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

Rohan V. Koodli[1,4], Benjamin Keep[2], Katherine R. Coppess[3], Fernando Portela[1], Eterna players[4,†], and Rhiju Das[1, 3, 4, *]

[1]Department of Biochemistry, [2]Department of Education, [3]Department of Physics, Stanford, CA 94305

and

[4]Eterna Massive Open Laboratory

[†] List of Eterna players who contributed move sets for EternaBrain are listed in Supplemental Table 1.

* Corresponding Author. E-mail: rhiju@stanford.edu

ABSTRACT

Folded RNA molecules underlie emerging approaches to disease detection and gene therapy. These applications require RNA sequences that fold into target base-pairing patterns, but computational algorithms generally remain inadequate for these RNA secondary structure design tasks. The Eterna project has collected over 1 million player moves by crowdsourcing RNA designs in the form of puzzles that reach extraordinary difficulty. Here, we present these data in the *eternamoves* repository and test their utility in training a multilayer convolutional neural network to predict moves. When pipelined with hand-coded move combinations developed by the Eterna community, the resulting EternaBrain method solves 61 out of 100 independent RNA design puzzles in the Eterna100 benchmark. EternaBrain surpasses all six other prior algorithms that were not informed by Eterna strategies and suggests a path for automated RNA design to achieve human-competitive performance.

Keywords: Deep learning, artificial intelligence, RNA design, secondary structure

INTRODUCTION

Due to its versatility and important roles throughout fundamental molecular 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 determined by its structure, and so these and future RNA technologies require computational methods to design specific base sequences that fold into a target structure or set of structures to carry out 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].

Numerous groups have developed RNA secondary structure design algorithms, including MODENA [4], RNAinverse [5], INFO-RNA [6], RNA-SSD [7], and NUPACK [8]. However, these methods have typically been tested on simple structures that do not capture the symmetries, duplex lengths, and sizes seen in natural RNAs or needed for biotechnology applications [9]. The incompleteness of these prior methods became clear with the release of the *Eterna* game [10] in 2011, which crowdsources RNA design as puzzles through an internet-scale videogame (Figure 1). Eterna players learn the basics of RNA design through examples that are initially tested through a computational model of secondary structure folding evaluated *in silico*. Then the players submit their RNA designs for lab challenges to receive wet-lab feedback on how their molecules fold through *in vitro* experiments on a weekly time scale, which expose the 'reality gap' in RNA design – the mismatch between current computational folding models and experiment. In preparation for these experimental challenges, players also challenge fellow

players 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 [9]. Since Eterna's inception, the community has grown to over 250,000 Eterna registered players as of 2018, with over 17,000 player-created puzzles. This community has been successful in designing RNAs, consistently outperforming prior RNA design algorithms in both *in silico* and *in vitro* tests [4][9].

As part of these efforts, players discovered classes of RNA secondary structures for which prior algorithms cannot find sequence solutions even *in silico*, i.e., when folded with computational energy models that can be rapidly evaluated [9]. Solutions to these design problems under the same computational energy models can be discovered by experienced human Eterna players. 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 players curated a benchmark of 100 problems of varying difficulty, called the 'Eterna100' [9]. By offering more difficult challenges than prior benchmark sets, these secondary structures allow stringent tests of advances in computational design methods. Second, player-created tutorial puzzles have 'canonized' new strategies for *in silico* RNA design. In 2016, Eterna developers have installed these puzzles as the standard progression of problems for new players. Also since 2016, Eterna players have agreed to donate the sequences of moves that lead to successful solutions to scientific research, resulting in a vast repository of player move set data. The current data set includes well over 1 million such moves.

The availability of such large move sets and player solutions suggests new approaches to solving the RNA secondary structure design problem. While previous RNA design algorithms used hierarchical decomposition of target structures, genetic algorithms, and probabilistic sampling of sequences [4, 6, 7], a recent generation of classification, translation, and game-playing algorithms make powerful use of statistical pattern recognition through multi-layer artificial neural networks [11]. Striking examples include the successes of Google DeepMind's *Go* game playing algorithms, AlphaGo [12] and AlphaGo Zero [13], which now outcompete expert human players. These approaches were inspired by the discovery that expert player 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 19x19=361 board positions for pieces, giving a baseline random guess accuracy of less than 0.5%) [12] [14].

Here, we test whether similarly high accuracies can be achieved in understanding and predicting moves of Eterna players. We present a data set of 1.8 million moves, called *eternamoves*, appropriately cleaned and labeled for machine learning applications. We then report tests of CNNs to predict these moves given the game state. We find that a subset of moves made by the most experienced players are sufficiently stereotyped to allow training of an automated neural network move predictor with accuracy above random baseline. We then characterize how well the resulting predictor can go beyond simply predicting moves one-at-a-time to instead conduct a sequence of moves to fully solve novel RNA secondary structure design problems (as evaluated by *in silico* folding models). We demonstrate an unexpected synergy between player's 'hand-crafted' strategies and this neural network approach, a marriage

of classic and modern game-playing strategies. Finally, we show that the resulting EternaBrain algorithm achieves performance on the Eterna100 benchmark that exceeds previously published methods that did not make use of Eterna strategies. In the discussion, we compare our results to those of SentRNA, an Eterna-inspired deep learning approach that has been developed in parallel with EternaBrain [15]. Taken together, these results suggest a promising route to achieving automated RNA design methods that might approach top human designers.

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 [12] [13] and also because of their ability to capture features used in actual Eterna gameplay. A CNN [16] is a specific type of neural network that hierarchically builds up complex features from simpler features that neighbor each other; it consists of convolutional and pooling layers, which retrieve subsections of the input features and make their own larger-scale features. A CNN's features and relationships are initially random but then learned from a large training set of input data and the output labels (here, the actual move made by a player on a given sequence). A CNN is a natural choice for tackling this problem because it mimics how human players visualize Eterna puzzles, learning features of the data from spatial information seen in the puzzle interface as well as in solution browsers.

We started by training the model on all player data with 1.8 million moves spanning 12 representative puzzles from the Eterna progression dataset, 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 [16]), and calculated Gibbs free energy of folding. These data on game state are passed as input features to the CNN (Table 2). The location of the player's chosen nucleotide change and the nucleotide identity itself were passed as labels to the CNN (Table 2). We chose to train one 'base predictor' CNN to predict the base change made by a player given the location of the change, and a separate 'location predictor' CNN to predict the location of the change. For the base predictor, we did not enforce that the neural network change the base through the move; this choice allowed us to test if the neural network learned that very basic fact about moves. For cross-validation, we split the data into train and test sets at random (see Methods). Training took three days using 4 NVidia Titan X GPUs.

