Automatically Learning an Evolvable Genotype-Phenotype Mapping*

Matthew Andres Moreno
BEACON Center
mmore500@msu.edu

Wolfgang Banzhaf
BEACON Center
banzhafw@msu.edu

Charles Ofria
BEACON Center
ofria@msu.edu

ABSTRACT

Two schemes for automatic generation of evolvable genotype-phenotype mappings are presented. Both schemes use an artificial neural network (ANN) autoencoder trained on phenotypes harvested from fitness peaks as the basis for a genotype-phenotype mapping. In the first, the decoder segment of a bottlenecked autoencoder is used as the genotype-phenotype mapping. In the second, a denoising autoencoder is used as the genotype-phenotype mapping. Automatic generation of evolvable genotype-phenotype mappings are demonstrated on the n-legged table problem, a toy problem that defines a simple rugged fitness landscape, and the Scrabble string problem, a more complicated problem that serves as a rough model for linear genetic programming. For both problems, the automatically generated genotype-phenotype mappings are found to enhance evolvability.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Matthew Andres Moreno, Wolfgang Banzhaf, and Charles Ofria. 2018. Automatically Learning an Evolvable Genotype-Phenotype Mapping. In *Proceedings of the Genetic and Evolutionary Computation Conference 2018 (GECCO '18).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Evolvability

important [evolvability search] These twin capacities are essential to evolutionary search. Without any heritable variation, evolution would have no raw material to select from and would stagnate. Without any viable variation, evolution would select against all novelty and again stagnate. Hence, evolutionary innovation depends the production of heritable, novel phenotypic variation, some of which must not be severely deleterious.

Indirect genotype-phenotype maps are known to be able to take advantage of regularities of an evolutionary search space to facilitate successful evolutionary search by biasing search towards viable outcomes and allowing large steps in the phenotype space to be made quickly [2]. Such genotype-phenotype maps that facilitate evolutionary search are described as "evolvable." Through generative encodings such as Hyper-NEAT, evolutionary computing researchers have attempted to achieve evolvable genotype-phenotype mapping through direct selection in the genotype space [10]. These approaches are akin to the biological realm, where embryological development is genetically-encoded. This paper proposes a different take on automatic generation of evolvable genotype-phenotype mapping by proposing two techniques to directly learn evolvable genotype-phenotype mappings through artificial neural network autoencoders. It is hoped that this project will yield useful evolutionary computing methodology to achieve automatic design of highly evolvable genotype-phenotype mappings.

Biological organisms exhibit evolvability + dogs + flies

+ define evolvability That is, biological organisms are thought to possess traits that facilitate the evolutionary process. The term evolvability was coined to describe such traits. A general consensus exists in the literature that evolvability stems from traits that facilitate the generation of *novel* heritable phenotypic variation that is *viable* [16]. [introduce evolvability signature]

1.2 Genotype-Phenotype Map

G-P map in biology [12] [1]

1.3 Evolvable Genotype-Phenotype Map

Design genotype-phenotype maps with high latent or innate evolvability.

Considering the other aspect of evolvability, focusing exclusively on bias towards viable variation, another theoretical distinction can be made between innate evolvability, latent evolvability and acquired evolvability. The terms latent evolvability and acquired evolvability were introduced by Reisinger et al. in [14] to discuss canalization, the ability of a population to control the variability generated among its offspring in order to exploit fitness biases specific to its environment.

Distinguishing between innate, latent, and acquired evolvability, it is key to observe that canalization is a âĂIJlearnedâĂİ bias, developed over the course of evolution in response. In the case of *Drosophilia melangoster*, the canalization is due to the lack of symmetry breaking information in the developmental process, which itself is defined by the genome of *Drosophilia*. Thus, the information enabling canalization is stored in the genome. As Reisinger et. al put it, âĂIJevolvability emerges over the course of evolution with a specific fitness function, and is defined within the terms of that

*Produces the permission block, and copyright information

functionâĂİ [14]. To better describe the learned nature of canalization, Reisinger et al. introduce the differentiation between latent evolvability and acquired evolvability. According to Reisinger et al., latent evolvability describes âĂIJthe representationâĂŹs underlying capacity for becoming evolvableâĂİ while acquired evolvability describes âĂIJevolvability learned in response to a particular fitness function.âĂİ In their experiments, acquired evolvability, which can be observed and quantified, is used as a proxy for latent evolvability. I introduce the term innate evolvability to describe bias towards viable variation that is inherent to a representational scheme. For example, Clune et al. identify bias towards phenotypic regularity, which in certain environments tends to be a useful trait, as an inherent trait of indirect genetic encoding [3].

