves-large*, appropriately cleaned and labeled for machine learning applications. We conduct tests of CNNs to predict these moves given the game state, and report generally poor results. However, we do find that a set of 30,477 moves made by the most experienced participants on more difficult puzzles (*eternamoves-select*) are sufficiently stereotyped to allow training of an automated neural network EternaBrain move predictor with accuracy well above random baseline. We then challenge the resulting predictor to go beyond simply predicting individual moves and to instead perform a series of moves to solve novel RNA

secondary structure design problems from scratch (as evaluated by *in silico* folding models). We find generally modest performance from this CNN-only approach, with results poorer than previously published algorithms. We then collate several of the participants' hand-crafted strategies and pipeline a single-action-playout (SAP) of these strategies to the EternaBrain neural network approach. We show that the resulting EternaBrain-SAP algorithm (Fig 1E) achieves excellent performance on the Eterna100 benchmark by completing 61 of 100 puzzles, exceeding the performance of methods published prior to our work. Furthermore, EternaBrain-SAP performs similarly to newer methods developed concomitantly by us and other groups using complementary algorithmic approaches. In the discussion, we compare EternaBrain-SAP's performance on the Eterna100 with these more recently-reported methods, including SIMARD [17], SentRNA [18], NEMO [19], antaRNA [20], MCTS-RNA [21], and the reinforcement learning methods of Eastman et al. [15] and LEARNA [16], drawing lessons for future efforts in automated RNA design. Additionally, we highlight the likelihood of future progress as other methods [4][22] and newer computational energy functions [23] are developed and tested on the same benchmark as well as on biological RNA structures.

Results

Initial training on 1.8 million moves

We tested several different neural network architectures and training sets for EternaBrain. We chose to use convolutional neural network (CNN) architectures, because of their success in other areas of game playing and machine learning [13][14] and also because of their expected ability to capture patterns visually recognized by humans during actual Eterna gameplay. A CNN [24] is a specific type of neural network that hierarchically builds up complex features from simpler features that neighbor each other. CNNs consist of convolutional and pooling layers which retrieve subsections of the input features and make their own larger-scale features. A CNN is a natural choice for tackling this problem because it mimics how human participants approach Eterna puzzles; that is, by learning features of the data from spatial information such as those seen in the puzzle interface and in solution browsers. Our work involves supervised learning: the CNN's features and relationships were randomly initialized and then iteratively trained on large input datasets (training sets) and assessed for their predictive power when applied to previously unseen datasets (test sets).

We first trained the model on a large data set of 1.8 million moves spanning 12 representative puzzles from the Eterna progression (S1 Fig), which we call the *eternamoves-large* data set. The information in player move set data includes nucleotide sequence, RNA secondary structure (in 'dot-bracket' notation widely used in the RNA literature [25]), and predicted minimum Gibbs free energy of folding. These data on the game state are passed as input features to the CNN (Table 1). The location of the player's chosen nucleotide change and the nucleotide identity itself were passed as labels to the CNN–they are the output to be predicted (Table 1). We chose to train one 'base predictor' CNN (BP-CNN) to predict the identity of the base change (e.g. A, U, C, or G) made by a player provided the location of the change; as well as a separate 'location predictor' CNN (LP-CNN) to predict the location of the change. For the BP-CNN, we did not require that the neural network change the base nor that it maintain Watson-Crick pairing at nucleotides that were supposed to be paired in the target structure. Doing so provided a means to assess if the neural network could independently derive these fundamental move requirements. For cross-validation, we randomly split the data into training and test sets (see Methods). Training took three days using 4 NVidia Titan X GPUs.

Random guessing of moves would give accuracies of 0.33 for the BP-CNN (if the base is forced to change from its starting identity to one of the three other bases) and 0.019 for the

Table 1. Input features used for training and testing EternaBrain convolutional neural network.

Description	Example	Encoded example
Base Sequence	CCAGAAAAAAAAACUGG	[[0, 0, 0, 1], [0, 0, 0, 1], [1, 0, 0, 0], [0, 0, 1, 0], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0], [0, 0, 0, 1], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 1, 0]]
Predicted ('Nature Mode') Structure in dot-bracket notation	((((.........)))) (dot-bracket)	2,2,2,2,1,1,1,1,1,1,1,1,1,3,3,3,3,3
Target Structure in dot-bracket notation	(((((.........))))) (dot-bracket)	2,2,2,2,2,2,1,1,1,1,1,1,1,1,3,3,3,3,3
Predicted Structure Energy	-2.2 kcal/mol	-2.2
Target Energy	4.9 kcal/mol	4.9
Predicted Structure in pairmap notation*	((((.........)))) (pairmap)	16,15,14,13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,3,2,1,0
Target Structure in pairmap notation*	(((((.........))))) (pairmap)	16,15,14,13,12,-1,-1,-1,-1,-1,-1,-1,-1,4,3,2,1,0
Locked bases	xooooooooooooooox	2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2

* Additional features used for training on *eternamoves-select* and not *eternamoves-large* data set.

https://doi.org/10.1371/journal.pcbi.1007059.t001

LP-CNN (based on the average inverse length of the puzzles). Our best training attempts achieved move accuracies for the BP-CNN and LP-CNN of 0.50 and 0.10, respectively on the training dataset- both higher than random guessing. However, when applied to the test dataset, the BP-CNN and LP-CNN achieved move accuracies of 0.34 and 0.021, respectively, which are not significantly better than random guessing. Furthermore, we discovered poor results in subsequent tests of this model applied to full puzzle playouts. In these playouts, we had the CNN predict both the identity and location of the move starting from its initial sequence. Then, we updated the puzzle state according to ViennaRNA energy calculations and repeated this process. The model was unable to solve the easiest puzzle on the Eterna100—*Simple Hairpin*—even with modifications to allow stochastic choice of moves after ranking by the CNN (see Methods).

The low test accuracies and problems in an actual puzzle playout suggested that the CNNs might be overfitting their parameters to the training data and not generalizing well to Eterna puzzles that were outside the dataset. To counteract this overfitting, we had the networks randomly exclude randomly chosen nodes in each of their layers (dropout) and thus reduce the chance of overfitting. However, increasing this dropout rate gave only marginal improvements in test accuracy, suggesting that there was high variance in the training data [26]. This could result if, in these 12 early Eterna puzzles, players were using a highly heterogeneous set of strategies or clicking randomly without a clear strategy. Other parameter changes to the network (e.g.. use of a single CNN to predict base and location; change in the type of neural network; number of layers; activation functions; see S1 Table) also did not significantly improve test accuracy. These results suggested that moves made by players across this entire set are not sufficiently similar to allow their automation, at least with the convolutional neural network architectures tested.

Training on selected subsets of players and puzzles

We hypothesized that training on a select number of advanced Eterna participants would minimize the variance in the dataset. To select these 'expert' Eterna participants, we identified players who had solved at least 3000 Eterna puzzles. The resulting dataset contained moves from 72 players (Fig 1D; S2 Table). In addition, instead of using the 12 introductory simple puzzles used for *eternamoves-large* data set described above, we chose to collate moves from 78

puzzles compiled during a revision of the game's main puzzle progression (S2 Fig). These puzzles were selected because they introduced key strategies used by Eterna players to solve many of the RNA design challenges on Eterna. The resulting dataset, dubbed *eternamoves-select*, comprised 30,447 moves. In addition to narrowing the training data to expert players and specialized puzzles, we added a more explicit representation of RNA secondary structure by introducing pairmaps. Rather than merely providing dot-bracket notation to the CNN, a structure pairmap uses indices of a list to explicitly show which bases are paired to other bases. The format of the pairmap is shown in Table 1.