As a baseline, random guessing of moves would give accuracies of 0.33 (if the move is forced to change the base to one of the three other bases) for the base predictor and 0.019 for the location predictor (based on the average inverse length of the puzzles). The best network that we trained on this large move set achieved training accuracies for the base predictor and location predictor (0.50 and 0.10, respectively) that were both higher than random guessing. However, the model's test accuracies (0.34 and 0.021, respectively) were not much better than baseline random guessing. Indeed, we observed poor results in subsequent tests of this model through full puzzle playouts. For these playouts, we had the CNN predict base and location for the puzzle starting from its initial sequence, updated the puzzle state according to ViennaRNA energy calculations,

and repeated the CNN's prediction mechanism. The model could not solve *Simple Hairpin* (the secondary structure designed by EternaBrain did not match the desired secondary structure), the easiest puzzle on the Eterna100, even with variations to allow stochastic choice of moves ranked by the CNN (see Methods).

The low test accuracies and problems in actual puzzle playout suggested that the model was severely overfitting to the training data and not generalizing well to other Eterna puzzles that were outside the dataset. To counteract the overfitting, we varied the dropout rate of the network. Dropout randomly drops a certain amount of nodes in each layer so that the network does not become too complex and overfit to the training data. However, dropout only increased the test accuracy marginally, suggesting that there was high variance in the training data [18], i.e., players were using a highly heterogenous set of strategies or clicking randomly without strategy. Other parameter changes to the network (use of a single CNN to predict base and location; change in the type of neural network; number of layers; activation functions) 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. One possible explanation is that inexperienced players may solve puzzles through trial-and-error, and most moves therefore do not convey useful strategies.

Training on Selected Subsets of Players and Puzzles

We hypothesized that training on fewer Eterna players would minimize the variance. Therefore, we decided to be more selective and trained on solutions from players with the most

experience in solving RNA design problems. To select these 'expert' Eterna players, we counted the number of puzzles solved for all Eterna players. For the cutoff, we decided to select players who were in the 99th percentile of all Eterna players, which left us with 72 Eterna players in the dataset and 30,447 moves (Figure 2a; Supplemental Table S2). We also selected 92 puzzles from the Eterna labs and progression that demonstrated key strategies used by Eterna players, and used all movesets for these puzzles. We called this specially culled subset of moves the *eternamoves-select* data set. In addition to narrowing the training data to experts and specialized puzzles, we added a more explicit representation of RNA secondary structure using pairmaps. As an alternative to the dot-bracket notation, 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 2.

Using these tightly filtered training data and additional structure representations increased the predictive power of the model. The final test accuracies for cross-validation were 0.51 for the base predictor and 0.34 for the location predictor (Figures 2b and 2c; Table 3). These test accuracies were higher than random baselines for this test set of 0.33 and 0.031, respectively.

The test accuracies on *eternamoves-select* suggested that moves of experienced players and moves on specially designed problems were sufficiently stereotyped to allow their automatic prediction. To further explore whether this move predictability involved strategies that generalized across different puzzles or between different expert players, we split the training and test sets in different ways. First, we trained our model on 15,182 moves from half of the puzzles in our training set. The test accuracies for another 15,282 moves from the held-out puzzles were 0.26 and 0.023 (base and location, respectively). These numbers are lower than for our initial CNN tests on *eternamoves-select*, where the training and test set drew moves from the same

puzzles. Indeed, the base prediction accuracy dropped below the random guess baseline value of 0.33 when the puzzles in the training and test set were separated. These observations suggested that the CNN predictions depend on the similarity between puzzles seen in training and testing. Second, we trained our model on moves derived from half of the expert players in our training set. The test accuracies on the moves from the other expert players were 0.38 and 0.11. These accuracies remained above random baselines (0.33 and 0.031, respectively) and suggest that while expert players vary in their puzzle solving styles, there are commonalities that can be successfully learned the CNN. Last, we trained our model on the 587 moves of a single expert player. The accuracies on the 29,877 moves of all other experts were 0.27 and 0.011, 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. Overall, we chose to move forward with the original CNN (trained on a random half of *eternamoves-select* across all puzzles and expert players) which gave the highest training and test accuracies, 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.

Playouts on the Eterna100 benchmark

After the cross-validation tests above, we set out to solve a complete puzzle with our CNN model trained on *eternamoves-select* and the simple playout scheme described above. Encouragingly, the model was able to solve several puzzles completely, including the first *Simple Hairpin* puzzle in the Eterna100 benchmark. However, it often could not completely solve longer puzzles, those greater than 40 nucleotides in length. In general, this approach of using the CNNs

to predict moves was able to solve up to 85 percent of any given puzzle, meaning that up to 85 percent of the base pairs predicted for the CNN solution matched up with the base pairs of the target secondary structure. However, simple inspection of the CNN's series of moves revealed minor but obvious mistakes, such as incorrect base pairings, mostly due to the randomness in the stochastic base selector. In order to prevent some of these mistakes, we followed the CNN with a stage that we called single action playout (SAP). SAP uses canonical strategies that are standard among experienced Eterna players and are taught to new players through the game's puzzle progression. 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. Each of the SAP strategies is summarized in Figure 3.

We called this enhanced CNN-then-SAP pipeline EternaBrain and tested it against the Eterna100 benchmark. These 100 puzzles span a wide range of different structural motifs. The results are shown in Figure 4a. We compared EternaBrain to six of the top algorithms: RNA-SSD, RNAinverse, DSS-Opt, NUPACK, INFO-RNA, and MODENA [18]. These algorithms solved 27, 28, 47, 48, 50, and 54 of 100 puzzles [9]. EternaBrain solved 61 of the 100 puzzles, surpassing the best prior algorithm MODENA by 7 puzzles.

Importantly, both the CNN and the SAP stages were necessary for this level of success. Using EternaBrain's CNN-based moves alone solves only 20 puzzles on the Eterna100. Using the SAP alone (i.e., just hand-coded canonical player strategies) solves 50 puzzles on the Eterna100, still 11 short of the CNN-SAP combination. We confirmed that several choices that we made for the CNN architecture and game state representations were important for the success of EternaBrain. Removing the sequence, the pairmap, or the dot-bracket representation from the

input features or not using dropout also decreased performance in Eterna100 playouts (from 61 to 51, 52, 56, and 57 puzzles solved, respectively). We also note that while EternaBrain's CNN-based moves have a stochastic component, its success on puzzles is consistent across runs with different random seeds (Table 1).

Performance on specific features of difficult puzzles

Different puzzles in the Eterna100 benchmark showcase different features that are difficult for RNA secondary structure design. Comparison of EternaBrain's performance across these different puzzle types clarifies its current abilities and limitations.

Simple Motifs - Stacks, Loops, Hairpins

The EternaBrain 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). These motifs frequently appeared in the training set, and EternaBrain's CNNs were able to solve them well when encountered in the separate test puzzles of the Eterna100. An example of a difficult puzzle that EternaBrain's CNN could solve well is *U* (Figure 5a), which was not solvable by five of the six prior algorithms or by SAP alone. 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 (to strengthen a base pair stack) and the G-tetraloop boost (a special nucleotide at the beginning of a loop to stabilize the loop; Figure 3). While these strategies are also sufficiently well known so as to be encoded in SAP (Figure 3), EternaBrain

also learned additional patterns that were not encoded in the SAP, as is demonstrated by the inability of SAP alone to solve *U* (Figure 4). These strategies were additional patterns specific to the puzzle which the CNN had to learn on its own.

