Functional Generative Design of Mechanisms with Recurrent Neural Networks and Novelty Search

Cameron R. Wolfe
The University of Texas at Austin
wolfe.cameron@utexas.edu

Cem C. Tutum
The University of Texas at Austin
tutum@cs.utexas.edu

Risto Miikkulainen
The University of Texas at Austin
risto@cs.utexas.edu

ABSTRACT

Consumer-grade 3D printers have made the fabrication of aesthetic objects and static assemblies easier, opening the door to automate the design of such objects. However, while static designs are easily produced with 3D printing, functional designs, with moving parts, are more difficult to generate: The search space is high-dimensional, the resolution of the 3D-printed parts is not adequate, and it is difficult to predict the physical behavior of imperfect, 3D-printed mechanisms. An example challenge for automating the design of functional, 3D-printed mechanisms is producing a diverse set of reliable and effective gear mechanisms that could be used after production without extensive post-processing. To meet this challenge, an indirect encoding based on a Recurrent Neural Network (RNN) is proposed and evolved using Novelty Search. The elite solutions of each generation are 3D printed to evaluate their functional performance in a physical test platform. The proposed RNN model successfully discovers sequential design rules that are difficult to discover with other methods. Compared to a direct encoding of gear mechanisms evolved with Genetic Algorithms (GAs), the designs produced by the RNN are geometrically more diverse and functionally more effective, thus forming a promising foundation for the generative design of 3D-printed, functional mechanisms.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Genetic algorithms**; *Shape analysis*; • **Applied computing** → *Computer-aided design*;

KEYWORDS

3D Printing, Functional Generative Design, Gear Mechanism, Recurrent Neural Networks, Novelty Search

ACM Reference Format:
Cameron R. Wolfe, Cem C. Tutum, and Risto Miikkulainen. 2019. Functional Generative Design of Mechanisms with Recurrent Neural Networks and Novelty Search. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3321707.3321785

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321785

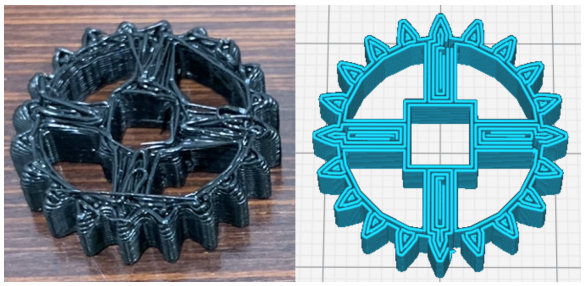

Figure 1: (a) A gear that was fabricated with an FDM 3D printer and (b) the gcode file that was used to prepare the fabrication of the gear. The imprecision of FDM printers on smaller, detailed parts can be observed.

1 INTRODUCTION

Fused Deposition Modeling (FDM), commonly known as 3D Printing, is a practical prototyping technique in which a digital model of a 3D object is sliced, via dedicated software, into thin layers and fabricated by fusing these layers on top of each other until the final product is produced. Its low cost and ease of use has made FDM the standard technology for consumer-grade 3D printers, and its wide adoption has made the fabrication of a variety of objects and assemblies easier than ever. However, despite the rising popularity of FDM 3D printers and wide availability of open source 3D printing designs in popular, online repositories [2, 3], the creation of 3D-printable designs with moving parts is still challenging [6, 15]. The high dimensional search space for the control parameters of designing and 3D printing functional assemblies is the primary reason for such challenges. Additionally, the resolution of 3D-printed objects is greatly limited by the software that prepares the 3D printing instructions for the 3D printer and the nature of the 3D-printing hardware (see Fig. 1) [12].

Previously, the design of functional mechanisms has heavily relied on the use of pre-existing design rules, analytical equations, or user input, thus limiting automation and creativity in the design process. Some recent studies have explored the ability of cognitive models to create functional mechanisms, such as wind-up toys and robots, that are able to accomplish functional tasks. In [5], researchers proposed a semi-automated method for the creation of animated, mechanical characters by non-expert designers using user-provided sketches of the characters' motion. Another interactive design method was developed in [16], which enables users to rearrange an existing mechanism to fit within a desired space. However, this method requires previous experience in mechanical design. In [11], a computational system was developed that constructs compact, internal mechanisms for wind-up toys that

produce user-requested part motions using a database of known mechanism elements.

Recently, some studies have focused on the design and optimization of gear mechanisms in particular. For example, multiobjective genetic algorithms were used together with a set of analytical equations to simultaneously optimize the efficiency and volume of a pair of gears in [9]. A similar approach was performed by [4] for the optimization of a larger system of gears. Additionally, Gologlu and Zeyveli [7] automated the preliminary stages of designing a gear system by using GAs to evolve a set of possible gear mechanisms as a means of support in the design process, wheras Savsani et.al. [10] utilized a particle swarm optimization algorithm to achieve a similar goal. These approaches for automating the design of gear mechanisms all utilize known analytical or emprical equations and design constraints, thus limiting the production of less intuitive, novel gear mechanisms.