Combining *eternamoves-select* and introducing pairmaps increased the predictive power of our CNNs. The final move accuracies for cross-validation were 0.51 for the BP-CNN and 0.34 for the LP-CNN (S3A and S3B Fig; Table 2) when applied to the test dataset of *eternamoves-select*. These move accuracies were higher than the expected random baselines for these puzzles of 0.33 and 0.031, respectively. We expected that the model's predictive powers would negatively correlate with the puzzle length, and this trend was indeed apparent. For example, the LP-CNN accuracy dropped from 0.43 to 0.28 when going from puzzles with lengths of up to 50 nucleotides (nts) to puzzles with lengths from 101 to 150 nts; Table 3). The BP-CNN accuracy reduced from 0.57 to 0.47 with the same increase in puzzle length (Table 3). We also expected that the model would have higher prediction accuracy at nucleotides that involved paired regions of the target structure. In those regions, the model can correct 'mismatches'–nucleotides that should be paired but are not A-U, G-C, or G-U–and such moves would provide obvious location and base choices. Surprisingly, our model exhibited slightly better performance in both base and location prediction in regions of the puzzle that were unpaired rather than paired in the target structure (0.62 vs. 0.49, base; 0.38 vs. 0.33, location; Table 3).

The improved CNN accuracies using *eternamoves-select* compared to *eternamoves-large* suggested that moves from experienced players and on specially designed problems were sufficiently stereotyped so as to allow for CNN prediction. To further explore whether these improvements could be generalized across different puzzles or between different expert players, we split *eternamoves-select* into training and test sets in different ways (see Table 2). Rather than allowing moves in the training and test sets to be drawn from the same puzzles (as above), we trained our CNNs on 15,265 moves from 39 of the 78 puzzles in our training set. The predictive accuracies on the 15,182 moves from the remaining 39 puzzles were 0.26 and 0.023 (BP-CNN and LP-CNN, respectively). These numbers were lower than for our initial CNN tests on *eternamoves-select*, in which the training and test set drew moves from the same puzzles. Indeed, the BP-CNN accuracy dropped below the random baseline value of 0.33 when the puzzles in the training and test set were separated. These observations indicate that the CNN predictions depend on the overlap between puzzles seen in training and testing. We next

Table 2. EternaBrain CNN accuracies on *eternamoves-select* with different splits of training and test sets.

Model	Training Accuracy	Test Accuracy
EternaBrain—base	0.71	0.51
EternaBrain—location	0.31	0.34
Half experts—base	0.66	0.38
Half experts—location	0.30	0.11
Half puzzles—base	0.70	0.27
Half puzzles—location	0.33	0.02
One expert—base	0.79	0.25
One expert—location	0.5	0.01

https://doi.org/10.1371/journal.pcbi.1007059.t002

Table 3. EternaBrain CNN accuracies on *eternamoves-select*, grouped by length of puzzle and paired/unpaired status of nucleotide at which move was applied.

Length of Puzzles	Number of Moves	Location Accuracy	Location Accuracy (%)	Base Accuracy	Base Accuracy (%)
1–50	1034	398	38%	537	52%
51–100	771	232	30%	385	50%
101–150	1329	346	26%	624	47%
151–400	313	32	10%	141	42%
All	3447	1008	29%	1687	49%
Paired					
1–50	687	257	37%	337	49%
51–100	674	201	29%	324	48%
101–150	1206	320	27%	557	46%
151–400	264	23	8%	114	43%
All	2831	801	28%	1332	47%
Unpaired					
1–50	348	141	40%	200	57%
51–100	96	30	31%	61	63%
101–150	122	26	21%	67	55%
151–400	50	9	18%	27	54%
All	616	206	33%	355	58%

https://doi.org/10.1371/journal.pcbi.1007059.t003

split the *eternamoves-select* dataset so that the moves in the training set derived from half of the expert players. The test accuracies of our models applied to the dataset of moves from the remaining half of expert players were 0.38 and 0.11. These accuracies remained above random baselines (0.33 and 0.031, respectively) and suggested that, while expert players vary in their puzzle solving styles, there are some commonalities that can be learned by the CNNs. Last, we trained our models on just 587 moves of a single expert player and tested its performance on the remaining 29,877 moves in the *eternamoves-select* dataset. The resulting test accuracies were 0.27 (BP-CNN) and 0.011 (LP-CNN)—significantly worse than above. However, this poor result may be due to the dramatic decrease in training data for the CNN compared to prior comparisons. Moving forward, we decided to utilize the CNN trained on our original random split of *eternamoves-select* across all puzzles and expert players, but taking note that its move prediction accuracy would likely depend on similarity of any new challenge puzzles to the puzzles in the training set. We named this predictor the EternaBrain CNN.

Playouts on the Eterna100 benchmark

After the cross-validation tests above, we set out to solve complete puzzles with the EternaBrain CNN model trained on *eternamoves-select* and the stochastic playout scheme described above. Encouragingly, the model was successfully able to solve several puzzles completely, including the *Simple Hairpin* puzzle, which the CNN trained on *eternamoves-large* failed to do, as described earlier. However, EternaBrain CNN failed to solve puzzles greater than 40 nucleotides in length. Visual inspection of the CNN's series of moves revealed minor but obvious mistakes, such as mismatched nucleotides at pairs of positions that should have been Watson-Crick paired in the target structure. In order to prevent some of these mistakes, we followed the CNN with a second stage that we called single action playout (SAP; Fig 1E). SAP uses six canonical strategies that are standard among Eterna players and are taught to new players during the game's main puzzle progression (Fig 2). SAP traversed the puzzle to find areas that were not folding correctly, implemented the relevant strategies, and accepted the sequences if they made these specific areas of the puzzle fold correctly; Fig 2 and Methods give more