Specific Orientation of Base Pairs

If the puzzle contained motifs whose stabilities depended on specific base pair compositions and orientations, the CNN was inadequate and SAP was critical for EternaBrain's success. An example of such a puzzle was *Chicken Tracks* (Figure 5b) which required a very specific orientation of the base pairs. The CNN was not accurate enough to orient base pairings precisely enough to do this on its own. 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

One of EternaBrain's successes can be attributed to its ability to solve repetitive structures better than previous automated RNA design algorithms. For repetitive structures, the SAP was able to solve many puzzles on its own. Such puzzles include *Thunderbolt*, *Shortie 4*, and *Shortie 6* (Figures 5c-d). For example, previous algorithms could solve puzzles with few stacks, such as *Shortie 4* (Figure 5d). However, when the number of stacks increased but the overall shape of the puzzle stayed the same, such as *Shortie 6* (Figure 5d), other algorithms struggled. EternaBrain, however, could solve both *Shortie 4* and *Shortie 6*. Given the success of

SAP alone on these puzzles, it appears that the moveset-trained CNN of EternaBrain was not needed for this success.

EternaBrain's failure to solve other puzzles in the benchmark appears to be a result of the incompleteness of the training data and player strategies in the algorithm. For example, *Hard Y* (Figure 5e and 5f), requires uncommon strategies, such as 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 did not help solve this puzzle since reorienting bases did not change the stability of the zigzag. Training the CNN on larger movesets and incorporating more sophisticated player strategies in SAP might resolve these issues and help EternaBrain complete more complex puzzles.

Efficiency

The successful puzzle solution by EternaBrain would not be useful if it takes more time than experienced players to solve puzzles. Figure 5 shows player and EternaBrain times for puzzles of varying lengths. Compared to expert Eterna players, EternaBrain, on a single 2.5 GHz Intel i5 CPU, takes less time across puzzles of varying lengths. EternaBrain's improvement in speed over human players becomes greater as the puzzles get longer, which might be expected since human players often take breaks while solving long puzzles. While EternaBrain takes less time to solve puzzles, it takes more moves on average to solve a puzzle than human Eterna players. We took random samples of 50 puzzles that both EternaBrain 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 is significantly faster than players in solving Eterna puzzles despite no specific optimization for speed at this stage.

DISCUSSION

EternaBrain attempts to solve the RNA secondary structure design problem based on a vast compilation of human player moves. We decided on the use of a convolutional neural network (CNN) since it can be 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 what mutation to make next. To reduce variance in the training set, we found it important to use movesets from expert players, leading to test accuracies better than random guesses. For the location predictor in particular, the test accuracy of 0.34 dramatically exceeded the baseline prediction accuracy for random guessing (0.01), similar to important early results in the development of Go playing automata.

Despite this non-trivial move prediction accuracy, we found that the CNNs alone were not sufficient to solve longer puzzles. Indeed, alternative splitting of test and training sets suggested that the CNNs' prediction accuracy depends on similarity of puzzles in the training set and any new challenges; such inability to extrapolate "out of sample" has been noted to be a limitation of artificial neural networks [20]. We therefore added a single-action playout based on compiled Eterna player strategies to aid the CNNs. We tested this combined EternaBrain algorithm on the complete Eterna100. On one hand, EternaBrain presently does not match the level of the top six Eterna human players who can solve all 100 puzzles. On the other hand,

EternaBrain outperforms all previous algorithms. For the cases it solves, EternaBrain is able to do so in a shorter time than human players.

It is interesting to compare EternaBrain to SentRNA, which also uses artificial neural networks to distill potentially useful information from Eterna's *in silico* gameplay [15]. SentRNA seeks to find solutions to RNA secondary design problems in 'one shot' rather than through EternaBrain's iterative moves. Furthermore, SentRNA differs from EternaBrain as it is trained on final player solutions 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 study find that neural network approaches alone give poor performance in test puzzles on the Eterna100 benchmark, in both cases solving less than half of the benchmark. The success of both studies required pipelining starting solutions from neural network approaches with hand-coded strategies that Eterna players collectively learned and 'canonized' in tutorial puzzles for new players. Interestingly, there are several puzzles that are solved by EternaBrain and not SentRNA (e.g., *Thunderbolt*; Figure 5c) and some that are solved by SentRNA and not EternaBrain (e.g., *Hard Y*; Figures 5e and f).

Given these initial results, we propose future updates that may allow automated design to reach the level of expert players. First, EternaBrain and SentRNA could be integrated, with each one providing starter solutions for the other. Second, the hand-coded player strategies used in both EternaBrain and SentRNA often involve multiple moves; these strategies could possibly be captured if EternaBrain's CNN is trained to make moves based not just on its current game state but also including immediately previous moves as input. Replacing EternaBrain's SAP with a

Monte Carlo tree search algorithm [20]— a tool used in several AI game-play algorithms — may be useful but would significantly increase computational costs. Last, implementing a reinforcement learning algorithm based on player strategies or large-scale playouts by EternaBrain on over 10,000 single-structure puzzles created by Eterna Players is worth exploration; this type of approach has been powerful recently in game-playing tasks but requires extensive computation for training [12].

In addition to its prospects for achieving human-competitive performance on single structure design, we propose that the EternaBrain framework will be useful for design of ligand-responsive multi-state riboswitches and for rapid design of large batches of RNA designs for high throughput *in vitro* tests, going beyond the *in silico* tests herein. For both problems, datasets involving hundreds of thousands of RNA molecules are accumulating in the Eterna project [21] and should provide rich resources for training EternaBrain and other AI-style approaches.

ACKNOWLEDGMENTS

We thank J. Nicol for expert technical assistance and J. Shi, M. Wu, P. Eastman, and B. Ramsundar for scientific discussions. 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.).

METHODS

Code Availability

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

Data Collection

We obtained data from historical logs of moves that Eterna players made while solving puzzles through 2016-2017. The data contain the progression of each location and nucleotide change as a player solves an Eterna puzzle. The data contain only move sets for successfully solved puzzles. Data were collected in accord with Stanford IRB Protocol 34669.

Data Encoding

Through the Eterna puzzle-solving interface (Figure 1b, 1c, 1d), 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" structure -- the secondary structure 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 [5], which provides the default energy model in Eterna.

All information given to Eterna players was encoded before being passed into the CNN (see Figure 6). Such information included the nucleotide sequence, natural and target structure and pairmaps, natural 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 natural 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 basepair. 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 [25] 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 2.

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 [23] activation function, and Adam [24] optimizer to minimize the error of the neural network. Construction and training of CNNs were carried out using Google's TensorFlow [25] machine learning framework. The models discussed in this work were trained on Nvidia Titan X GPUs available on Stanford's Sherlock cluster.