To summarize, latent evolvability describes a representational schemeâĂŹs potential to support canalization. Acquired evolvability describes actual canalization exhibited by an evolving population in response to a particular fitness environment. Innate evolvability refers to nonlearned bias towards viable variation. These distinctions emphasize the fact that bias towards viable variation can arise through canalization, which is a learned trait where learning is enabled by the representational scheme that relates genotype and phenotype, or can result from qualities innate to a representational scheme, such as a bias towards phenotypic regularity.

+ evolvable G-P maps + It is of theoretical interest to study genotype-phenotype maps and their relation to evolvable and practical interest to be able to work with them: more evolvable G-P maps enables more sophisticated digital evolution. + manual design [4, p 223] "artificial evolutionary developmental systems have yet to achieve convincing success on anything beyond quite simple tasks... finding effective developmental recipes for growing complex physical structures and control systems has proven to be an extremely daunting task... Researchers always return to the same question: What aspects of natural development can actually contribute to improving artificial design, rather than being merely scientifically interesting to implement?" + evolve them [13]

1.4 Learning a Genotype-Phenotype Map

I propose a propose a third way: directly learn an evolvable genotype-phenotype mapping using machine learning techniques for unsupervised learning. This idea is inspired by the notion that natural evolution learns to generalize [7] [17]

1.5 Autoencoder Neural Networks

The two proposed genotype-phenotype map learning techniques are based on artificial neural network autoencoders. These are networks that are trained to regurgitate as output the input that they were provided. Such networks are used to discover efficient lower-dimensional codings for datasets and, more recently, as method for generative modeling [6, 9].

The first technique uses a bottlenecked autoencoder. Figure 1 provides a schematic impression of such an autoencoder. This autoencoder has a small layer in the middle that information must pass through to reach the output. Thus, the autoencoder is forced to learn a compact representation for the input it is trained with that can pass through the bottleneck. The part of the autoencoder that precedes the bottleneck is called the encoder and the part that follows is called the decoder. This autoencoder will be trained to encode phenotypes taken from fitness peaks throughout an evolutionary search space. The idea of this approach is that, because the bottleneck provides a compact representation of those high-fitness phenotypes, using the decoder as a genotype-phenotype mapping will readily allow mutation to move the phenotype between otherwise distant fitness peaks.

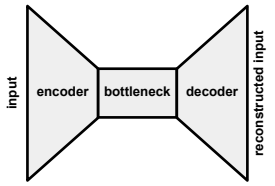

Figure 1: Schematic of a bottlenecked autoencoder.

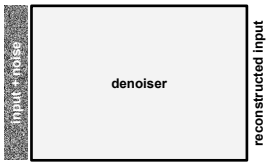

Figure 2: Schematic of a denoising autoencoder.

The second technique uses a denoising autoencoder. Figure 2 provides a schematic depiction of such an autoencoder. These autoencoders have not bottleneck. Instead, they are trained to reconstruct noisy input into its original unadulterated form. This autoencoder will be trained to reconstruct phenotypes taken from fitness peaks throughout an evolutionary search space. For this approach, the entire denoising autoencoder would be used as the genotype-phenotype mapping. The idea of this design is that mutations that would otherwise be deleterious will effectively be interpreted as noise and prevented from being expressed by the genotype-phenotype mapping. Effectively, this mapping should flatten out the valleys between local fitness peaks to allow evolution to more readily drift between those local fitness peaks.