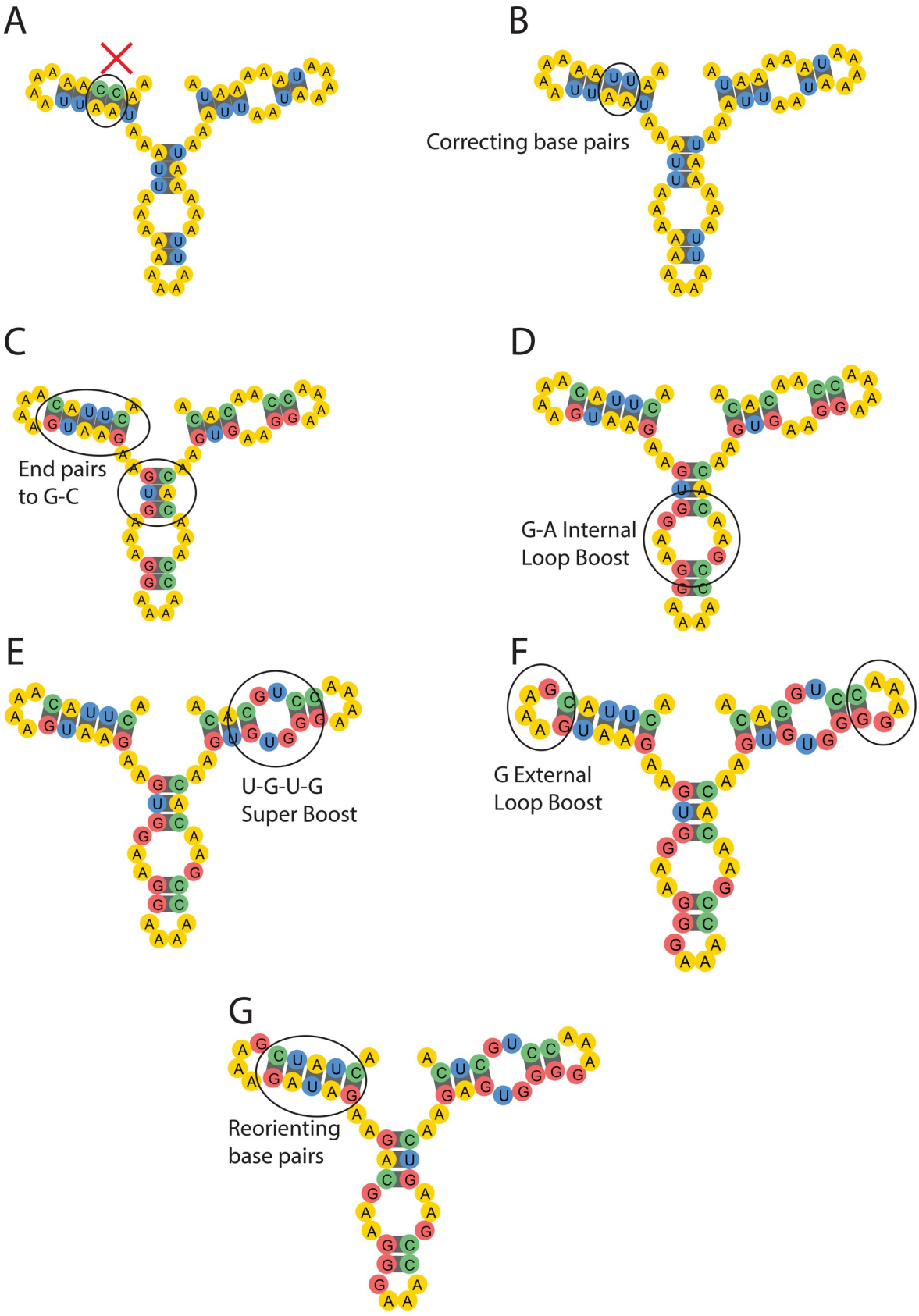

A
B
Correcting base pairs
C
End pairs
to G-C
D
G-A Internal
Loop Boost
E
U-G-U-G
Super Boost
F
G External
Loop Boost
G
Reorienting
base pairs

Fig 2. The 6 strategies included in the SAP. (A) The original state of the puzzle before SAP. This represents a puzzle initiated with an arbitrary sequence of nucleotides; panel displays the target structure, where mismatched nucleotides (C-A) are highlighted. (B) The first step of the SAP is to correct mismatched pairs. Here, the cytosine nucleotides are switched to uracil to pair with adenine. (C) Changing end pairs to G-C. Changing base pairs that are at the edges of stems and flank loops to G-C pairs lowers the free energy of the molecule. (D) G-internal loop boost. The first nucleotide in an internal loop on either side is switched to a guanine. (E) U-G-U-G super boost. In an internal loop with 2 unpaired bases on either side, the 2 bases are changed to uracil and guanine, in that order, on either side. (F) G-hairpin boost. The first nucleotide in each strand of a hairpin loop is changed to a guanine. (G) Reorienting base pairs. Target base pairs that are not predicted to be folded correctly are 'flipped' to lower the energy of the structure. Here, alternating the A-U pairs lowers the energy of the stack. The 5' end of each puzzle is at the top left, with the puzzle drawn counter-clockwise from that point.

https://doi.org/10.1371/journal.pcbi.1007059.g002

detailed descriptions of the SAP's standard Watson-Crick pairing rule, the G-C end pair rule, the G-internal loop and hairpin-loop boosts, the U-G-U-G super boost, and the flipping pairs strategy.

We called this enhanced pipeline the EternaBrain-SAP method and tested it against the Eterna100 benchmark. These 100 puzzles span a wide range of different structural motifs but are known to be solvable in the Turner1999 energy model, which is widely used in RNA structure prediction software packages including the original RNAinverse and NUPACK [6][9]. EternaBrain-SAP was able to solve 61 of the 100 puzzles, surpassing six of the algorithms previously benchmarked with the Eterna100 puzzles (Fig 3A). We note, however, that newer design methods using complementary algorithmic strategies provide similar or better performance compare to EternaBrain-SAP, and these will be discussed below.

Importantly, both the CNN and the SAP stages were necessary for achieving this performance. Using EternaBrain's CNN-based moves alone solves only 20 puzzles on the Eterna100. Using the SAP alone (i.e., hard-coded canonical player strategies) solves 50 puzzles on the Eterna100, 11 short of the 61 puzzles solved by the CNN-SAP combination. However, it is important to note that the puzzles become significantly more difficult, as evidenced by a steep decrease in the number of players and prior algorithms who have solved them with increasing number (Fig 3A and [10]); and so this represents a significant improvement over SAP alone. We confirmed that several choices that we made for the CNN architecture and game state representations (Table 1) were important for the success of EternaBrain-SAP. Removing the sequence, the pairmap, or the dot-bracket representation as input features or not using dropout also decreased performance in Eterna100 playouts (from 61 to 51, 52, 56, and 57 puzzles solved, respectively). A puzzle-by-puzzle breakdown of the performance of the CNN alone, the SAP alone, and CNN-SAP with these reduced-input-feature training sets is given in Fig 3B. We also note that while EternaBrain's CNN-based moves have a stochastic component (see Methods), its success on puzzles is consistent across runs using different random seeds (Table 4).

Performance on specific features of difficult puzzles

The Eterna100 benchmark puzzles showcase structural features that are difficult for RNA inverse folding algorithms to design. Comparison of EternaBrain-SAP's performance across different puzzle types clarifies its current abilities and limitations.

Simple motifs—Stacks, loops, hairpins

The EternaBrain-SAP algorithm was particularly successful at solving puzzles that contained several stacks (i.e., RNA stems), loops (i.e., internal loops or two-way junctions), and hairpins (i.e., external loops). One example of a difficult puzzle that EternaBrain's CNN was able to solve is *U*. This puzzle contains several short stacks and short loops abutted next to each other (Fig 4A), and was not solvable by the SAP alone or by five of the six prior design algorithms