The CNNs alone were not sufficient to completely solve a puzzle. A collection of canonical strategies for solving Eterna puzzles has been compiled by players. To complement the CNNs, we implemented player strategies using the SAP. 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 Figure 3. The SAP change specific nucleotides in the puzzle according to the player strategy and would compare 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 natural 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 Figure 3).

During development of the model, we evaluated the model 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 differently [26]. Playout tests were carried out on the 100 secondary structures of the Eterna100 benchmark [9]. To ensure fair comparison to prior work [9], 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.

LEGENDS

Figure 1. Overview of Eterna and EternaBrain.

(A) Overview of EternaBrain framework.

(B-C) Puzzle-solving interface presented to human players of Eterna.

(B) The desired target structure for the RNA molecule, as indicated by the bullseye in the bottom left (with an orange square).

(C) The nature-mode structure indicated by the leaf in the bottom left. Since the bases in the top right should be paired with each other, this puzzle is not yet folding correctly; this status is shown by the red indicator in the top left corner.

(D) 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.

Figure 2. Distribution of Player Experience and Accuracies

(A) The distribution of player experience.

(B) Cross-validation accuracy of location predictor.

(C) Cross-validation accuracy of base predictor.

Figure 3. The 6 strategies included in the SAP.

(A) The original state of the puzzle

(B) Correcting incorrect pairings

(C) Changing end pairs to G-C

(D) G-internal loop boost

(E) U-G-U-G super boost

(F) G-tetraloop boost

(G) Reorienting base pairs

Figure 4. EternaBrain Performance.

(A) Performance of EternaBrain and 6 top algorithms on Eterna100 benchmark.

(B) Performance of Alternative Model Constructions

Figure 5. Puzzles

(A) U Solution

(B) Thunderbolt Solution

(C) Chicken Tracks Solution

(D) Shortie 4 Solution

(E) Shortie 6 Solution

(F) Hard Y - Natural Structure

(G) Hard Y - Target Structure

Figure 6. Time needed to solve puzzles by Eterna human experts and EternaBrain. Data are median times; puzzles that were only solvable by human experts are not shown.

Table 1. Number of times out of 5 EternaBrain could solve each EternaBrain puzzle.

Table 2. How features were encoded in the EternaBrain convolutional neural networks.

Table 3. Training and test accuracies of convolutional neural networks trained on the filtered *eternamoves-select* data set.

REFERENCES

1. Wiedenheft, B., Sternberg, S. H., & Doudna, J. A. (2012). RNA-guided genetic silencing systems in bacteria and archaea. *Nature*, *482*(7385), 331.

2. Reynolds, A., Leake, D., Boese, Q., Scaringe, S., Marshall, W. S., & Khvorova, A. (2004). Rational siRNA design for RNA interference. Nature biotechnology, 22(3), 326.

3. Bonnet, É., Rzążewski, P., & Sikora, F. (2017). Designing RNA Secondary Structures is Hard. *arXiv preprint arXiv:1710.11513*.

4. Taneda, A. (2011). MODENA: a multi-objective RNA inverse folding. *Advances and applications in bioinformatics and chemistry: AABC*, *4*, 1-12.

5. Hofacker, I. L. (2003). Vienna RNA secondary structure server. *Nucleic acids research*, *31*(13), 3429-3431.

6. Busch, A., & Backofen, R. (2007). INFO-RNA—a server for fast inverse RNA folding satisfying sequence constraints. *Nucleic acids research*, *35*(suppl_2), W310-W313.

7. Andronescu, M., Fejes, A. P., Hutter, F., Hoos, H. H., & Condon, A. (2004). A new algorithm for RNA secondary structure design. *Journal of molecular biology*, *336*(3), 607-624.

8. Wolfe, B. R., Porubsky, N. J., Zadeh, J. N., Dirks, R. M., & Pierce, N. A. (2017). Constrained multistate sequence design for nucleic acid reaction pathway engineering. *Journal of the American Chemical Society*, *139*(8), 3134-3144.

9. Anderson-Lee, J., Fisker, E., Kosaraju, V., Wu, M., Kong, J., Lee, J., ... & Das, R. (2016). Principles for predicting RNA secondary structure design difficulty. *Journal of molecular biology*, *428*(5), 748-757.

10. Lee, J., Kladwang, W., Lee, M., Cantu, D., Azizyan, M., Kim, H., ... & Das, R. (2014). RNA design rules from a massive open laboratory. *Proceedings of the National Academy of Sciences*, *111*(6), 2122-2127.

11. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11-26.

12. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484-489.

13. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Chen, Y. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354.

14. Shi, J., Das, R., & Pande, V. S. (2018). SentRNA: Improving computational RNA design by incorporating a prior of human design strategies. *arXiv preprint arXiv:1803.03146*.

15. Clark, C., & Storkey, A. (2015, June). Training deep convolutional neural networks to play go. In *International Conference on Machine Learning* (pp. 1766-1774).

16. Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China.*

17. Ramlan, E. I., & Zauner, K. P. (2008). An extended dot-bracket-notation for functional nucleic acids. In E. Csuhaj-Varju´, R. Freund, M. Oswald and K. Salomaa (Eds.), *International Workshop on Computing with Biomolecules* (pp. 75-86).

18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

19. Matthies, M. C., Bienert, S., & Torda, A. E. (2012). Dynamics in sequence space for RNA secondary structure design. *Journal of chemical theory and computation*, *8*(10), 3663-3670.

20. Matthies, M. C., Bienert, S., & Torda, A. E. (2012). Dynamics in sequence space for RNA secondary structure design. *Journal of chemical theory and computation*, *8*(10), 3663-3670.

21. Wu, M. J., Andreasson, J. O., Kladwang, W., Greenleaf, W. J., & Das, R. (2017). Prospects for recurrent neural network models to learn RNA biophysics from high-throughput data. *bioRxiv*, 227611.

22. Chaslot, G., Bakkes, S., Szita, I., & Spronck, P. (2008). Monte-Carlo Tree Search: A New Framework for Game AI. In *AIIDE*.

23. Minai, A. A., & Williams, R. D. (1993). On the derivatives of the sigmoid. *Neural Networks*, *6*(6), 845-853.

24. Kingma, D. P., & Adam, B. J. (2017). A Method for Stochastic Optimization. Cornell University Library. *arXiv preprint arXiv:1412.6980*.

25. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* (Vol. 16, pp. 265-283).

26. Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, 111-147.

