Learning an Evolvable Genotype-Phenotype Mapping

Matthew Andres Moreno
BEACON Center
Michigan State University
mmore500@msu.edu

Wolfgang Banzhaf
BEACON Center
Michigan State University
banzhafw@msu.edu

Charles Ofria
BEACON Center
Michigan State University
ofria@msu.edu

ABSTRACT

We present AutoMap, a pair of methods for automatic generation of evolvable genotype-phenotype mappings. Both use an artificial neural network 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 serves as the genotype-phenotype mapping. In the second, a denoising autoencoder serves 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

• **Computing methodologies → Generative and developmental approaches**; *Genetic algorithms*;

KEYWORDS

genetic algorithms, adaptive representations, indirect encodings, genotype-phenotype map, evolvability, deep learning

ACM Reference Format:
Matthew Andres Moreno, Wolfgang Banzhaf, and Charles Ofria. 2018. Learning an Evolvable Genotype-Phenotype Mapping. In *Proceedings of the Genetic and Evolutionary Computation Conference 2018 (GECCO '18),* Hernan Aguirre and Keiki Takadama (Eds.). ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3205455.3205597

1 INTRODUCTION

Successful evolutionary search depends on the production of meaningful phenotypic variation that can be inherited by offspring. Without useful heritable variation evolution stagnates. The capacity of a population to generate useful heritable phenotypic variation is a key component of evolvability [29]. Different evolving systems can exhibit different degrees of evolvability. Natural systems, in particular, are argued to exhibit greater evolvability than computational systems [31]. Understanding — and replicating — the evolvability of natural evolution is an open problem in computational evolution research [18].

Evolvability is desirable in artificial evolution systems for practical ends — more evolvable systems will help evolutionary algorithms to tackle sophisticated problems more effectively and efficiently [2, 23]. Understanding evolvability is of great scientific interest for both evolutionary biologists and evolutionary computing researchers [18, 21], not only for optimization but also with respect to questions related to the evolution of complexity and open-ended evolution [11, 14].

Indeed, there has been great interest in studying evolvability using computational systems and, in particular, developing techniques to promote evolvability in digital evolution [3, 4, 12, 16, 18, 19, 24, 26, 27]. Inspired by recent theoretical advances applying learning theory to the topic of evolvability [15, 32], we propose a methodology based on autoencoder artificial neural networks that allows evolvable genotype-phenotype encodings to be learned by training on phenotypes harvested from local fitness peaks. We call our approach AutoMap. One variant of AutoMap employs a denoising autoencoder to learn a representation that buffers phenotypes near local fitness peaks against mutation until a mutational threshold is reached where the phenotype shifts to the vicinity of a different local fitness peak. The second AutoMap variant employs a bottlenecked autoencoder to learn a representation where small steps in the genotype space yield significant phenotypic novelty while protecting phenotypic viability. In principle, the AutoMap methodology generalizes across a wide variety of computational evolution domains.

2 BACKGROUND

In this section, we will review evolvability from a biological perspective before shifting to a computational evolution perspective to discuss the relationship between the genotype-phenotype map and evolvability. We will then touch on recently proposed equivalences between evolvability and machine learning. Finally, we will put together a high-level understanding of the artificial neural network autoencoders AutoMap employs.

2.1 Evolvability

Although definitions of evolvable across the literature can be inconsistent, most tie in to the notion of traits that facilitate the generation of *novel* heritable phenotypic variation that is *useful* [29]. Let us examine a pair of biological examples of non-arbitrary phenotypic outcomes under mutation to build our intuition for evolvability.