A
Simple_hairpin
Arabidopsis_Thaliana_6_RNA_-_Difficulty_Level_0
Prion_Pseudoknot_-_Difficulty_Level_0
Human_integrated_adenovirus_-_Level_0
The_Gammaretrovirus_Signal_-_Diffuculty_Level_0
Saccharomyces_Cerevisiae_-_Difficulty_Level_0
The_fractal
G-C_Placement
The_Sun
Frog_Foot
InfoRNA_test_16
Mat_–_Martian_2
square
Six_legd_turtle_2
Small_and_Easy_6
Fractile
Six_legd_Turtle
snoRNA_SNORD64
Chalk_Outline
InfoRNA_bulge_test_9
Tilted_Russian_Cross
ACTUALLY_Small_And_Easy_6
Shortie_4
Shape_Test
The_Minitsry
stickshift_
U
Still_Life_(Sunflower_In_A_Vase)
Quasispecies_2-2_Loop_Challenge
Corner_bulge_training_
Spiral
InfoRNA_bulge_test
Worm_1
just_down_to_1_bulge
Iron_Cross
loops_and_stems
Water_Strider
The_turtle(s)_move(s)
Adenine
Tripod_5
Shortie_6
Runner_
Recoil
[CloudBeta]_An_Arm_and_a_Leg_1.0_
[CloudBeta]_5_Adjacent_Stack_Multi-Branch_Loop
Triple_Y
Misfolded_Aptamer
Flower_power
Kudzu
1,2,3and4bulges
medallion
[RNA]_Repetitious_Sequences_8_10
Documenting_repetitious_behavior
7_multiloop
Kyurem_7
JF1
multilooping_fun
Multiloop...
hard_Y
Mat_-_Elements_&_Sections
Chicken_feet
Bug_18
Fractal_star_x5
Crop_Circle_2
Branching_Loop
Bug_38
Simple_Single_Bond
Taraxacumr_officinale
Headless_Bug_on_Windshield
Pokeball
Variation_of_a_crop_circle
Loop_next_to_a_Multiloop
Snowflake_4
Mat_-_Cuboid
Misfolded_Aptamer_6
Snowflake_3
Hard_Y_and_a_bit_more
Mat_-_Lot_2-2_B
Shapes_and_Energy
Spiral_of_5s
Campfire
Anemone
Fractal_3
Kyurem_5
Snowflake_Necklace_(_or_v2.0_)
Methaqualone_C16H14N2O_Structural_Representation
Cats_Toy_2
Zigzag_Semicircle
Short_string_4
Gladius
Thunderbolt
Mutated_chicken_feet
Chicken_Tracks
Looking_Back_Again
Multilooping_6
Cesspool
Hoglafractal
Bullseye
Shooting_Star
Teslagon
RNASSD...27
RNAinverse...28
DSSOPT...47
NUPACK...48
INFORNA...50
MODENA...54
EternaBrain_SAP...61
B
Simple_hairpin
Arabidopsis_Thaliana_6_RNA_-_Difficulty_Level_0
Prion_Pseudoknot_-_Difficulty_Level_0
Human_integrated_adenovirus_-_Level_0
The_Gammaretrovirus_Signal_-_Diffuculty_Level_0
Saccharomyces_Cerevisiae_-_Difficulty_Level_0
The_fractal
G-C_Placement
The_Sun
Frog_Foot
InfoRNA_test_16
Mat_–_Martian_2
square
Six_legd_turtle_2
Small_and_Easy_6
Fractile
snoRNA_SNORD64
Chalk_Outline
InfoRNA_bulge_test_9
Tilted_Russian_Cross
This_is_ACTUALLY_Small_And_Easy_6
Shortie_4
Shape_Test
The_Minitsry
stickshift_
U
Still_Life_(Sunflower_In_A_Vase)
Quasispecies_2-2_Loop_Challenge
Corner_bulge_training_
Spiral
InfoRNA_bulge_test
Worm_1
just_down_to_1_bulge
Iron_Cross
loops_and_stems
Water_Strider
The_turtle(s)_move(s)
Adenine
Tripod_5
Shortie_6
Runner_
Recoil
[CloudBeta]_An_Arm_and_a_Leg_1.0_
[CloudBeta]_5_Adjacent_Stack_Multi-Branch_Loop
Triple_Y
Misfolded_Aptamer
Flower_power
Kudzu
1,2,3and4bulges
medallion
[RNA]_Repetitious_Sequences_8_10
Documenting_repetitious_behavior
7_multiloop
Kyurem_7
JF1
multilooping_fun
Multiloop...
hard_Y
Mat_-_Elements_&_Sections
Chicken_feet
Bug_18
Fractal_star_x5
Crop_Circle_2
Branching_Loop
Bug_38
Simple_Single_Bond
Taraxacumr_officinale
Headless_Bug_on_Windshield
Pokeball
Variation_of_a_crop_circle
Loop_next_to_a_Multiloop
Snowflake_4
Mat_-_Cuboid
Misfolded_Aptamer_6
Snowflake_3
Hard_Y_and_a_bit_more
Mat_-_Lot_2-2_B
Shapes_and_Energy
Spiral_of_5s
Campfire
Anemone
Fractal_3
Kyurem_5
Snowflake_Necklace_(_or_v2.0_)
Methaqualone_C16H14N2O_Structural_Representation
Cats_Toy_2
Zigzag_Semicircle
Short_string_4
Gladius
Thunderbolt
Mutated_chicken_feet
Chicken_Tracks
Looking_Back_Again
Multilooping_6
Cesspool
Hoglafractal
Bullseye
Shooting_Star
Teslagon
CNN_alone...20
SAP_alone...50
No_energy...51
No_sequence...51
No_pairmap...52
No_dropout...57
No_dot_bracket...58
EternaBrain_SAP...61

Fig 3. EternaBrain performance. (A) Performance of EternaBrain and 6 previously published algorithms on Eterna100 benchmark. EternaBrain solves 61/100, followed by MODENA (54/100), INFO-RNA (50/100), NUPACK (48/100), DSS-Opt (47/100), RNAinverse (28/100), and RNA-SSD (27/100). (B) Performance of Alternative Model Constructions. The CNN alone could solve only 20/100, and the SAP alone could solve 50/100. Removing various input features passed into the CNN resulted in drops in performance, confirming the importance of these features.

https://doi.org/10.1371/journal.pcbi.1007059.g003

(Fig 3A). This example suggests that EternaBrain had successfully learned from its training data how to strengthen stacks and to stabilize loops. Specifically, EternaBrain's CNN was able to learn strategies like the G-C end pair strategy (which strengthens a base pair stack) and the G-hairpin boost (which stabilizes an internal loop); see Fig 2C and 2F. While these strategies are well known and, indeed, are encoded in the SAP (Fig 2), the EternaBrain CNN learned further patterns that were not encoded in the SAP, as is demonstrated by the inability of SAP alone to solve the puzzle *U* (Fig 3B).

Specific orientation of base pairs

If the puzzle contained motifs whose solutions critically depended on unique sequences, the CNN was inadequate and SAP was important for EternaBrain-SAP's success. An example of such a puzzle was *Chicken Tracks* which required a very specific orientation of base pairs (Fig 4B). As a result, the SAP was used heavily in solving *Chicken Tracks*. By locating the areas that were not folding correctly and then using the canonical player strategy of reorienting base pairs, the SAP was able to find the optimal orientation of the base pairs to correctly solve *Chicken Tracks*.

Repetitive structures

Some of EternaBrain-SAP's successes can be attributed to its ability to solve repetitive structures better than previous automated RNA design algorithms. Example puzzles that include many repetitive structures include *Thunderbolt*, *Shortie 4*, and *Shortie 6* (Fig 4C–4E). Previous algorithms were able to solve puzzles with several stacks, such as *Shortie 4* (Fig 4D). However, when the number of stacks increased, e.g., in *Shortie 6* (Fig 4E), other algorithms struggled. EternaBrain-SAP, however, could solve both *Shortie 4* and *Shortie 6*. *Thunderbolt* involves repeating elements within a large structure (Fig 4C) and also was not solvable by prior algorithms (Fig 3A). Given that SAP was able to solve each of these puzzles alone (Fig 3B), the moveset-trained CNN of EternaBrain appears to offer little benefit over SAP alone in stabilizing repetitive structures.

EternaBrain-SAP's failure to solve other puzzles in the benchmark seems, in part, to be a result of the incompleteness of the training data for the EternaBrain CNN and of the player strategies in the SAP algorithm. For example, *Hard Y* (Fig 4F), requires uncommon strategies, including a different type of boost (a stabilizer mutation at the beginning of a loop) to stabilize a special 'zigzag' structural motif which did not appear in the training set. The SAP was unable to solve this puzzle since reorienting bases without the required boost did not stabilize this zigzag. Training the CNN on larger movesets and incorporating more sophisticated player strategies in SAP might resolve these issues and help EternaBrain-SAP complete more complex puzzles.

Discussion

EternaBrain-SAP attempts to solve the RNA secondary structure design problem by learning from a large compilation of human player moves and incorporating a special set of canonical player strategies. We decided to use a convolutional neural network (CNN) since it can be

Table 4. EternaBrain-SAP performance on Eterna100 upon five additional playouts on the 61 puzzles it solved in its first run.