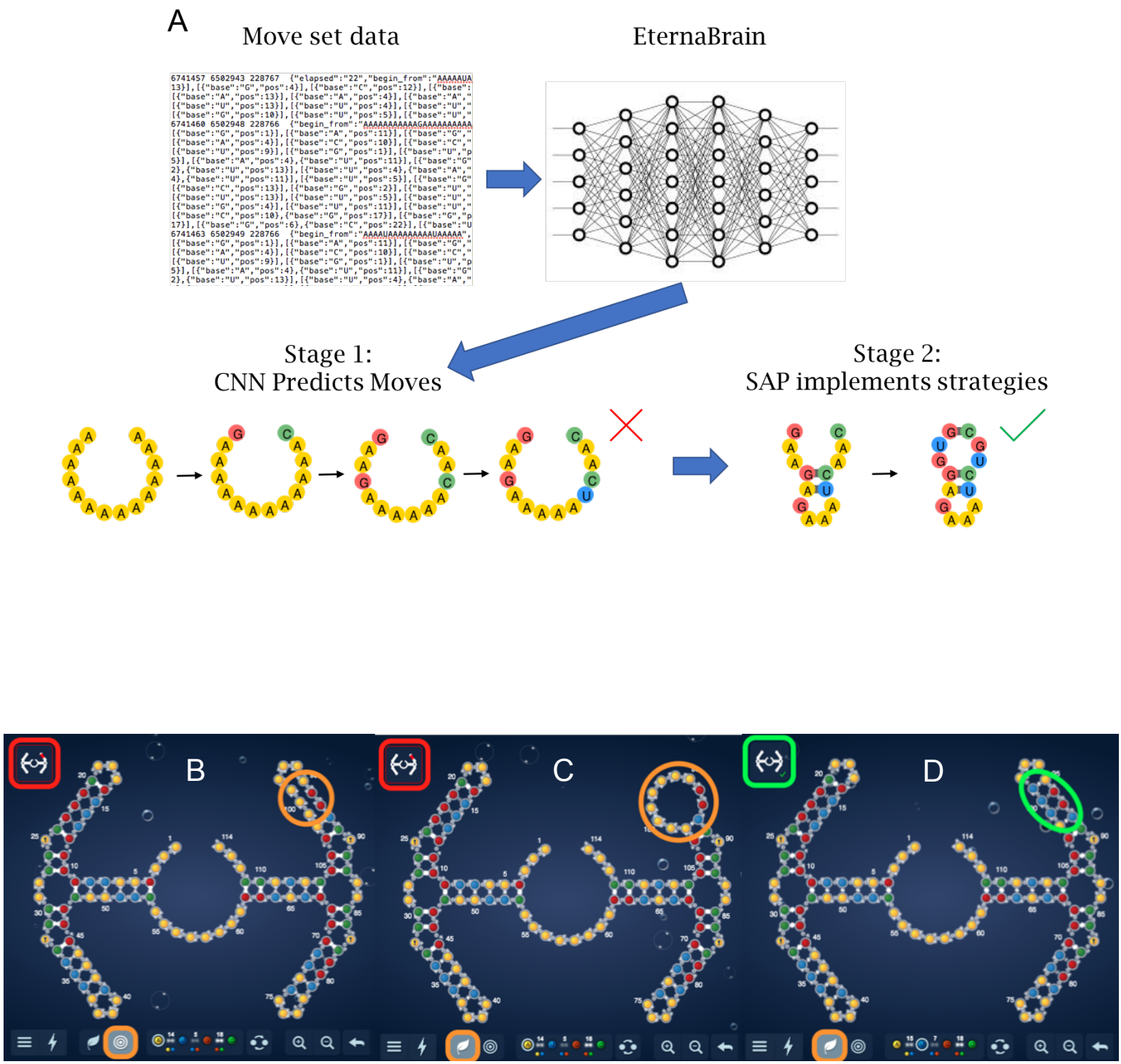
A
Move set data
EternaBrain
Stage 1:
CNN Predicts Moves
Stage 2:
SAP implements strategies
B
C
D

Figure 1

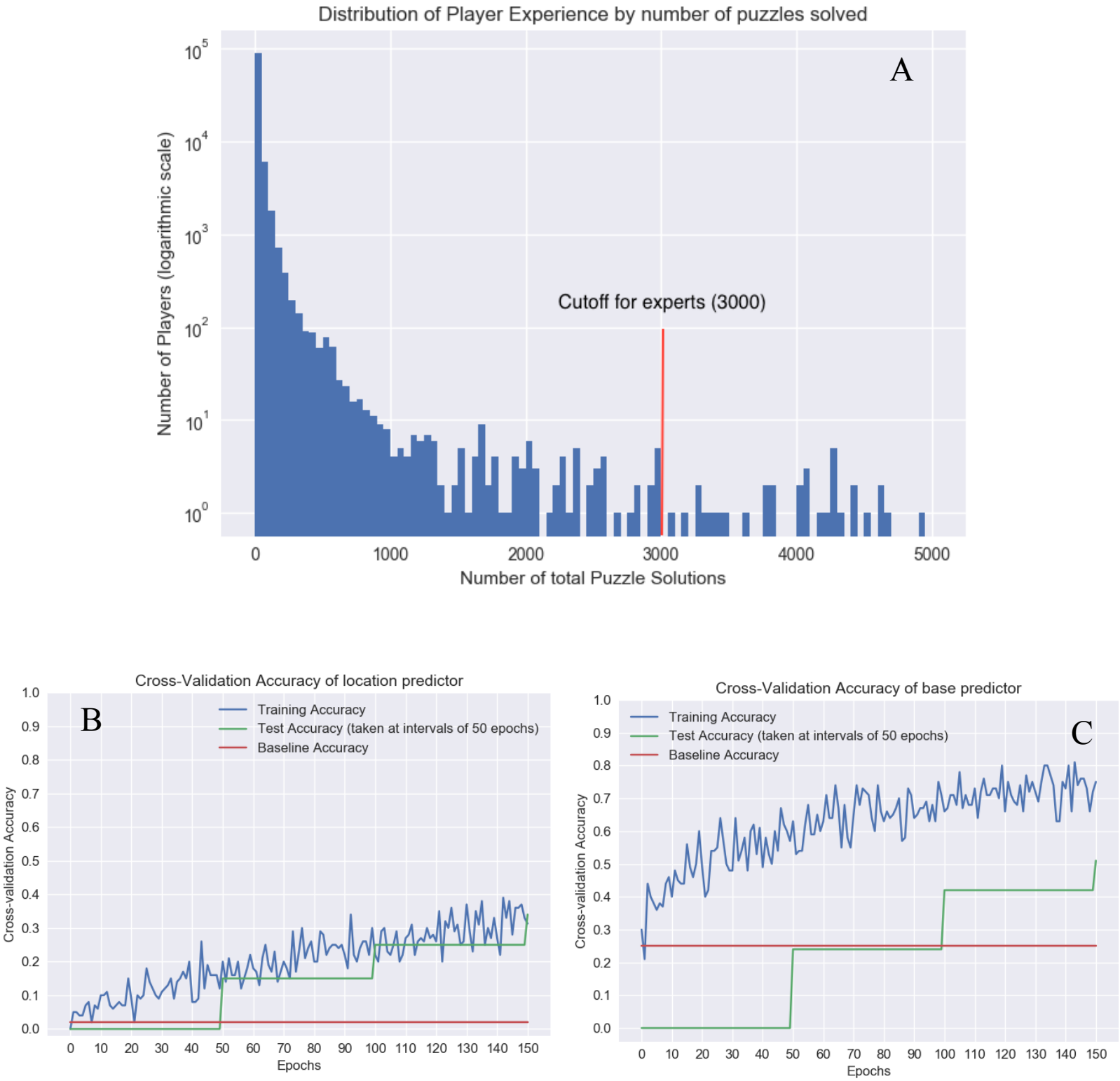
Distribution of Player Experience by number of puzzles solved
A
Cutoff for experts (3000)
Number of Players (logarithmic scale)
Number of total Puzzle Solutions
B
Cross-Validation Accuracy of location predictor
Training Accuracy
Test Accuracy (taken at intervals of 50 epochs)
Baseline Accuracy
Cross-validation Accuracy
Epochs
C
Cross-Validation Accuracy of base predictor
Training Accuracy
Test Accuracy (taken at intervals of 50 epochs)
Baseline Accuracy
Cross-validation Accuracy
Epochs