Then [8]

2 TOY PROBLEM DESCRIPTION

A simple problem is used to investigate the genotype-phenotype map generation techniques proposed. This problem, the n-legged table problem, models a table design scenario in which stable tables are highly advantageous. In this scenario, the phenotype of a table is nothing more than a collection of continuous-valued individual leg lengths. All other details of table design are neglected. The stability of a table is assumed to result solely from uniform lengths between table legs. Clearly, as n grows beyond eight or so this toy problem begins to lose a meaningful connection to real world tables. (When was the last time you saw a fifty-legged table?) However, mathematically (and intuitively) the n-legged table problem scales easily. We arbitrarily use $n = 100$ for all experiments.

This toy problem was chosen because it creates an easy-to-characterize rugged fitness landscape. Because unstable tables are disadvantageous, mutations to level tables tend to be deleterious. Thus, evolving between different table heights — i.e. escaping local maxima — becomes a tricky challenge.

The details of specific evaluation criteria, phenotype representations, and genotype representations used for this toy problem can be found in Section 3.

3 METHODS

A pair of experiments was performed to assess the evolvability of the denoising autoencoder and the bottleneck decoder genotype-phenotype maps in relation to the direct genotype-phenotype map. A high-level overview of these experiments is provide immediately below. Methodological details can be found in the subsequent subsections.

First, in evolvability signature experiments the novelty and deleteriousness of mutation was assessed under these three genotype-phenotype maps. Populations of 300 individuals were evolved using a particular map for 50 generations. Then, the individual with the best fitness score over those 50 generations was isolated. Mutant offspring of that individual were assessed for phenotypic novelty in relation to their parent and fitness score. Phenotypic novelty was assessed as the euclidean distance between a parent phenotype $\vec{p}$ and a child phenotype $\vec{x}$. Fitness was calculated according to the same criteria used during the 50 generations of evolution (described below). Forty replicate evolutionary runs were performed for each genotype-phenotype map. For each replicate run, 10,000 mutant offspring of the sampled champion were assessed. The novelty and fitness scores of these mutant offspring (pooled over all 40 replicate runs) was used to generate the evolvability signatures presented in Section 4 [16].

Second, experiments were performed to assess the ability of the three genotype-phenotype maps to facilitate traversal of the evolutionary search space in response to a selective pressure. Specifically, a selective pressure for short (i.e. zero) table height was added. Table height is calculated as the mean leg length of a table. For each map, three replicate populations of 300 individuals were evolved for 5,000 generations. These populations were initialized to have table height of approximately 1,000. Response to selective pressure for short table height was assessed by tracking mean table height of the populations generation-by-generation.

3.1 Bottleneck Autoencoder ANN Architecture and Training

The bottleneck autoencoder was composed of two architectural components: an encoder and a decoder. The encoder consisted of a 100-to-1 fully-connected linear layer with bias. The decoder consisted of a 1-to-100 fully-connected linear layer with bias. Thus, the bottleneck consisted of 1 float value.

The was network was trained for 200 epochs by stochastic gradient descent with parameter settings:

- learning rate: 1^{-3}
- momentum: 0.7
- batch size: 16

Loss was defined as mean square error of the difference between the presented phenotype and output (the reconstructed phenotype).

Training data for the bottleneck encoder was generated from 250 populations of 300 individuals evolved with direct genotype-phenotype map. Initial leg lengths of each separate population were taken from a Gaussian random walk seeded uniformly between 0 and 1000 with $\mu = 0, \sigma = 1.0$ where each 100 consecutive values was taken as the initial leg lengths of a particular individual. This step was performed to ensure that the 250 populations were well-spread throughout the phenotype space. The 250 populations were evolved separately for 50 generations using the operators and settings described for the direct encoding for the evolvability signature experiments. The 7500 phenotypes — vectors of 100 float values — present in the populations after 50 generations of evolution were taken as training data. Leg lengths in the phenotypes were normalized to the range $(0, 1)$ for the training process.

3.2 Denoising Autoencoder ANN Architecture and Training

The denoising autoencoder consisted of a 100-to-100 fully-connected linear layer without bias.

The was network was trained for 2500 epochs by stochastic gradient descent with parameter settings:

- learning rate: 1^{-4}
- momentum: 0.9
- batch size: 2048

Model parameters were initialized uniformly between 0.005 and 0.015. During training, parameters were clamped in the range $(0, 1)$. During the training process, Gaussian noise with $\mu = 0, \sigma = 0.025$ was introduced to the input presented to the autoencoder. Loss was defined as mean square error of the difference between the original phenotype without Gaussian noise and output (the reconstructed phenotype).