On the other hand, recent advances in generative models have shown promising results in the field of generative design. For example, electromechanical robots, comprised of elementary building blocks (rods and joints), were autonomously designed and optimized in a study performed by Lipson and Pollack [8]. In [14], a Generative Adversarial Network (GAN) model was used to learn an efficient representation of a 3D shape data set and generate novel shapes that did not exist within the training set. Although these shapes were not physically fabricated, the 3D-GAN was successful in generating novel, high-resolution designs. Additionally, although the use of generative models has been mostly limited to static designs, Tutum et.al. [13] recently integrated a Variational Autoencoder (VAE) with a surrogate-based optimization method to generate 3D-printable springs, which were then fabricated and physically evaluated.

In this paper, the design of 3D-printable, functional gear mechanisms is automated using an indirect encoding based on a Recurrent Neural Network (RNN) and the Novelty Search criterion, thus laying a foundation for the use of generative models in automating the design of functional mechanisms. During evolution, the elite solutions of each generation are 3D printed and functionally evaluated with a physical test platform. The paper is organized as follows: First, the methodology is introduced, followed by details regarding the implementation of evolution with the Novelty Search criterion. Next, the physical evaluation setup for mechanism testing is described. The quantitative results of gear mechanisms generated by RNNs are then given and compared with those produced by a baseline GA model. Finally, a brief discussion regarding the outcomes of this methodology and future work is presented.

2 METHODOLOGY

The overall methodology (see Fig.2) involves a Recurrent Neural Network (RNN), the Novelty Search algorithm, and a 3D printer to fabricate mechanism designs for functional testing (see Sec. 2.6). Evolution begins with a population of 150 randomly initialized RNNs, each of which encodes a single gear mechanism, and continues for 40 generations. Evolution is driven by Novelty Search, which assigns high fitness to RNNs that produce unique mechanisms. The generative RNN model used in this experiment designs gear mechanisms by sequentially adding gears into an empty system until a full mechanism is produced. The mechanisms with the highest novelty score after each generation are fabricated with a 3D printer and placed into an archive of elite solutions. Each mechanism that is fabricated is evaluated using a physical experiment (see Sec. 2.6) and assigned a distance score based on this physical experiment to better understand its functional capabilities.

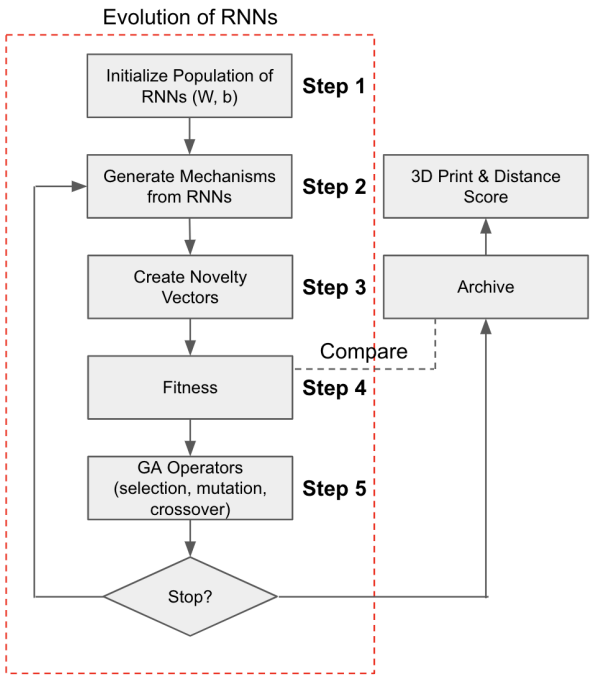

Figure 2: Overall methodology for evolving RNNs that indirectly encode gear mechanisms. Evolution is driven by Novelty Search. The mechanism with highest novelty score is fabricated for physical evaluation during each generation.

Additionally, direct encodings of gear mechanisms are evolved for comparison with the gear mechanisms evolved using generative RNN models. All aspects of the two experiments, aside from the way in which mechanisms are represented, are identical.

2.1 Constraint Handling for Mechanisms

In order to generate valid gear mechanisms, some design constraints must be satisfied, such as limiting the size of the mechanism and avoiding collisions of gears within the mechanism. During evolution, the fitness of infeasible solutions (i.e., those that violate constraints) is assigned to be worse than all feasible solutions and is further penalized by the severity of the constraint violation (e.g., the size of the mechanism is significantly over the limit, several gears are colliding within the mechanism, etc.). Therefore, the evolutionary process favors not only unique mechanisms, but also feasible mechanisms that can be fabricated with a 3D printer.

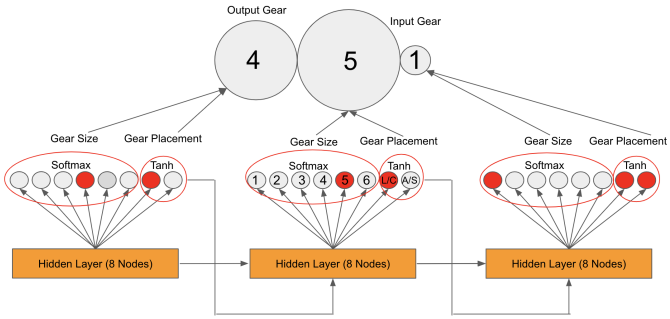

Figure 3: The RNN model encodes a gear mechanism by sequentially placing gears into an empty design area. Red output nodes have high activation values, while grey nodes have low activation values. Labels 1-6 represent gear size, L/C represents linear vs. coaxial placement and A/S represents continue adding gears vs. stop adding gears.