Figure 2

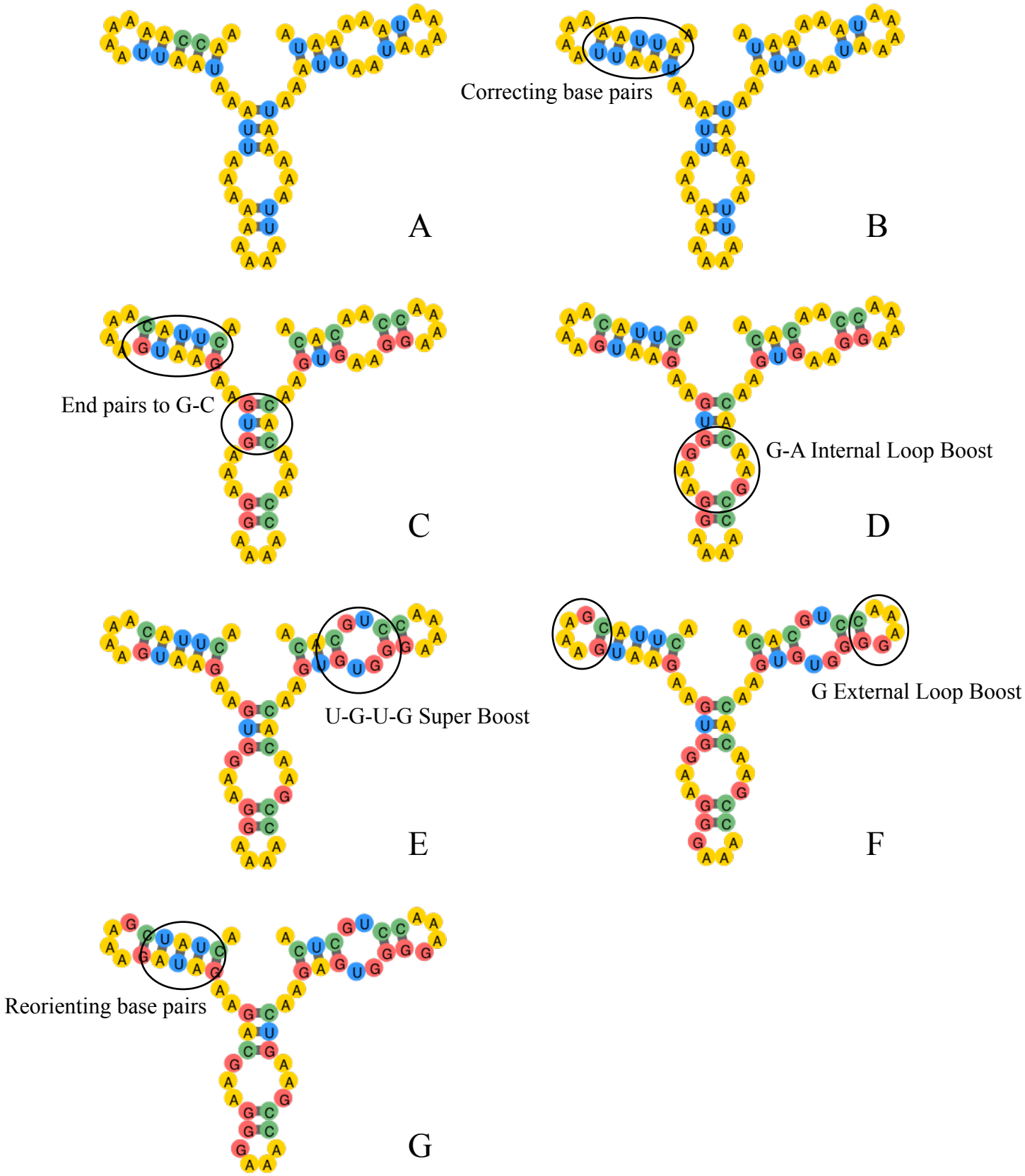
A
Correcting base pairs
B
End pairs to G-C
C
G-A Internal Loop Boost
D
U-G-U-G Super Boost
E
G External Loop Boost
F
Reorienting base pairs
G