The same training data generated for the bottleneck autoencoder was used to train the denoising encoder.

3.3 Genotype-Phenotype Maps

Three genotype-phenotype maps were used: the direct genotype-phenotype map, the bottleneck decoder encoding, and the denoising encoding. In these experiments, the phenotype is 100 floats (representing 100 individual leg lengths). Thus, for all three maps, the phenotype space is $\mathcal{R}^{100}$.

The direct genotype-phenotype map is the identity function so under the direct map, the genotype space is $\mathcal{R}^{100}$.

Under the bottleneck map, just the decoder component of the full bottleneck autoencoder is used as the genotype-phenotype map. The bottleneck autoencoder used in these experiments narrows down to a single node layer, so under this map the genotype space is $\mathcal{R}$.

Under the denoising encoding, the full denoising autoencoder is used. The shape of input to the denoising autoencoder is identical to the shape of the phenotype (a hundred-float vector), so under this map the genotype space is $\mathcal{R}^{100}$.

3.4 Operators

For all experiments, tournament selection with $k = 5$ was used. Two-point crossover was also used in all experiments. (Note, though, that when the bottleneck genotype-phenotype is was employed the genotype is a member of $\mathcal{R}$ so crossover has no effect.) All offspring engaged in crossover with one other individual with a probability 0.5. Mutation was performed by site-wise Gaussian perturbation of the genome. For evolvability signature experiments, mutation of genome elements was $\mu = 0, \sigma = 0.1$ with a per-individual probability of 0.2 and a subsequent per-site probability of 0.01. For response to selection experiments, mutation of genome elements was $\mu = 0, \sigma = 0.1$ with a per-individual probability of 0.2 and a per-site probability of 0.2. (For all experiments with the bottleneck map, a per-site probability of 1 $\sigma = 0.1$ was employed.)

These particular operators were used due to their easy accessibility in DEAP. As these experiments were demonstrative instead of applied, the primary criteria for operators and parameters needed to meet was simply not being obscure enough to distract from the new genotype-phenotype mapping techniques being demonstrated. It would be beneficial to repeat these experiments with other operator and parameter choices popular in the literature and in applied use to investigate the generalizability of those new techniques.

3.5 Evaluation Functions

For evolvability signature experiments, "level-ness" was the sole criterion determining fitness score. For a phenotype $\vec{x}$, the fitness score was computed as

$$-\sigma(\vec{x})$$

where σ represents calculation of standard deviation. Note that in all experiments performed, selection was performed to maximize (not minimize) fitness score. Under this criterion, each level table of a particular height is a local fitness peak because any single-site mutation increases the leg height variance. This fitness criteria was chosen for evolvability signature experiments because of its ruggedness.

For response to selection experiments, the "level-ness" and absolute height of a table were both factored into the fitness score. For a phenotype $\vec{x}$, the fitness score was computed as

$$-\sigma(\vec{x}) - |\mu(\vec{x})/10|$$

where σ represents calculation of standard deviation and μ represents calculation of mean. Under this criterion, a selection pressure for short tables is applied. The global fitness score peak is the table with all legs length 0. However, the phenotypic fitness landscape remains rugged with local peaks occurring as before at level tables. Thus, the ability of genotype-phenotype map to facilitate evolution will be reflected by the ability of a population to escape local fitness peaks and progress towards the global fitness peak.

3.6 Implementation

The evolutionary computing components of this project were implemented using the Distributed Evolutionary Algorithms for Python package, which allows for rapid prototyping and extreme flexibility [5]. The artificial neural network autoencoder components of this project were implemented using PyTorch, a Python-based deep learning framework [11].