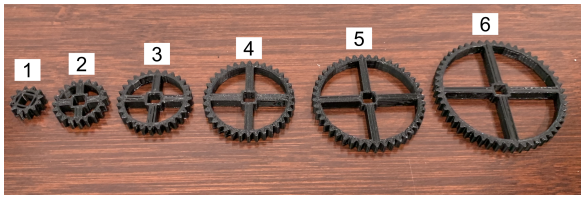

Figure 4: RNN may choose to place one of these six different sizes of gears into the system at every time step.

2.2 Method-I: Generative RNN Model

This section outlines the architecture of the generative RNN model used for the encoding of gear mechanisms. The term RNN refers to a fully-connected neural network that contains a recurrent connection. This recurrent connection connects the hidden layer of the RNN to itself and creates a "loop" inside of the network's architecture that allows the RNN to be activated many times sequentially. These sequential activations, or time steps, each yield a separate output, thus allowing the RNN to output different values over time.

The RNN architecture used in this study, illustrated in Fig. 3, contains one hidden layer and two weight matrices (including bias), which connect the input and previous hidden layer to the hidden layer and the hidden layer to the output layer, respectively. The size of the generative RNN's hidden layer was chosen by qualitatively evaluating the output of RNNs with hidden layers of various sizes, ranging from 1 to 15 hidden nodes. After these network architectures were evaluated based on the diversity of the outputs they produce, a hidden layer of size 8 was determined to yield the most diverse set of outputs. Additionally, the input and output layers each consist of eight nodes.

Within the network, the hidden layer contains a recursive connection, which allows the RNN to be activated over multiple time steps. During each time step, the previous output layer is used as input to the RNN. The hidden layer is obtained by concatenating the input layer with the hidden layer from the previous time step and passing this vector through a weight matrix. The hidden layer is then activated with an element-wise hyperbolic tangent activation function and passed through another weight matrix to yield the output. For the first time step, the previous hidden and output layers, which are both used as input into the RNN's hidden layer, are initialized to vectors of ones because there is no previous hidden or output state. The RNN is executed recursively to individually place each gear into a mechanism until the mechanism is fully constructed, as can be seen in Fig. 3. At each time step of the RNN, a single gear is added into the mechanism. This gear may be placed linear (i.e., side-by-side) or coaxial (i.e., along the same center of axis) with respect to the previous gear added into the system by the RNN. A mechanism can contain a maximum of six gears and no fewer than two gears.

At each time step, the first six RNN output nodes are activated with a softmax activation function and represent the size of the gear to be placed into the mechanism. There are a total of six different gear sizes that can be added into a mechanism, as seen in Fig. 4. The first six RNN outputs are chosen to utilize a softmax activation function because a single gear must be placed into the system at each time step, and the size of this gear is represented by the index of the output node having the highest value. Therefore, after softmax is applied to these six outputs, the index of the output node with the highest probability is used to determine the size of the gear to be placed into the system at that time step (e.g., if the 2nd output node has the highest value after softmax is applied, then gear size 2 is chosen). The same gear can be added into a mechanism multiple times.

The other two output nodes are activated with a hyperbolic tangent activation function. These final two outputs determine how to place the current gear into the mechanism with respect to the previous gear (linear vs. coaxial) and whether RNN should continue adding gears into the mechanism, respectively. For these outputs, the hyperbolic tangent activation function is chosen so that thresholds can be easily placed on the output values and used to interpret how the mechanism should be constructed. For the output representing the gear's placement, a value below -0.5 represents a forwards coaxial connection, a value above 0.5 represents a backwards coaxial connection, and any intermediate value represents a linear connection. For the second output, a value above 0.9 causes the RNN to stop inserting gears into the mechanism, while any value below 0.9 allows the RNN to insert another gear into the mechanism, assuming that the maximum of six gears within the mechanism has not yet been reached.

All output nodes, eight in total, are used as input for the following time step. The equations for the activation of the generative RNN model are as follows:

$$\begin{aligned} \boldsymbol{h_t} &= \alpha(\boldsymbol{W}\,\boldsymbol{x_t} + \boldsymbol{R}\boldsymbol{h_{t-1}} + \boldsymbol{b_w}) \\ \boldsymbol{o_t} &= \alpha(\boldsymbol{Z}\boldsymbol{h_t} + \boldsymbol{b_z}) \bigcup \sigma(\boldsymbol{Y}\boldsymbol{h_t} + \boldsymbol{b_y}) \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is the input vector at time t, $\boldsymbol{h}$ is the hidden layer at time t, $\boldsymbol{b}$ is the bias vector, $\boldsymbol{W}$ is the input to hidden layer weight matrix, $\boldsymbol{R}$ is the hidden layer to hidden layer weight matrix, $\boldsymbol{Z}$ and $\boldsymbol{Y}$ are the hidden layer to output layer weight matrix (separated to emphasize the use of two different output activation functions), α