A developmental constraint against certain non-viable phenotypic variation in *Drosophila melanogaster* was discovered through artificial selection experiments [7, 30]. In these experiments, researchers were able to successfully select for bilaterally symmetric

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205597

phenotypic criteria, such as uniform reduction of eye size, but were unable to successfully select for bilaterally asymmetric phenotypic traits, such as different-sized eyes. Tuinstra et al. hypothesize that the very nature of the developmental process constrains the phenotypic variation that can be observed in offspring, in this case curtailing offspring that lack bilateral symmetry. Specifically, they hypothesize that a lack of bilateral symmetry-breaking information during the embryological development of *Drosophila* explains the negative result of artificial selection for bilaterally asymmetric phenotypic traits. In this way, the distribution of phenotypic diversity in offspring is biased away from (likely not useful) asymmetric variation.

In addition to qualities that constrain against non-viable mutational outcomes, biological organisms can possess qualities that facilitate increased heritable variation for a phenotypic trait. Somatotropin, also known as growth hormone, has widespread anabolic effects on tissues throughout the body. Mutations that affect somatotropin regulation, structure, or the response by cells all provide avenues for substantial heritable variation in body size [8]. Dog breeds exhibit a range of body weights spanning nearly an order of magnitude. Indeed, among certain groups of dogs, much of this variation can be explained by just six genes, several of which are associated with pathways somatotropin participates in [25]. The presence of such hormonal signaling pathways can be viewed as making a broad range of heritable phenotypic variation more readily realizable via mutation.

2.2 Genotype-Phenotype Map and Evolvability

In biological science, phenotype refers to an organism's observable characteristics (morphological, behavioral, physiological, chemical, etc.) that govern its interaction with the environment and ultimately determine its fitness. Genotype refers to the heritable information that shapes the phenotype displayed by the individual, i.e., the organism's DNA. Development is the process through which an organism's genotype and environment interact to determine its phenotype. It can be useful to abstract development as a mathematical function that takes genetic information as its input and outputs phenotypic characteristics. This mathematical function representing development is referred to as the genotype-phenotype map [1].

The nature of the genotype-phenotype map employed in an evolving system influences that system's evolvability [22]. It is of theoretical interest to study genotype-phenotype maps and their relation to evolvability. In computational evolution, it is also of practical interest to implement evolvable genotype-phenotype maps: more evolvable genotype-phenotype mappings appear to enable more sophisticated digital evolution. Let us discuss three theoretical constructs that are useful to understanding the relationship between the genotype-phenotype map and evolvability: latent evolvability, acquired evolvability, and innate evolvability.

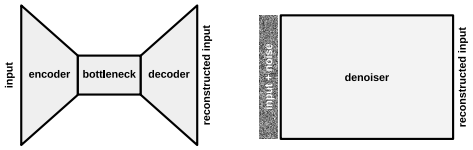
The terms latent evolvability and acquired evolvability were introduced in [24] to discuss canalization, the ability of a population to bias the types of phenotypic variability generated among its offspring in order to exploit fitness biases specific to its environment. Reisinger and Miikkulainen argue that canalization is a "learned" bias, developed over the course of evolution in response to selection pressure in a particular environment [24]. Latent evolvability describes a genotype-phenotype map's potential to exhibit canalization while acquired evolvability describes actual canalization exhibited by an evolving population in response to a particular fitness environment. We introduce the term innate evolvability to describe bias towards viable variation that is inherent to a genotype-phenotype map. For example, Clune et al. identify bias towards phenotypic regularity, which in certain environments tends to be a useful trait, as an inherent quality of indirect genetic encoding [5].

Innate, latent, and acquired evolvability each represent an opportunity for intervention by digital evolution practitioners in pursuit of evolvability. Researchers can manually design an architecture for an evolving system (often inspired by the developmental processes in nature) that exhibits strong innate [6] or latent [24] evolvability. Indeed, hand-design of genotype-phenotype maps is ubiquitous in evolutionary computing [31]. Researchers have built genotype-phenotype mappings that exhibit latent [23] and innate [6, 28] evolvability. Such hand-designed genotype-phenotype mappings have been used to solve complex problems [3, 34]. Unfortunately, existing manually-designed genotype-phenotype mappings tend to be domain-specific — a scheme useful for artificial neuroevolution, for example, is not likely to be useful for linear genetic programming. In addition, sophisticated genotype-phenotype mappings — particularly those mimicking low-level aspects of embryological development — can prove difficult to design and implement [9, p 223].