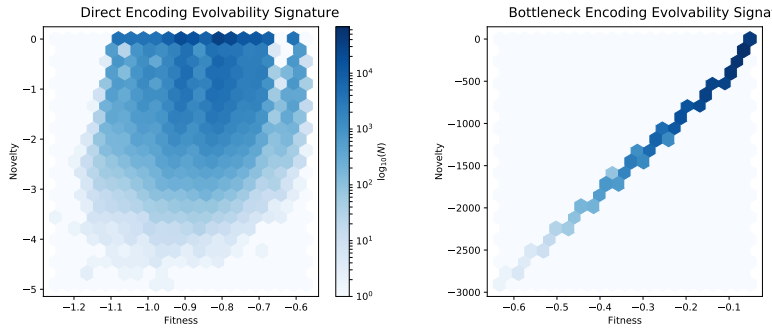

Figure 3: Evolvability signatures for, left to right, direct, bottleneck, and denoising genotype-phenotype maps.

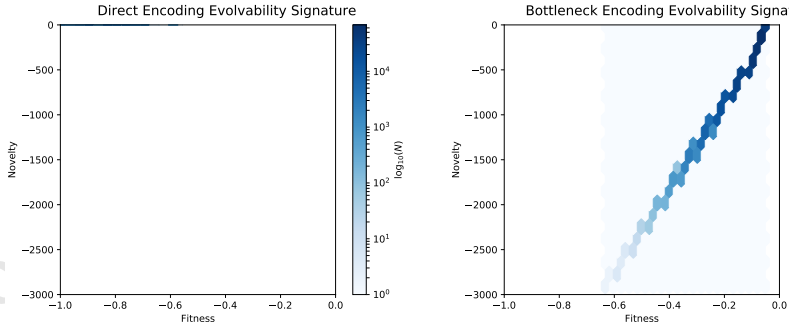

Figure 4: Same-scale evolvability signatures for, left to right, direct bottleneck, and denoising genotype-phenotype maps.

Experiments were run on a PC (no GPU). In total, the experiments reported used approximately 12 hours of compute time.

Code components for this project are available at https://github.com/mmore500/cse-848-project.

4 RESULTS

Assessment of evolvability signatures and observed response to selective pressure towards a global fitness peak shows that both autoencoder-derived genotype-phenotype maps enhance evolvability relative to the direct encoding.

4.1 Evolvability Signatures

Figure 3 provides evolvability signatures for the direct, bottleneck, and denoiser genotype-phenotype maps. These evolvability signatures are heat maps that summarize the outcomes observed under mutation for

As is expected, in all the direct and bottleneck evolvability signatures, offspring fitness tends to decrease somewhat as is expected. Surprisingly, under the denoising mapping this is not true. This suggests that the denoising mapping is better able to generate novelty without losing of fitness or, put another way, that mutation tends to be mildly deleterious.

It must be noted that in Figure 3, the evolvability signature for each genotype-phenotype map are at radically different absolute scales. Figure **??** compares the evolvability signatures of the three

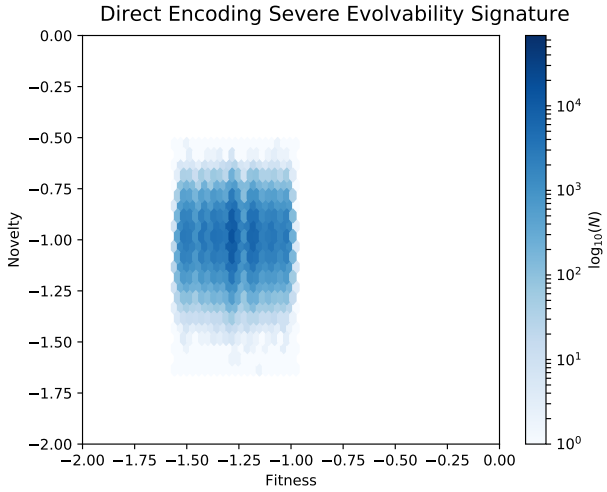

Figure 5: Same-scale severe evolvability signatures for, left to right, direct and denoising genotype-phenotype maps.