Puzzle	Number of Times Solved out of 5	Puzzle	Number of Times Solved out of 5
Simple_hairpin	5	medallion	4
Arabidopsis_Thaliana_6_RNA_-_Difficulty_Level_0	5	[RNA]_Repetitious_Sequences_8_10	5
Prion_Pseudoknot_-_Difficulty_Level_0	5	Documenting_repetitious_behavior	5
Human_integrated_adenovirus_-_Level_0	4	7_multiloop	5
The_Gammaretrovirus_Signal_-_Diffuculty_Level_0	5	Kyurem_7	0
Saccharomyces_Cerevisiae_-_Difficulty_Level_0	5	JF1	5
The_fractal	5	multilooping_fun	0
G-C_Placement	5	Multiloop . . .	3
The_Sun	5	hard_Y	0
Frog_Foot	5	Mat_-_Elements_&_Sections	0
InfoRNA_test_16	5	Chicken_feet	0
Mat_–_Martian_2	5	Bug_18	0
square	5	Fractal_star_x5	5
Six_legd_turtle_2	5	Crop_Circle_2	0
Small_and_Easy_6	0	Branching_Loop	0
Fractile	5	Bug_38	0
Six_legd_turtle_2	5	Simple_Single_Bond	0
snoRNA_SNORD64	5	Taraxacumr_officinale	5
Chalk_Outline	4	Headless_Bug_on_Windshield	0
InfoRNA_bulge_test_9	4	Pokeball	1
Tilted_Russian_Cross	5	Variation_of_a_crop_circle	3
This_is_ACTUALLY_Small_And_Easy_6	5	Loop_next_to_a_Multiloop	0
Shortie_4	5	Snowflake_4	0
Shape_Test	3	Mat_-_Cuboid	3
The_Minitsry	5	Misfolded_Aptamer_6	2
stickshift_	5	Snowflake_3	0
U	4	Hard_Y_and_a_bit_more	0
Still_Life_(Sunflower_In_A_Vase)	5	Mat_-_Lot_2–2_B	0
Quasispecies_2–2_Loop_Challenge	4	Shapes_and_Energy	0
Corner_bulge_training_	5	Spiral_of_5s	0
Spiral	5	Campfire	0
InfoRNA_bulge_test	5	Anemone	3
Worm_1	4	Fractal_3	5
just_down_to_1_bulge	3	Kyurem_5	0
Iron_Cross	5	Snowflake_Necklace_(_or_v2.0_)	0
loops_and_stems	5	Methaqualone_C16H14N2O_Structural_Representation	0
Water_Strider	5	Cats_Toy_2	0
The_turtle(s)_move(s)	5	Zigzag_Semicircle	0
Adenine	5	Short_string_4	0
Tripod_5	4	Gladius	0
Shortie_6	5	Thunderbolt	4
Runner_	5	Mutated_chicken_feet	0
Recoil	5	Chicken_Tracks	2
[CloudBeta]_An_Arm_and_a_Leg_1.0_	5	Looking_Back_Again	0

(*Continued*)

Table 4. (Continued)

Puzzle	Number of Times Solved out of 5	Puzzle	Number of Times Solved out of 5
[CloudBeta]_5_Adjacent_Stack_Multi-Branch_Loop	5	Multilooping_6	0
Triple_Y	0	Cesspool	0
Misfolded_Aptamer	4	Hoglafractal	0
Flower_power	0	Bullseye	0
Kudzu	0	Shooting_Star	0
1,2,3and4bulges	0	Teslagon	0

https://doi.org/10.1371/journal.pcbi.1007059.t004

trained to extract information from the nearest neighbors of elements in the feature space, mimicking how Eterna players look at the local neighborhoods of structures and nucleotides to decide their next move. To reduce variance in the training set, we found it important to use movesets only from expert players. By reducing the full *eternamoves-large* moveset (1.8 million moves) to the *eternamoves-select* moveset (~30,000 moves), we were able to achieve test accuracies after CNN training that were better than random guesses. Notably, for the location predictor (LP-CNN), the test accuracy of 0.34 substantially exceeded the baseline prediction accuracy for random guessing (0.019).

Despite decreased variance from training on the *eterna-select* moveset, we found that the resulting EternaBrain CNN had difficulty with solving longer puzzles. Indeed, alternative splitting of test and training sets suggested that the CNNs' prediction accuracy largely depends on similarity of puzzles in the training set with the puzzles of any new challenges. The inability to extrapolate "out of sample" has been noted to be a limitation of artificial neural networks [27]. To overcome this limitation, we added a single-action playout (SAP) algorithm based on compiled Eterna player strategies to aid the CNN model. We tested this hybrid EternaBrain-SAP algorithm on the Eterna100, and found that it could solve 61 of 100 puzzles. On one hand, this performance is better than other algorithms that had been tested on the Eterna100 at the time of development (54 of 100 was the previous maximum). On the other hand, the EternaBrain-SAP performance is similar to or worse than newer algorithms (SIMARD, sentRNA, the reinforcement learning algorithm of Eastman et al., NEMO) that have been developed concomitantly or after EternaBrain and whose performance on the Eterna100 has been reported in newer papers or preprints [15][16][17][18][19]. Furthermore, none of these methods match the level of the top ten Eterna human players, who can solve all 100 puzzles of the Eterna100 benchmark.

It is instructive to compare EternaBrain-SAP to the newer generation of RNA design algorithms. Like EternaBrain-SAP, the new methods SentRNA, the Eastman et al. method, and LEARNA use artificial neural networks to distill potentially useful information from gameplay and solve 80, 60, and 65 out of 100 Eterna100 puzzles, respectively [15][16][18]. SentRNA seeks to find solutions to RNA secondary design problems in 'one shot' rather than through EternaBrain's iterative moves [18]. Furthermore, SentRNA differs from EternaBrain as it is trained on Eterna player's solutions (the *eternasolves* data set) rather than the individual moves that lead to solutions, and it makes use of a three-layer fully-connected neural network rather than EternaBrain's deep convolutional neural network. Despite these differences, both the SentRNA and our EternaBrain-SAP study find that neural network approaches alone give poor performance in test puzzles (in both cases solving fewer than half of the Eterna100 puzzles). The success of both studies required pipelining starting solutions from neural network approaches with hand-coded strategies that Eterna players collectively learned and 'canonized'