Figure 3

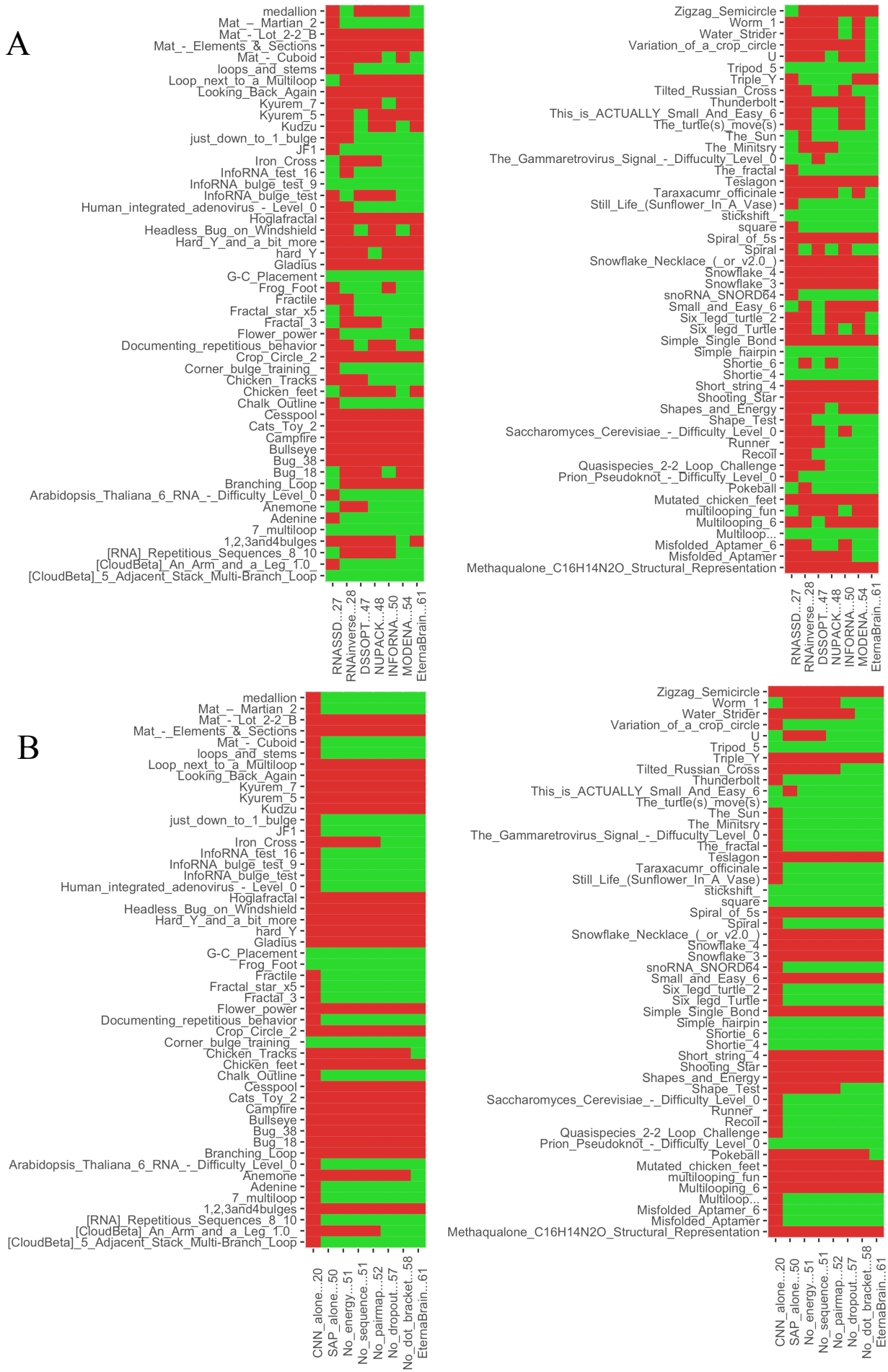
A
medallion
Mat_–_Martian_2
Mat_-_Lot_2-2_B
Mat_-_Elements_&_Sections
Mat_-_Cuboid
loops_and_stems
Loop_next_to_a_Multiloop
Looking_Back_Again
Kyurem_7
Kyurem_5
Kudzu
just_down_to_1_bulge
JF1
Iron_Cross
InfoRNA_test_16
InfoRNA_bulge_test_9
InfoRNA_bulge_test
Human_integrated_adenovirus_-_Level_0
Hoglafractal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

Figure 4

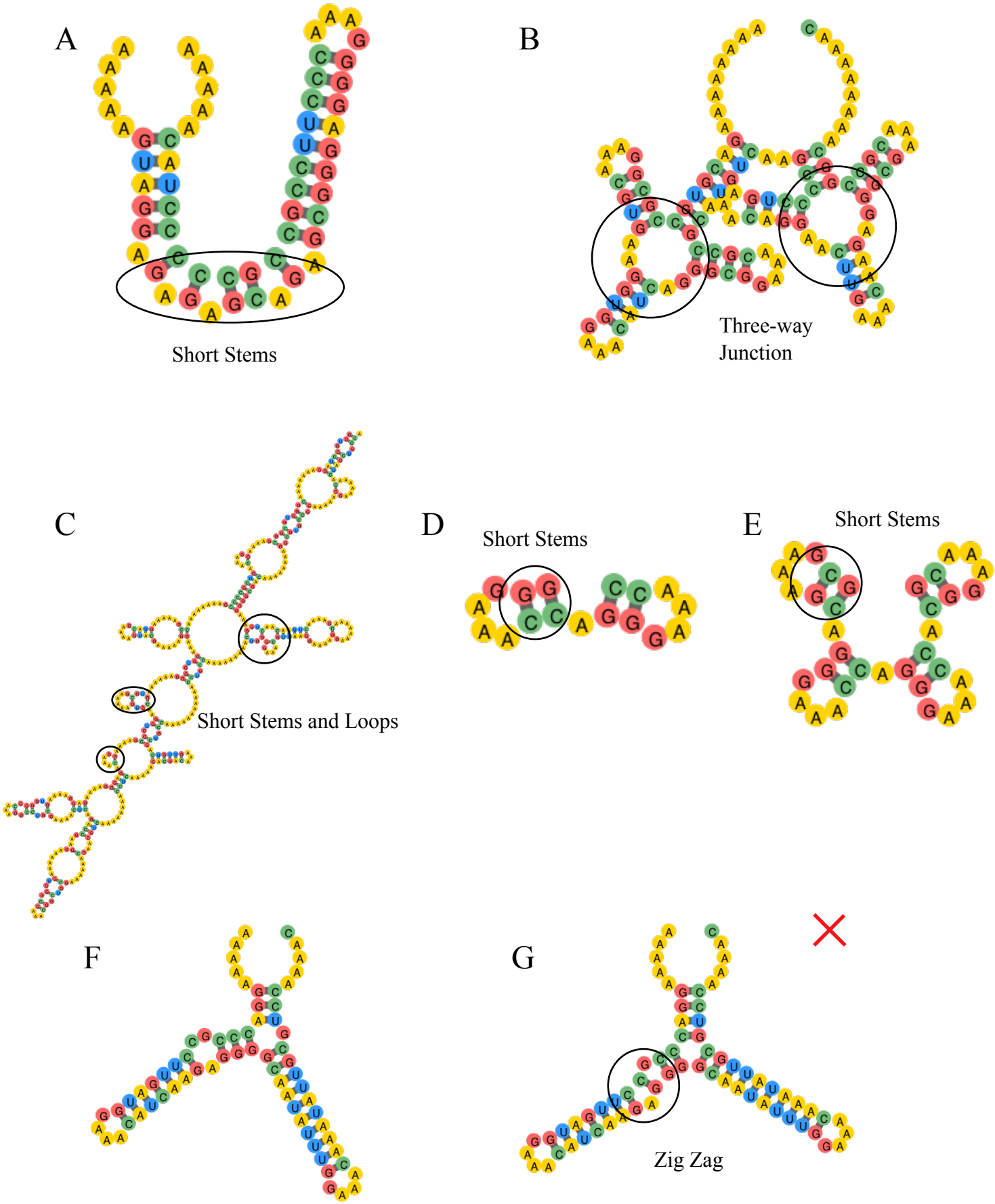
A
Short Stems
B
Three-way
Junction
C
Short Stems and Loops
D
Short Stems
E
Short Stems
F
G
Zig Zag