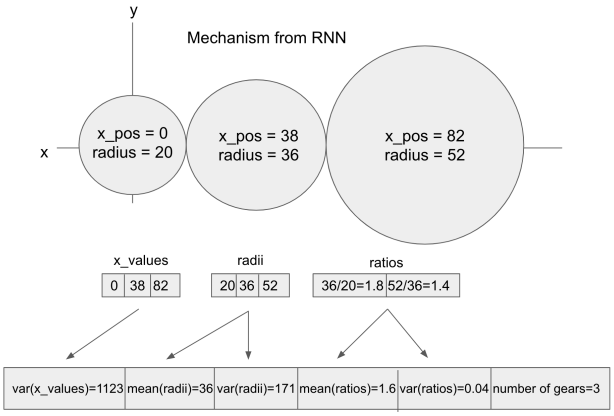

Figure 5: Demonstrates how a novelty vector is created from a gear mechanism.

is the element-wise hyperbolic tangent activation function, and σ is the softmax activation function.

The generative RNN models are evolved for 40 generations using GA and the Novelty Search objective. The details of evolution are outlined in section 2.5.

2.3 Method II: Direct Representation

In order to have a baseline model to compare with generative RNNs, a direct representation of gear mechanisms is created and evolved. This representation is comprised of an ordered list of gears that lists the size of each gear within a mechanism. This list of gears is then combined with a list of placements for each gear with respect to the previous gear. A gear and placement list together encode a single mechanism, which contains between two and six total gears. Similar to RNN evolution, these direct representations are evolved for 40 generations using GA and Novelty Search. Aside from the way in which mechanisms are represented and created, the evolutionary process for each of these methods is identical.

2.4 Novelty Search

Novelty Search is chosen as the objective function: *i*) to allow evolution to perform well in deceptive domains, *ii*) to handle the limited number of fitness evaluations due to the expense of fabricating each mechanism design. The number of feasible solutions that satisfy the various design constraints is low. Therefore, searching for an effective gear mechanism would be more difficult if a specific objective was chosen (e.g., maximizing the distance the car is pulled along the track). Novelty Search discovers innovative designs by aiming to maximize the diversity of mechanisms within the population and minimize the violation of constraints, which allows a more productive region of the large search space to be explored.

In order to assess novelty, a distance metric is created between mechanisms. For every mechanism, a vector, called the novelty vector, can be created that describes the mechanism's structural properties. The values within this novelty vector include various characteristics of the mechanism, such as the ratios between gears (i.e., the quotient of radii between two adjacent gears). The novelty vector for each mechanism includes the following values:

- variance(X): Variance of x-axis position values of the gears (gears are placed from the front to the back of the gear box),
- mean(ratios): Mean of gear ratios (quotient of adjacent gear radii),
- variance(ratios): Variance in gear ratios (quotient of adjacent gear radii),
- mean(radii): Mean of radii for all gears put into the mechanism,
- variance(radii): Variance in radii for all gears put into the mechanism,
- Total Number of Gears

The process of creating a novelty vector is illustrated in Fig. 5. Using this vector, each mechanism's novelty score is determined by finding the minimum Euclidean distance between its novelty vector and all novelty vectors within an archive of vectors from previous generations. After each generation, the vector corresponding to the most novel mechanism in the population is added into the archive. This method of evaluating mechanisms allowed for fitness to be determined without fabricating and testing every possible solution, which is expensive due to the time it takes to 3D print and physically test a mechanism. Instead, only the individual with the maximum novelty score after each generation is fabricated for more detailed physical evaluation, thus allowing innovative, high-performing designs to be discovered without significant testing overhead.

Algorithm 1 Evolution of RNNs

```
procedure EVOLVE(rnn_pop)
    pop_size := 150
    n_gen := 40
    archive := ∅
    pop := random_init_RNN_pop(pop_size) STEP1
    generation := 0
    while generation < n_gen do
        vectors := ∅
        for i from 0 to pop_size do
            output = RNN.forward(pop[i]) STEP2
            vectors.append(get_vector(output)) STEP3
        best_individual := null
        best_novelty := 0
        for i from 0 to pop_size do
            fitness := distance(vectors[i], archive) STEP4
            pop[i].fitness := fitness
            if fitness > best_novelty then
                best_individual := pop[i]
                best_novelty := fitness
        archive.append(best_individual)
        pop := select_tourn(pop)
        pop := mutation_and_crossover(pop) STEP5
        generation := generation + 1
```

2.5 Evolution Process

The evolution process for generative RNNs is illustrated in Fig. 2 and outlined in more detail in Algorithm 1. Evolution begins by randomly initializing a population of 150 generative RNNs, each of which encodes a single mechanism. During every generation, the mechanism encoded by each RNN is generated as described in Sec. 2.2. For every mechanism, the novelty vector is found and used to determine the mechanism's novelty score, which is assigned as the mechanism's fitness. After fitness is assigned using the Novelty Search criterion, tournament selection, with a tournament size of three individuals, is used to select individuals for the next generation and mutation and crossover are applied to the resulting population. These experiments utilize Gaussian weight mutation, where the probability of mutation is 0.8, and uniform crossover, where the probability of crossing over two networks is 0.4. However, crossover is applied separately for first and second layer weight matrices, such that first-layer weight matrices are always crossed over with other first-layer weight matrices and vice versa. The probabilities of mutation and crossover are set relatively high to encourage exploration of the mechanism design space and allow a diverse set of mechanisms to be found and evolved. The evolution process continues for 40 generations.