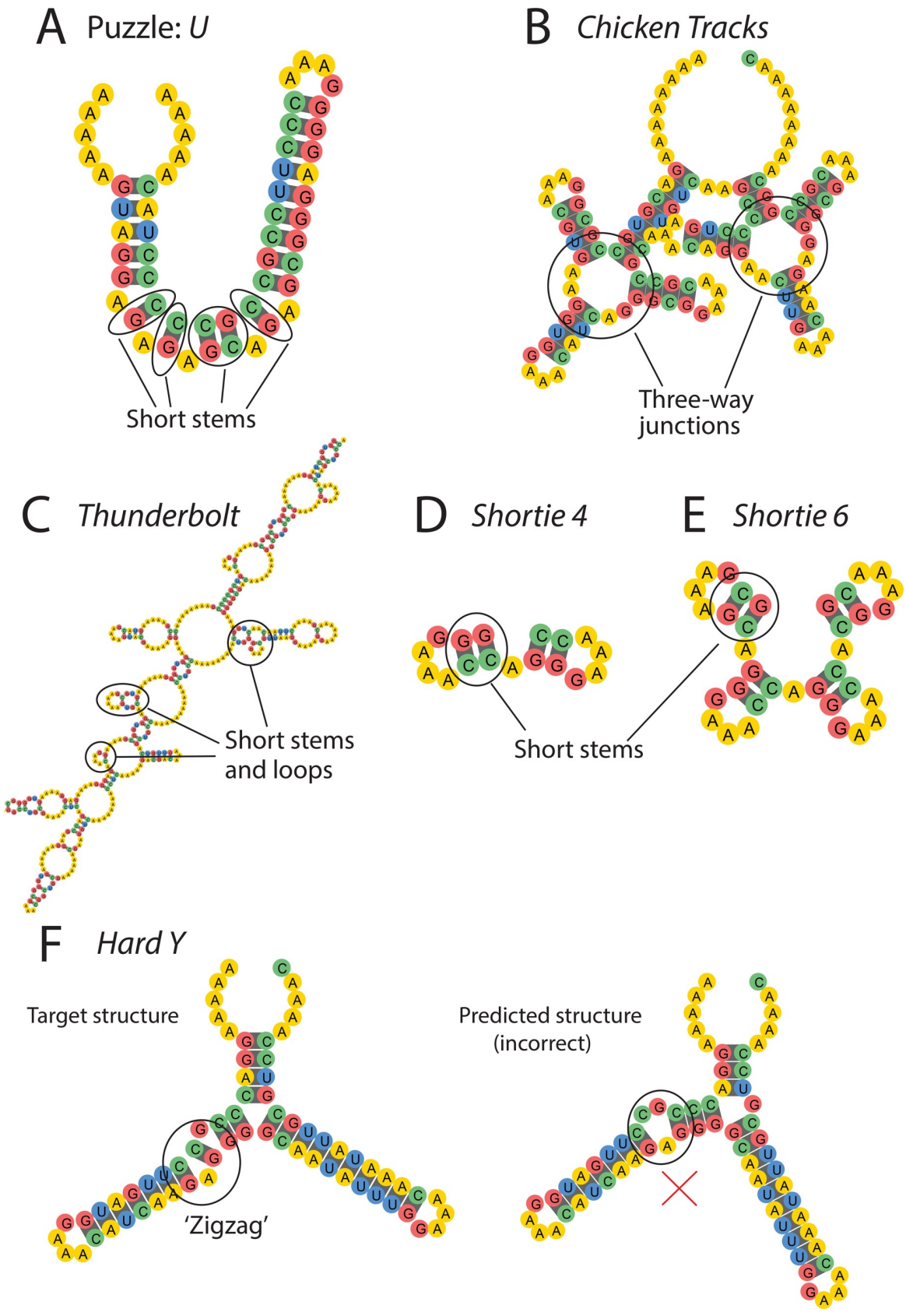

A Puzzle: U
Short stems
B Chicken Tracks
Three-way junctions
C Thunderbolt
Short stems and loops
D Shortie 4
Short stems
E Shortie 6
F Hard Y
Target structure
'Zigzag'
Predicted structure (incorrect)

Fig 4. Example EternaBrain-SAP solutions to Eterna100 puzzles. (A) *U* solution highlights the fact that the EternaBrain CNN alone can solve puzzles with short stems. (B) *Chicken Tracks* solution: EternaBrain-SAP can solve puzzles with three stems intersecting in one internal loop. (C) *Thunderbolt* solution demonstrates that EternaBrain-SAP can solve large puzzles (400 nucleotides long) and solve loops and stems in combination. (D) *Shortie 4* solution shows EternaBrain-SAP can solve puzzles with multiple short stems (2 nucleotides long). (E) *Shortie 6* is quite similar to *Shortie 4*, but with the same motif (short stems) repeated. The other algorithms mentioned could not solve *Shortie 6* because of the repeated motifs. (F) *Hard Y*—target structure (left) vs nature-mode (right) structure. EternaBrain-SAP could not solve *Hard Y* because it required use of a little-used strategy to solve a motif called a zigzag. Since the strategy is not often used by players, the EternaBrain CNN did not learn the strategy and the strategy was not included in the SAP. In each panel, the 5' end of each puzzle is at the top left, with the puzzle drawn counter-clockwise from that point.

https://doi.org/10.1371/journal.pcbi.1007059.g004

in tutorial puzzles for new players. The Eastman et al. and LEARNA algorithms [15][16] are reinforcement learning methods which have not leveraged prior human solutions or strategies at all; nevertheless, both use additional hand-coded rules to enforce Watson-Crick pairing of nucleotides during design and, in the case of LEARNA, an additional refinement step. Furthermore, at least the Eastman et al. method does not learn 'standard' human strategies like the G-boosts (Fig 2), which might explain weaker performance compared to EternaBrain-SAP and SentRNA.

Given these initial results from neural network methods, we propose future updates that may allow automated design to reach the level of expert players and to allow for a general strategy that will work on more complex RNA design problems including multi-state switch design and 3D structure design. First, EternaBrain-SAP and SentRNA achieve success on different problems in the Eterna100. This observation suggests that the two methods could be integrated, with each one providing starter solutions for the other, or the convolutional neural network architecture from EternaBrain used as an alternative neural network for SentRNA. Second, the hand-coded player strategies used in both EternaBrain and SentRNA often involve multiple moves. These strategies could possibly be captured by the neural networks underlying all four available deep learning approaches if they are trained to make moves based not just on its current game state but also including immediately previous moves as input. For example, such training would allow EternaBrain-SAP to ensure Watson-Crick compatibility across nucleotides that are paired in the target structure, an 'obvious' feature that the CNN is not always recognizing and that had to be hard-coded into the Eastman et al. method and LEARNA. Finally, it is important to point out that more advanced deep learning architectures may outperform these approaches and achieve human-level performance. Recent results in protein structure prediction (13th Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction, http://predictioncenter.org/casp13/index.cgi) demonstrate how rapidly such gains are happening. These methods have benefitted strongly from large data sets drawn from the large public archives of protein sequence and structural data; for the RNA design problem, the *eternamoves* and *eternasolves* data sets provide potentially analogous data.

Complementary gains in automated RNA design methods may come not from fine-tuning the neural network architectures or training sets, but through adoption or integration of more completely distinct strategies. Methods like SIMARD [17][28], antaRNA [20], and MCTS-RNA [21] achieve performances similar to or slightly better than EternaBrain (54 to 67 out of 100) while using quite different but complementary simulated annealing, ant colony optimization, and Monte Carlo tree search strategies, respectively. Perhaps most strikingly, one of us (FP) has recently reported that a nested Monte Carlo (NEMO) method can solve 98/100 puzzles in the Eterna100 benchmark, only missing the last two problems, called *Shooting Star* and *Teslagon* [19]. While NEMO fills in candidate solutions from 5′ to 3′, like SentRNA, other aspects of the method are completely distinct, and NEMO does not use a neural network approach. Second, we note that human Eterna players often solve complex RNA design puzzles

by manually preparing sub-puzzles and using detailed mathematical reasoning to infer solutions. Neither of these steps is recorded in the Eterna move sets or has, to our knowledge, been captured in design algorithms. Finally, other methods like RNAifold [4] and the recently presented RNAstructure *Design_preselected* routine [22] have yet to be tested on the Eterna100 benchmark; they may show promise on puzzles that are not solvable by other methods.

In addition to their prospects for achieving human-competitive performance on *in silico* single structure design, we speculate that CNN-based move prediction and deep learning frameworks will be useful for design of functional RNAs that work *in vitro* or *in vivo*. The Eterna project is currently soliciting designs for ligand-responsive multi-state riboswitches and for redesigning large biological RNAs like the ribosome for *in vitro* tests. Current computational design methods are not able to automatically provide solutions to these problems. When these computational methods are ready, datasets involving hundreds of thousands of RNA molecules are accumulating in the Eterna project [27] and should provide rich resources for training and prospective tests.

Methods

Ethics statement

This study was approved by the Stanford Institutional Review Board (Approval Number 34669), and falls under "Waiver of Consent" (participants notified through Eterna End User License Agreement).

Code availability

Code for training EternaBrain and solving puzzles with Eterna-SAP is freely available for non-commercial use at *https://github.com/EteRNAgame/EternaBrain*.

Data encoding

Through the Eterna puzzle-solving interface (Fig 1A–1C), players can mutate an RNA molecule's base sequence by selecting an RNA nucleotide base (A, U, G, or C) and the location on the puzzle where they would like to make the change. Players can see the 'target' structure–the secondary structure they are trying to achieve–and the 'nature-mode' (or 'natural') structure–the secondary structure predicted to be the minimum free energy conformation for the current sequence. When the nature-mode and target structures match, the player has solved the puzzle. In both target and nature-mode states, players can see the predicted free energy of the molecules (in kilocalories per mole). These values are routinely used by players to guide moves that make their RNA structure more stable. These energies and structures are calculated using Vienna 1.8.5 [6], which provides the default energy model in Eterna; additional tests with newer energy functions are possible in EternaBrain but are not reported here.