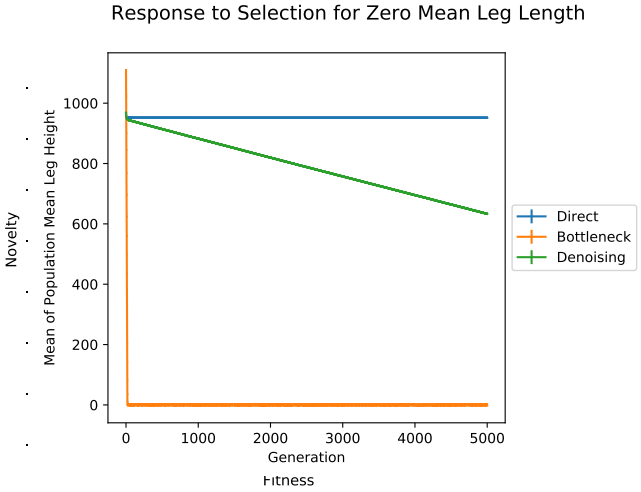

Figure 6: Response to short-table selection pressure under different genotype-phenotype maps.

genotype-phenotype maps at the same absolute scale. It can clearly be seen that the bottleneck mapping can generate much more novelty per mutational step than either of the other mappings. Note that the absolute fitness scores of nearly all offspring under the bottleneck mapping are greater than the absolute fitness of offspring under the direct mapping. The same is true of the denoising mapping.

Because mutation under the direct and denoising mappings tend to have relatively small phenotypic effects, "severe" evolvability signatures were generated to better compare the evolvability of these two mappings. These severe signatures, shown in Figure 5, are generated as before, except instead of applying a single mutation to generate mutant offspring from a parent 100 mutations were applied. Thus, these charts reflect the novelty and fitness outcomes of larger mutational steps in the genotype space. From this comparison, it can be seen that somewhat greater novelty tends to be generated under the direct mapping. However, much better fitness outcomes are observed with the denoising mapping. Again, novelty seems to be generated with little or no fitness cost. Thus, the denoising mapping appears to produce more useful variation than the direct mapping.

4.2 Response to Selection

Figure 6 plots mean table height by generation under selection for both short table height and table stability (described in detail in Section 3). Error bars representing standard deviation between three replicate runs, although so minuscule as not to be easily visible, are provided every 1,000 generations.

Under the direct mapping, a slight decrease in mean table height is observed for a few generations after initialization. However, no further decrease in mean table height was observed over the course of evolutionary runs. These runs ended with a mean table height of approximately 950. Direct-encoded populations were trapped at local fitness peaks and unable to respond to selective pressure for short table height.

Under the bottleneck mapping, a severe decrease in mean table height is observed after initialization. Well within 100 generations, the populations have come close to the global fitness peak — a level table with height 0. Populations with the bottleneck mapping were able to quickly respond to selective pressure for short table height.

Finally, under the denoising mapping, a slight drop-off in mean table height is observed after initialization. Then, a steady decrease in table height was observed for the remainder of the evolutionary runs. These runs ended with a mean table height of approximately 630. Although not as quickly as under the bottleneck mapping, under the denoising mapping populations were still able to respond to selective pressure for short table height.

5 DISCUSSION

Evolvability signature analysis indicate that, in the n-leg table domain, the bottleneck and denoising mapping are both more evolvable than the direct encoding. The bottleneck mapping allows large mutational steps to be taken through the phenotype space relative to the denoising mapping and the direct mapping. Both tend to have mutational outcomes that are superior in fitness to those of the direct encoding. The bottleneck and denoising mappings were observed to free evolution from getting stuck at local fitness peaks, as occurred under the direct mapping, and instead enable progress towards a global fitness peak. Thus, in the n-leg table domain, the proposed genotype-phenotype maps seem to enhance evolvability in both a practical and theoretical sense.

Hand-design of genotype-phenotype maps is common practice in evolutionary computing. These genotype-phenotype mappings, such as NEAT, have been used with good success and studied extensively [15]. In particular, there has been interest in designing dynamic genotype-phenotype mappings that are influenced by the contents of a genome, as occurs in nature, so that an evolvable mapping may themselves be evolved [13]. Nonetheless, existing genotype-phenotype mappings can be domain-specific; a scheme

useful for artificial neuroevolution, for example, likely isn't useful in a linear genetic programming context.