The novelty score for each mechanism, which is used to assign fitness, is calculated by finding the minimum Euclidean distance between a candidate mechanism's novelty vector and all novelty vectors within the archive of elite solutions from previous generations. However, infeasible mechanisms (i.e., those that violate constraints) are always assigned fitness values lower than that of any feasible mechanism in the current population. For infeasible mechanisms, fitness is obtained by subtracting the magnitude of the mechanism's constraint violation from the lowest feasible novelty score in the population. Each mechanism that is added into the archive of elite solutions is fabricated and evaluated using the physical experiment setup (see Sec. 2.6) to determine its distance score. It should be noted, however, that this distance score is unrelated to the evolution process and is rather used to evaluate the functional effectiveness of the mechanisms produced during evolution.

The process for evolving direct mechanism representations was identical to the evolution of RNNs, aside from the way in which mechanisms were represented.

2.6 Evaluation Setup

The overall physical test setup can be seen in Fig. 6. The rail tracks, car being pulled and gear box are all 3D printed. A rope, which wraps around the car, is connected to the output axle of the gear mechanism, located inside of the gear box. Torque is applied to the input axle of the gear mechanism by a rubber band that is twisted once around the axle. The axle is then released and spun by the rubber band, which causes the rest of the mechanism to be set in motion. The rotation of the output axle then wraps the rope around the pulleys on both ends of the output axle and causes the car to be pulled. The rail track is 35 inches long. The distance that the car travels along the track is recorded for each archived mechanism using the ruler located next to the rail track. The sail that is attached to the car is used to keep the rope out of the track so that it does not affect the car's movement. A video of the working mechanism can be seen at [1].

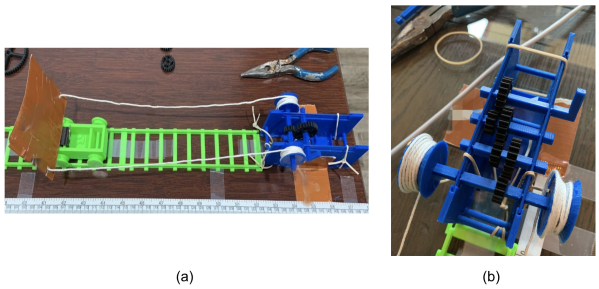

Figure 6: (a) A top-view of the setup for physical experiments. The small car is pulled towards the gear box by the rope attached to the output axle and the traveled distance is recorded. (b) A closer visualization of the pulling mechanism.

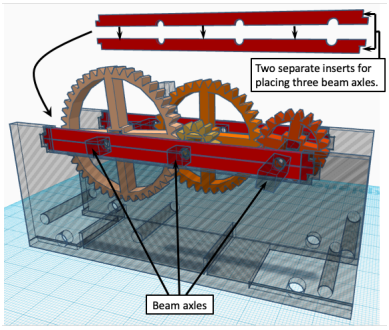

Figure 7: Visualization of the pulling mechanism. The inserts (red parts, used to hold gears in place) are modified and re-printed for every design that is physically tested. All other components (gears, axles, and main body) are reused for each design.

On the other end of the track, the gear box is taped to the surface of the table. This gear box, seen in Fig. 6, was designed to be modular, such that the box and gears do not have to be 3D printed with each evaluation. Instead, only the inserts that hold the gear axles in place are unique to each mechanism (see Fig. 7). Therefore, these inserts are the only parts that must be 3D printed for the physical evaluation of each mechanism. The location of the holes within the inserts that hold the gear mechanism's axles in place are automatically computed for each archived solution such that the gears within the mechanism mesh perfectly. This modular design greatly reduces the time required for testing each mechanism within the archive of novel mechanisms.

Designing mechanisms that perform well in this environment is a difficult task. While a certain mechanism may have high output speed, its torque may not be enough to pull the car along the track because the weight of the car is not negligible. Because of the inverse correlation between the output speed and torque, each mechanism must find an effective balance between these two objectives to pull the car longer distances. In fact, many mechanisms having a high