All information given to Eterna players was encoded before being passed into the CNN (see Table 1). Such information included the nucleotide sequence, nature-mode and target structure and pairmaps, nature-mode and target energy, and locked bases, as follows. The nucleotide bases were encoded using a standard 'one-hot' representation over four input layers; A, U, G, C were mapped to [1,0,0,0], [0,1,0,0], [0,0,1,0], and [0,0,0,1], respectively. Simultaneous mutations of bases were treated as separate changes, and any copying or resetting of bases sequences were encoded as [1,1,1,1] in the otherwise one-hot input for A, U, G, or C. For encoding the 'nature-mode' structure (minimum free energy structure predicted by Vienna 1.8.5) and target structure, dot-bracket notation was converted to one-hot representation, with

unpaired bases set to zero and left- and right-paired bases set to one in three separate input layers. Another representation of the nature-mode and target structures used is structure pairmaps. Each entry in a pairmap, corresponding to a particular location in the sequence, stores the index of the base with which it forms a base pair. For example, if the base at location 1 is paired to location 10, then index 0 in the pairmap would contain the number 9, and index 9 in the pairmap would contain the number 0 (using a list starting at index 0). For training, the player's chosen base and the location of base change were provided as labels and a standard soft-max [29] loss function computed agreement of the neural network predictions and the actual player moves. An example of the final data encoding is shown in Table 1.

Model construction and evaluation

Two CNNs were built: one for predicting the RNA base, and one for predicting the location of the base change. After running several experiments on different CNN architectures, the following architecture was used for each CNN: 10 convolutional layers (with convolution size 9 in the dimension that indexes across RNA sequence position, stride 2x2, and pooling; the number of neurons in each layer were 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024) followed by 4 fully-connected layers (1024, 1024, 2048, 4096), a dropout rate of 0.1, a sigmoid activation function [30], minibatch size of 100 solutions, and Adam optimizer [31] to minimize the error of the neural network. A modest hyperparameter search was carried out before settling on this architecture, as described in S1 Table. Construction and training of CNNs were carried out using Google's TensorFlow machine learning framework [32].

The models discussed in this work were trained on Nvidia Titan X GPUs available on Stanford's Sherlock cluster.

For puzzle playouts, the CNN initially attempted to solve the puzzle by iteratively choosing moves and updating the game state. The number of such moves was chosen to be the length of the puzzle multiplied by three. The move choice was stochastic, with probabilities of base and location based on the respective CNN output renormalized to 1.0. If this stage was not able to solve the puzzle, the SAP was applied, implementing the player strategies in Fig 2. The SAP changed specific nucleotides in the puzzle according to the player strategy and compared if the nature-mode structure of the RNA more closely matched the target structure than if the player strategy had not been implemented. If the nature-mode structure more closely matched the target structure, then the resulting nucleotide sequence was kept. (For speed, closeness of two secondary structures was defined based on the length of the largest matching subsequence of the two secondary structures written in dot-bracket notation, using the SequenceMatcher class in Python.) This process was repeated for all of the player strategies. The current implementation of SAP uses a few straightforward player strategies, favoring simplicity over computational complexity (see Fig 2).

During development of the model, we evaluated the CNN using cross-validation, training on 30,000 moves and testing on the remaining moves; test moves were pulled randomly from the data set unless noted otherwise [29]. Playout tests were carried out on the 100 secondary structures of the Eterna100 benchmark [10]. To ensure fair comparison to prior work [10], we did not include puzzle constraints on, e.g., minimum or maximum number of A-U pairs, which arise in some of the Eterna100 puzzles when they are played by humans online.

Descriptions of each SAP strategy

Single-action playout (SAP) consists of the application of six strategies:

1. The first step of the SAP is to correct incorrect base pairings. The algorithm traverses the entire length of the sequence, identifying any mismatches in base pairings. Any incorrect pairings will be mutated to A-U or G-C, depending on one of the bases in the mismatch.
2. Second, the algorithm will identify any base pairs flanking a hairpin loop or an internal loop and change that base pair to G-C, since G-C pairs at the end of stacks lower the energy of that subregion.
3. Third, within any internal loop, the algorithm will change the first base on either side of the loop (see Fig 3B). This prevents mismatches with the flanking pairs of the stack.
4. Fourth, in any internal loop with two bases on either side, a U and a G are placed on both sides. This is a strategy (U-G-U-G 'super-boost') discovered by Eterna players which drastically lowers the free energy of the RNA molecule and takes advantage of a special parameter for tandem G•U's in current RNA nearest-neighbor folding models.
5. Fifth, the SAP places a guanine at the beginning of any hairpin loop to lower the overall energy of the hairpin, again reflecting a special trend in current RNA nearest-neighbor folding models.
6. Lastly, if the puzzle is still not folding into the desired shape, the SAP will randomly flip base pairs in target helices that are not predicted to fold properly. Eterna players often do this flipping, as an incorrect orientation could potentially be the reason why that specific area is not folding correctly.

   See Fig 2 for illustrations of these six strategies.

Puzzle-solving efficiency

The successful puzzle solution by EternaBrain-SAP would not be useful if it takes more time than experienced players to solve puzzles. S4 Fig shows player and EternaBrain-SAP times for puzzles of varying lengths. On one hand, EternaBrain-SAP generally takes more moves (approximately 2-fold) than top Eterna players to solve most puzzles, across all puzzle lengths. However, this disadvantage is ameliorated by the speed with which EternaBrain-SAP selects its moves: on a single 2.5 GHz dual-core Intel i5 CPU, the automated method takes less total time than top players across puzzles of varying lengths. We took random samples of 50 puzzles that both EternaBrain-SAP and players solved in order to see if there was a statistically significant difference in completion time. A two-sample t-test gave a p-value of 0.020, indicating that EternaBrain-SAP is significantly faster than players in solving Eterna puzzles despite no specific optimization for speed at this stage.

Supporting information

S1 File. List of Eterna contributors to this study. Participants who tooks part in solving the puzzles used to develop EternaBrain, with names listed in order of contributions, from most to least.
(DOCX)

S1 Fig. The twelve puzzles used in the *eternamoves-large* dataset. Most bases are adenine to represent the initial state of the puzzle before any mutations are made. Some bases that are not A represent "locked" bases which cannot be mutated. The 5' end of each puzzle is at the top left, with the puzzle drawn counter-clockwise from that point.
(PDF)

S2 Fig. (This page and previous two pages) The 78 puzzles used in the *eternamoves-select* dataset. Some puzzle structures are repeated because the locked bases (bases which cannot be mutated) are different for the two puzzles. The 5' end of each puzzle is at the top left, with the puzzle drawn counter-clockwise from that point.
(PDF)

S3 Fig. Training the EternaBrain convolutional neural network on *eternamoves-select*. (A) Cross-validation accuracy of location predictor per epoch of training. (B) Cross-validation accuracy of base predictor per epoch of training. An epoch of training is one complete pass of backpropagation, retraining, and receiving feedback of performance over all 30,447 solutions (305 mini-batches of 100 solutions each).
(PDF)

S4 Fig. Time and number of moves of EternaBrain-SAP compared to Eterna human players. (A) Median time and (B) number of moves needed to solve Eterna100 puzzles. Puzzles that were only solvable by Eterna human players are not shown.
(PDF)

S1 Table. Hyperparameter search for neural net architecture.
(DOCX)

S2 Table. Number of solutions and moves for top players used in *eternamoves-select*.
(DOCX)

Acknowledgments

We thank J. Nicol for expert technical assistance; J. Shi, M. Wu, P. Eastman, and B. Ramsundar for scientific discussions; and M. Gotrik for comments on the manuscript. We acknowledge the following Eterna participants for designing the puzzles used to develop EternaBrain: Kieros, jandersonlee, drake178, Brourd, hoglahoo, Janelle, eternacloud, Hyphema, RedSpah, nihilnove, steven123505, player4596, portalbob340, mat747, Jieux, redsoxwy, Eli Fisker, Red-Simple, Malcolm, firedrake969, pdub93, ElNando888, Dennis9600, Nidoking, cake. We further acknowledge Eterna participants for taking part in solving the puzzles used to develop EternaBrain; the names of these contributors are provided in S1 File.