Figure 5

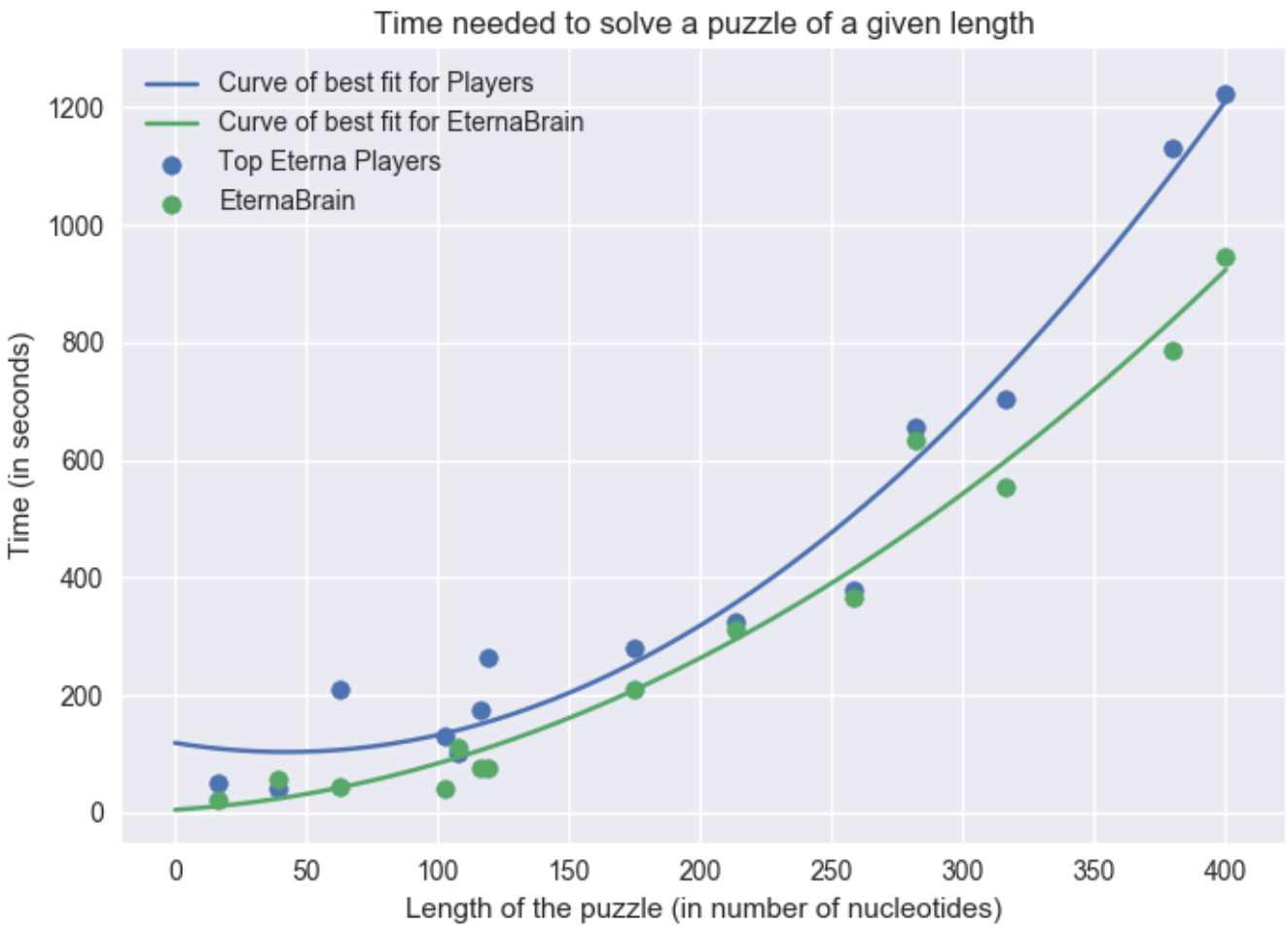
Time needed to solve a puzzle of a given length
Curve of best fit for Players
Curve of best fit for EternaBrain
Top Eterna Players
EternaBrain
Time (in seconds)
0
200
400
600
800
1000
1200
0
50
100
150
200
250
300
350
400
Length of the puzzle (in number of nucleotides)

Figure 6

Puzzle	Number of Times Solved out of 5
medallion	4
Mat – Martian 2	5
Mat - Cuboid	3
loops and stems	5
just down to 1 bulge	3
JF1	5
Iron Cross	5
InfoRNA test 16	5
InfoRNA bulgetTest 9	4
InfoRNA bulge test	5
Human integrated adenovirus – Level 0	4
G-C Placement	5
Frog Foot	5
Fractile	5
Fractal star x5	5
Fractal 3	5
Documenting repetitious behavior	5
Corner bulge training	5
Chicken Tracks	2
Chalk Outline	4
Arabidopsis Thaliana 6 RNA – Difficulty Level 0	5
Anemone	3
Adenine	5
7 multiloop	5
[RNA] Repetitious Sequences 8 10	5
Shortie 4	5
Shortie 6	5
Runner_	5
Recoil	5
Quasispecies 2-2 Loop Challenge	4
Prion Pseudoknot – Difficulty Level 0	5
Misfolded Aptamer	4

Puzzle	Number of Times Solved out of 5
[CloudBeta] An Arm and a Leg 1.0	5
[CloudBeta] 5 Adjacent Stack Multi-Branch Loop	5
Worm 1	4
Water Strider	5
Variation of a crop circle	3
U	4
Tripod 5	4
Tilted Russian Cross	5
Thunderbolt	4
This is ACTUALLY Small And Easy 6	5
The turtle(s) move(s)	5
The Sun	5
The Ministry	5
The Gammaretrovirus Signal – Diffuculty Level 0	5
The fractal	5
Taraxacumr officinale	5
Still Life (Sunflower In A Vase)	5
stickshift_	5
square	5
Spiral	5
snoRNA SNORD64	5
Six legd Turtle	5
Six legd turtle 2	5
Simple hairpin	5
Shape Test	3
Saccharomyces Cerevisiae – Difficulty Level 0	5
Pokeball	1
Multilooping 6	3
Multiloop..	3
Misfolded Aptamer 6	2

Table 1

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

Table 2

Description	Raw	Encoded
Base sequence	CCAGAAAAAAAAACUGG	4,4,1,3,1,1,1,1,1,1,1,1,1,4,2,3,3
Natural Structure in dot-bracket notation	((((.....)))) (dot-bracket)	2,2,2,2,1,1,1,1,1,1,1,1,1,1,3,3,3,3
Target Structure in dot-bracket notation	(((((.....))))) (dot-bracket)	2,2,2,2,2,1,1,1,1,1,1,1,1,3,3,3,3,3
Natural Energy	-2.2 kcal/mol	-2.2
Target Energy	4.9 kcal/mol	4.9
Natural Structure in pairmap notation	((((.....)))) (pairmap)	16,15,14,13,-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,-1,4,3,2,1,0
Locked bases	xoooooooooooooox	2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2

Table 3