The proposed techniques to generate evolvable genotype-phenotype mappings are in principal generalizable to any context in which many phenotypes at local fitness peaks can be surveyed (i.e. extensive exploratory evolution with a direct encoding isn't computationally prohibitive) and phenotypes can be represented as a continuous-valued, constant-length vector. Using these techniques may reduce the human labor and expertise required to design evolvable genotype-phenotype mappings for new evolutionary computing domains. Further, these autoencoder-based approaches might yield more evolvable genotype-phenotype maps than human design for existing evolutionary computing domains. However, it must be noted that the proposed techniques have only been demonstrated on a toy problem for which it is trivial to manually design an evolvable genotype-phenotype mapping. Further, it is certainly true that for more complex applications, autoencoder design itself typically requires skilled human input. Thus, these automatic genotype-phenotype mappings cannot entirely sidestep the need for manual labor. In fact, in more complex domains, these techniques will introduce a new cost: computation. Training autoencoders in complex domains, especially the process of developing autoencoder design, will require significant computational resources.

Future work with these autoencoder-based genotype-phenotype maps will focus on demonstrating their utility in more complex domains. It will be interesting to research how to adapt autoencoder architecture and training to a domain less well-understood than the n-leg problem. For example, questions such as "What should the size of the bottleneck be for a bottlenecked autoencoder?" or "What type of noise should a denoising autoencoder be trained with?" will need to be addressed. Ultimately, work should apply these techniques to a domain where human-designed genotype-phenotype mappings have significant shortcomings to attempt to surpass the performance of human-designed maps.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Yuhua Li for providing the MATLAB code of the *BEPS* method.

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. The work is supported by the National Natural Science Foundation of China under Grant No.: 61273304 and Young Scientists' Support Program (http://www.nnsf.cn/youngscientists).

REFERENCES

[1] P Alberch. 1991. From genes to phenotype: dynamical systems and evolvability. *Genetica* 84, 1 (1991), 5–11.
[2] Jeff Clune, Benjamin E Beckmann, Charles Ofria, and Robert T Pennock. 2009. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2764–2771.
[3] Jeff Clune, Charles Ofria, and Robert T Pennock. 2008. How a generative encoding fares as problem-regularity decreases. In *International Conference on Parallel Problem Solving from Nature*. Springer, 358–367.
[4] Keith L Downing. 2015. *Intelligence emerging: adaptivity and search in evolving neural systems*. MIT Press.
[5] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research* 13, Jul (2012), 2171–2175.
[6] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[7] Kostas Kouvaris, Jeff Clune, Loizos Kounios, Markus Brede, and Richard A Watson. 2017. How evolution learns to generalise: Using the principles of learning theory to understand the evolution of developmental organisation. *PLoS computational biology* 13, 4 (2017), e1005358.
[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.
[9] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. 2014. Autoencoder for words. *Neurocomputing* 139 (2014), 84–96.
[10] Henok Mengistu, Joel Lehman, and Jeff Clune. 2016. Evolvability search: directly selecting for evolvability in order to study and produce it. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*. ACM, 141–148.
[11] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. PyTorch. (2017).
[12] Massimo Pigliucci. 2010. Genotype–phenotype mapping and the end of the âĂŸgenes as blueprintâĂŹmetaphor. *Philosophical Transactions of the Royal Society B: Biological Sciences* 365, 1540 (2010), 557–566.
[13] Joseph Reisinger and Risto Miikkulainen. 2007. Acquiring evolvability through adaptive representations. (2007), 1045–1052.
[14] Joseph Reisinger, Kenneth O. Stanley, and Risto Miikkulainen. 2005. Towards an Empirical Measure of Evolvability. In *Genetic and Evolutionary Computation Conference (GECCO2005) Workshop Program*. ACM Press, Washington, D.C., 257–264.
[15] Kenneth O Stanley and Risto Miikkulainen. 2002. Efficient evolution of neural network topologies. 2 (2002), 1757–1762.
[16] Danesh Tarapore and Jean-Baptiste Mouret. 2015. Evolvability signatures of generative encodings: beyond standard performance benchmarks. *Information Sciences* 313 (2015), 43–61.
[17] Richard A Watson and Eörs Szathmáry. 2016. How can evolution learn? *Trends in ecology & evolution* 31, 2 (2016), 147–157.