Author Contributions

Conceptualization: Rohan V. Koodli, Benjamin Keep, Rhiju Das.

Data curation: Rohan V. Koodli, Benjamin Keep, Fernando Portela.

Formal analysis: Rohan V. Koodli, Rhiju Das.

Funding acquisition: Rhiju Das.

Investigation: Rohan V. Koodli, Rhiju Das.

Methodology: Benjamin Keep, Rhiju Das.

Software: Rohan V. Koodli.

Supervision: Benjamin Keep, Rhiju Das.

Validation: Katherine R. Coppess, Rhiju Das.

Writing – original draft: Rohan V. Koodli, Rhiju Das.

Writing – review & editing: Rohan V. Koodli, Benjamin Keep, Katherine R. Coppess, Fernando Portela, Rhiju Das.

References

1. Wiedenheft B., Sternberg S. H. & Doudna J. A. RNA-guided genetic silencing systems in bacteria and archaea. *Nature* 482, 331–338 (2012). https://doi.org/10.1038/nature10886 PMID: 22337052
2. Reynolds A. et al. Rational siRNA design for RNA interference. *Nat. Biotechnol.* 22, 326–330 (2004). https://doi.org/10.1038/nbt936 PMID: 14758366
3. Bonnet É., Rzążewski P. & Sikora F. Designing RNA Secondary Structures is Hard. *Research in Computational and Molecular Biology* 248 (2017).
4. Garcia-Martin J. A., Clote P. & Dotu I. RNAiFOLD: a constraint programming algorithm for RNA inverse folding and molecular design. *J. Bioinform. Comput. Biol.* 11, 1350001 (2013). https://doi.org/10.1142/S0219720013500017 PMID: 23600819
5. Taneda A. MODENA: a multi-objective RNA inverse folding. *Adv. Appl. Bioinform. Chem.* 4, 1–12 (2011). PMID: 21918633
6. Hofacker I. L. Vienna RNA secondary structure server. *Nucleic Acids Res.* 31, 3429–3431 (2003). https://doi.org/10.1093/nar/gkg599 PMID: 12824340
7. Busch A. & Backofen R. INFO-RNA—a server for fast inverse RNA folding satisfying sequence constraints. *Nucleic Acids Res.* 35, W310–3 (2007). https://doi.org/10.1093/nar/gkm218 PMID: 17452349
8. Andronescu M., Fejes A. P., Hutter F., Hoos H. H. & Condon A. A new algorithm for RNA secondary structure design. *J. Mol. Biol.* 336, 607–624 (2004). https://doi.org/10.1016/j.jmb.2003.12.041 PMID: 15095976
9. Wolfe B. R., Porubsky N. J., Zadeh J. N., Dirks R. M. & Pierce N. A. Constrained multistate sequence design for nucleic acid reaction pathway engineering. *J. Am. Chem. Soc.* 139, 3134–3144 (2017). https://doi.org/10.1021/jacs.6b12693 PMID: 28191938
10. Anderson-Lee J. et al. Principles for predicting RNA secondary structure design difficulty. *J. Mol. Biol.* 428, 748–757 (2016). https://doi.org/10.1016/j.jmb.2015.11.013 PMID: 26902426
11. Lee J. et al. RNA design rules from a massive open laboratory. *Proc Natl Acad Sci USA* 111, 2122–2127 (2014). https://doi.org/10.1073/pnas.1313039111 PMID: 24469816
12. Liu W. et al. A survey of deep neural network architectures and their applications. *Neurocomputing* 234, 11–26 (2017).
13. Silver D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484–489 (2016). https://doi.org/10.1038/nature16961 PMID: 26819042
14. Silver D. et al. Mastering the game of Go without human knowledge. *Nature* 550, 354–359 (2017). https://doi.org/10.1038/nature24270 PMID: 29052630
15. Eastman P., Shi J., Ramsundar B. & Pande V. S. Solving the RNA design problem with reinforcement learning. *PLoS Comput. Biol.* 14, e1006176 (2018). https://doi.org/10.1371/journal.pcbi.1006176 PMID: 29927936
16. Runge, F., Stoll, D., Falkner, S. & Hutter, F. Learning to Design RNA. arXiv preprint arXiv:1812.11951 (2018).
17. Sav, S., Hampson, D. J. D. & Tsang, H. H. SIMARD: A simulated annealing based RNA design algorithm with quality pre-selection strategies. in 2016 IEEE Symposium Series on Computational Intelligence (SSCI) 1–8 (IEEE, 2016). https://doi.org/10.1109/SSCI.2016.7849957
18. Shi J., Das R. & Pande V. S. SentRNA: Improving computational RNA design by incorporating a prior of human design strategies. *arXiv preprint arXiv*: 1803.03146 (2018).
19. Portela F. An unexpectedly effective Monte Carlo technique for the RNA inverse folding problem. *BioRxiv* (2018). https://doi.org/10.1101/345587
20. Kleinkauf R., Houwaart T., Backofen R. & Mann M. antaRNA—Multi-objective inverse folding of pseudoknot RNA using ant-colony optimization. *BMC Bioinformatics* 16, 389 (2015). https://doi.org/10.1186/s12859-015-0815-6 PMID: 26581440
21. Yang X., Yoshizoe K., Taneda A. & Tsuda K. RNA inverse folding using Monte Carlo tree search. *BMC Bioinformatics* 18, 468 (2017). https://doi.org/10.1186/s12859-017-1882-7 PMID: 29110632
22. Bellaousov S., Kayedkhordeh M., Peterson R. J. & Mathews D. H. Accelerated RNA secondary structure design using preselected sequences for helices and loops. *RNA* 24, 1555–1567 (2018). https://doi.org/10.1261/rna.066324.118 PMID: 30097542

23. Mathews D. H. et al. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc Natl Acad Sci USA* 101, 7287–7292 (2004). https://doi.org/10.1073/pnas.0401799101 PMID: 15123812

24. Wu J. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology*. Nanjing University. China 5–23 (2017).

25. Ramlan, E. I. & Zauner, K.-P. An Extended Dot-Bracket-Notation for Functional Nucleic Acids. in Oesterreichische Computer Gesellschaft (eds. E, C.-V., Freund, R., Oswald, M. & Salomaa, K.) 75–86 (Oesterreichische Computer Gesellschaft, 2008).

26. Srivastava N., Hinton G., Krizhevsky A., Sutskever I. & Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1929–1958 (2014).

27. Matthies M. C., Bienert S. & Torda A. E. Dynamics in sequence space for RNA secondary structure design. *J. Chem. Theory Comput.* 8, 3663–3670 (2012). https://doi.org/10.1021/ct300267j PMID: 26593011

28. Hampson, D. J. D. & Tsang, H. H. Using matching substructures as an optimization objective for RNA design. in 2017 IEEE Symposium Series on Computational Intelligence (SSCI) 1–7 (IEEE, 2017). https://doi.org/10.1109/SSCI.2017.8285345

29. Stone M. Cross-Validatory Choice and Assessment of Statistical Predictions. 111–147 (Journal of the Royal Statistical Society, 1974).

30. Minai A. A. & Williams R. D. On the derivatives of the sigmoid. Neural Netw. 6, 845–853 (1993).

31. Kingma D. P. & Ba J. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv: 1412.6980 (2014).

32. Abadi M. et al. Tensorflow: a system for large-scale machine learning. USENIX 16, 265–283 (2016).