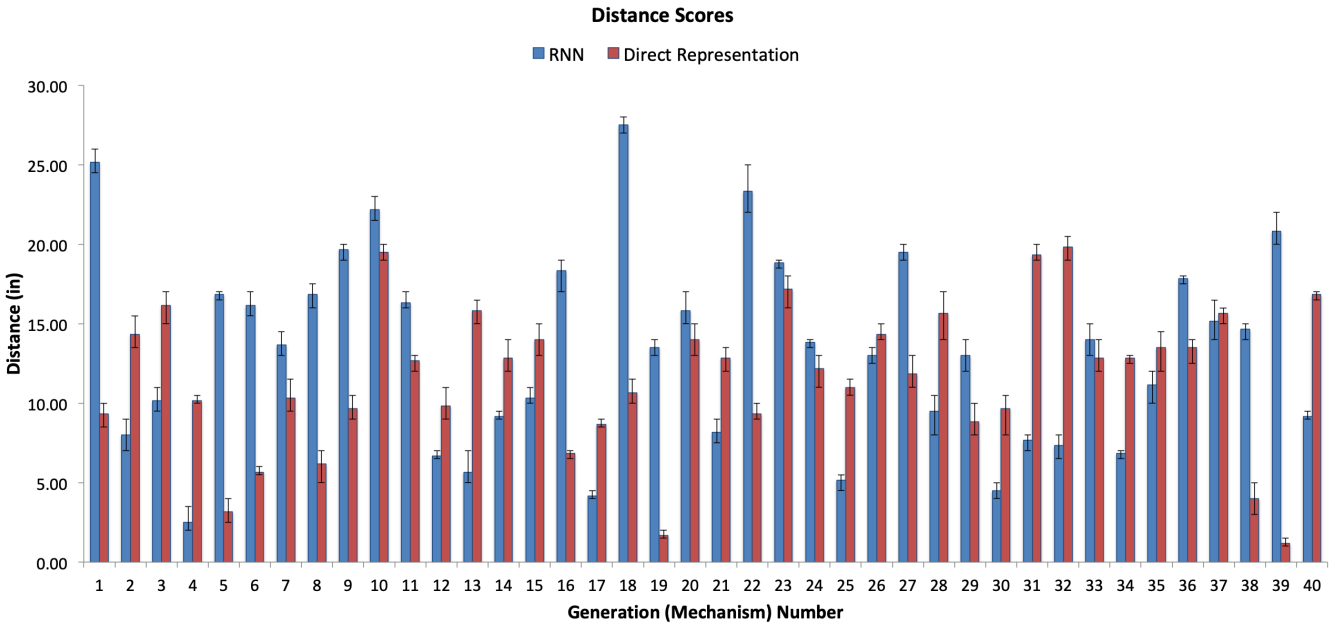

Figure 8: Combined distance score results for all archived mechanisms generated by RNN and Direct Representation experiments. The error bars represent the maximum and minimum distances for the three tests that were performed for each mechanism, while the actual value reflects the average of the three trials.

output speed and low torque, or vice versa, are only able to pull the car a few inches along the track. Because of such challenges, designing a gear mechanism to properly accomplish this task is not trivial.

3 RESULTS

In this section, the results for the evolution of RNN and direct representation models are presented. Two different experiments were performed to test the ability of the proposed generative RNN model and direct encodings to produce a diverse and functionally effective set of mechanisms. The direct representation experiment is used as a baseline to determine the effectiveness of the generative RNN model. For both experiments, at each generation of evolution, the mechanism with the highest novelty score was fabricated and physically tested to determine its distance score. Moreover, each physical test consisted of three separate trials to ensure the consistency of the results. Distance scores were collected separately for RNN and direct representation experiments. The results for both methods are shown in Fig. 8 and discussed in more detail in the next two subsections.

3.1 Results of RNN Evolution

As seen in Fig. 8, the generative RNN model managed to produce a diverse set of mechanisms that pulled the car varying distances. Among all mechanisms generated by RNN, the standard deviation of distance scores was 6.14 inches. The three highest-performing mechanisms, each of which utilized a complex coaxial structure, pulled the car 27.5, 25.2, and 23.3 inches (mechanisms 18, 1, and 22, respectively). The design patterns of these mechanisms can be seen

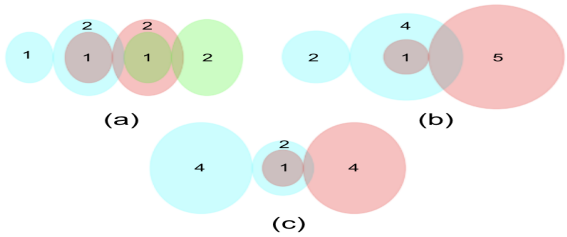

Figure 9: Mechanism 1 (a), 18 (b), and 22 (c) produced by evolution of RNNs. Gears of the same color exist within the same plane and are connected linearly, while gears of different colors are coaxial. The numbers on each gear represent the gear type.

in Fig. 9. Additionally, out of the 40 archived mechanisms, eight of the mechanisms contained coaxial structure. Typically, such coaxial designs are difficult to successfully create because they contain a larger number of gears, which increases the probability of violating design constraints. However, the generative RNN model was able to efficiently place gears into the system by learning repeating, coaxial design patterns that avoided collisions between gears and minimized the overall size of the mechanism.

When examining the mechanism structures in Fig. 9, it is clear that the generated mechanisms follow sequential, repetitive design patterns. For example, in mechanism (a), the pattern of placing a small gear adjacent to a large gear is repeated several times in the mechanism's structure. Such patterns are quite common in mechanisms resulting from the evolution of RNNs. In Fig. 10, the

composition of the RNN network that encodes mechanism (a) from Fig. 9 is displayed. In this figure, it can be observed that the RNN's hidden state follows a distinct pattern while outputting gears at each time step. When a smaller gear is placed into the system, the hidden layer values contain a pattern that causes the next gear outputted by the RNN to be large. Similarly, when a larger gear is placed into the system, the hidden layer values contain a pattern that causes the next gear outputted to be small. This pattern is repeated in every time step of the RNN's activation, thus revealing that the RNN utilizes the history of previously chosen gears to design the mechanism. In this case, it is clear that the RNN, through evolution, has learned a design rule for creating complex mechanisms, which causes it to always pair large gears with small gears. By following this design rule, the RNN creates an output mechanism that has a repeating, coaxial structure, which allows the mechanism, despite being complex, to be packed into a smaller area and satisfy design constraints.

Such patterns were common to many of the RNNs that outputted complex mechanism structures, thus revealing the RNN's ability to learn successful design patterns and use the history of previously chosen gears to produce unique mechanisms. Moreover, such an ability to learn useful design techniques allowed the generative RNN models to avoid constraint violations without compromising the complexity of resulting mechanisms, thus focusing evolution on a more efficient region of the search space. Because RNN is able produce unique mechanisms within the constraints of the design space, such mechanisms achieve relatively high novelty scores, which allow these unique mechanisms to be explored, evolved, and physically tested.

3.2 Results of Direct Representation Evolution

The evolution of direct representations also produced a set of feasible mechanisms. The overall standard deviation of distance scores is 4.6 inches. The three highest-performing mechanisms pulled the car 19.8, 19.5 and 19.3 inches (mechanisms 32, 12, and 31, respectively). These mechanisms are shown in Fig. 11. Out of all 40 archived mechanisms, only three of them utilized coaxial gears. None of these coaxial mechanisms achieved distance scores as high as those discovered by generative RNN models, which indicates that the evolution of direct representations was unsuccessful in finding effective coaxial design patterns. The rest of the archived mechanisms contained no coaxial gears in their structure.

The infeasible mechanisms generated by evolving direct representations of gear mechanisms can be examined to better understand the model's difficulties with evolving complex and effective mechanisms, such as those discovered in the generative RNN experiment. As can be seen in Fig. 12, coaxial design patterns produced by the evolution of direct representations are comprised of random sets of gears, having no distinct design patterns. Because these gears are not packed efficiently into the system, they cannot fit into the available design space and, as a result, violate constraints, thus giving these mechanisms a low novelty score and preventing them from being explored during evolution. Additionally, the direct representation does not utilize the history of previously selected gears in the mechanism or learn any useful design patterns, which means the model must find efficiently-packed, complex mechanisms through random search. Therefore, the direct representation model is less efficient in learning the feasible design space for coaxial mechanisms and struggles to evolve complex structures due to a lack of feasible, complex designs.

4 DISCUSSION AND FUTURE WORK

The proposed generative RNN model is shown to be effective in evolving complex and feasible gear mechanisms despite the limited number of evaluations. Application of various genetic operators (tournament selection, uniform crossover and Gaussian weight mutation) enabled RNNs to find a balance between complexity and lack of constraint violation, thus creating mechanisms that are more diverse and effective than those produced by the direct encoding. Despite the success of the proposed methodology, there are a couple of promising improvements to consider. First of all, automating the generative procedure with a realistic physics simulator would enable a more detailed sampling of mechanisms in the design space and automate the process of physically fabricating and testing mechanisms. However, the use of such simulators with 3D-printable object models is limited by the need for computational resources and the complexity of properly capturing the physical interaction between the parts of a mechanism (e.g., surface friction, nonlinear material properties, etc.). Additionally, the generation of gear mechanisms with continuous design parameters (e.g., radii of gears, size of gear teeth, shape of gear teeth, etc.) with RNN could be explored.

5 CONCLUSIONS

This paper proposes a novel generative RNN model for producing a variety of functional, 3D-printed gear mechanisms. The methodology consists of two main components: an RNN to indirectly encode gear mechanisms and Novelty Search to evolve a diverse set of feasible mechanism designs that can be 3D printed and function well in a physical environment. When compared to a direct encoding of such gear mechanisms, the RNN model managed to produce a more structurally and functionally diverse set of mechanism designs given a limited number of design evaluations. Furthermore, the RNN was able to learn sequential design rules through evolution and was proven to consistently utilize such design rules to create more efficient mechanism structures.

6 ACKNOWLEDGMENTS

The authors wish to acknowledge the funding and support provided by the BEACON Research Center at Michigan State University.

REFERENCES

[1] [n. d.]. The link to the video showing how the gear mechanism works. https://drive.google.com/file/d/1XCMk8Oer9jWAL26kb3PKmFFdGs2vI-id/view?usp=sharing. (Online; accessed 2-February-2019).
[2] [n. d.]. PLA Spring Motor Demonstrator. https://www.thingiverse.com/thing:402412. (Online; accessed 2-February-2019).
[3] [n. d.]. PLA Spring Motor, Rolling Chassis. https://www.thingiverse.com/thing:430050. (Online; accessed 2-February-2019).
[4] Harrison L Bartlett, Brian E Lawson, and Michael Goldfarb. 2018. On the design of power gear trains: Insight regarding number of stages and their respective ratios. *PloS one* 13, 6 (2018), e0198048.
[5] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 83.

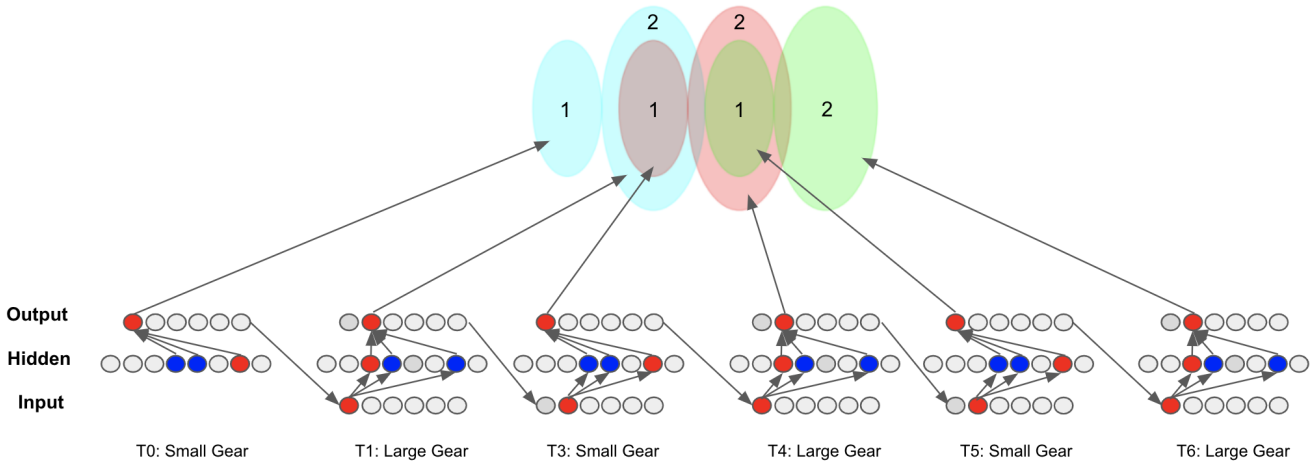

Figure 10: Visualization for design patterns found in the network structure for mechanism 1 from RNN evolution. The RNN memorizes the use of previous gears in the system to create sequential patterns in the output mechanism.

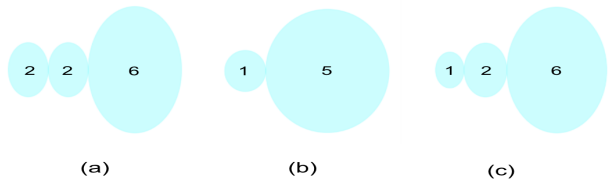

Figure 11: Mechanisms 32 (a), 31 (b), and 10 (c) generated by evolution of direct representations. Gears of the same color are connected linearly.

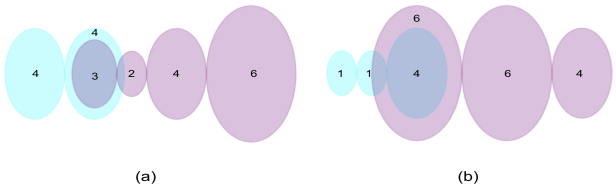

Figure 12: Example of infeasible coaxial mechanisms generated by evolution of direct representations. The randomness and inefficiency of the designs can be observed. The numbers on each gear represent the gear type.

[6] EngineerDog.com. 2017. A Practical Guide to FDM 3D Printing Gears. Retrieved January 25, 2019 from https://engineerdog.com/2017/01/07/a-practical-guide-to-fdm-3d-printing-gears/

[7] Cevdet Gologlu and Metin Zeyveli. 2009. A genetic approach to automate preliminary design of gear drives. *Computers & Industrial Engineering* 57, 3 (2009), 1043–1051.

[8] Hod Lipson and Jordan B Pollack. 2000. Automatic design and manufacture of robotic lifeforms. *Nature* 406, 6799 (2000), 974.

[9] Daniel Miler, Dragan Žeželj, Antonio Lončar, and Krešimir Vučković. 2018. Multi-objective spur gear pair optimization focused on volume and efficiency. *Mechanism and Machine Theory* 125 (2018), 185–195.

[10] V Savsani, RV Rao, and DP Vakharia. 2010. Optimal weight design of a gear train using particle swarm optimization and simulated annealing algorithms. *Mechanism and machine theory* 45, 3 (2010), 531–541.

[11] Peng Song, Xiaofei Wang, Xiao Tang, Chi-Wing Fu, Hongfei Xu, Ligang Liu, and Niloy J Mitra. 2017. Computational design of wind-up toys. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 238.

[12] Y. Tlegenov, K. Telegenov, and A. Shintemirov. 2014. An open-source 3D printed underactuated robotic gripper. In *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. 1–6. https://doi.org/10.1109/MESA.2014.6935605

[13] Cem C. Tutum, Supawit Chockchowwat, Etienne Vouga, and Risto Miikkulainen. 2018. Functional Generative Design: An Evolutionary Approach to 3D-Printing. *CoRR* abs/1804.07284 (2018). arXiv:1804.07284 http://arxiv.org/abs/1804.07284

[14] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. 2016. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS 2016)*. Barcelona, Spain.

[15] Eric A. Yu, Jin Yeom, Cem C. Tutum, Etienne Vouga, and Risto Miikkulainen. 2017. Evolutionary Decomposition for 3D Printing. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*. ACM, New York, NY, USA, 1272–1279. https://doi.org/10.1145/3071178.3071310

[16] Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-aware retargeting of mechanisms to 3D shapes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 81.