Selection techniques like evolvability selection [18], modularly varying environments [12], the provision of a large number of niches for different tasks [19], etc. can be employed to steer evolving populations towards regions of the genotype space that bestow acquired evolvability. Although these approaches have been demonstrated successfully, they are inherently limited by the latent evolvability of the genotype-phenotype mapping with which they are employed [24].

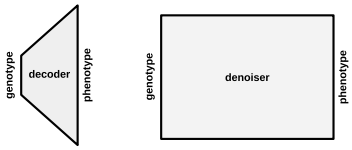
2.3 Evolution and Learning

In this article, we propose a new approach to achieve evolvability: directly learning an evolvable genotype-phenotype map from a training set of phenotypes harvested from local fitness peaks. This idea is inspired by recent work framing evolution and evolvability in the context of learning theory [15, 32]. These authors point out that phenotypic variation generated under mutation and recombination is fundamentally shaped by an evolutionary history. By this observation, they suggest an analogy between evolution and a machine learning system that exploits past experience to make informed decisions. The key point is that undirected genetic variation can lead to structured phenotypic variation under a genotype-phenotype map that is shaped by evolutionary history [32]. Kourvais et al. specifically suggest that evolution might learn to generalize from past experience, essentially learning the structural regularities of phenotypes that were successful in the past and generating new phenotypes that are variations on that theme [15]. Our proposed AutoMap approach makes this hypothesized learning process explicit.



(a) bottleneck autoencoder (b) denoising autoencoder

Figure 1: Schematics of bottlenecked a denoising autoencoders.



(a) bottleneck map (b) denoiser map

Figure 2: Schematics of genotype-phenotype maps constructed with a bottlenecked autoencoder and a denoising autoencoder.

2.4 Autoencoder Neural Networks

We propose to learn evolvable genotype-phenotype mappings via autoencoder neural networks. Autoencoders are artificial neural 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 a method for generative modeling [13, 17]. We will use two specific types of autoencoders: the bottlenecked autoencoder and the denoising autoencoder.

Figure 1a provides a schematic depiction of a bottleneck 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 inputs it is trained with. The part of the autoencoder that precedes the bottleneck is called the encoder and the part that follows is called the decoder.

Figure 1b provides a schematic depiction of a denoising autoencoder. These autoencoders are trained to take noisy input and, from that noisy input, recover a signal in its original unadulterated form.

Both the bottleneck and denoising autoencoders are unsupervised learning models: they do not require labeled training data. Instead, these autoencoders learn to exploit structure in the unlabeled training data in order to perform compression or reconstruction of input.

3 METHODS

3.1 Learning an Evolvable Genotype-Phenotype Mapping

The pair of proposed techniques to automatically learn evolvable genotype-phenotype mappings are based on artificial neural network autoencoders trained to encode phenotypes taken from local fitness peaks spread throughout an evolutionary search space. These phenotypes are gathered by evolving a large number of replicate direct-encoded populations and retaining champion individuals from each population.

The first approach, the bottleneck map, uses just the decoder portion of the bottleneck autoencoder. The decoder serves as the genotype-phenotype map so the genotype is now in the bottleneck vector space while the phenotype remains in the same vector space as before. 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 2a illustrates how the decoder component of the bottleneck encoding is employed to define a genotype-phenotype mapping.

The second approach, the denoiser map, employs the entire denoising autoencoder as the genotype-phenotype mapping. Note that the genotype and phenotype remain in the same, equivalent vector spaces. The idea of this design is that mutations that would otherwise be deleterious will 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 in order to allow evolution to more readily drift between peaks. Figure 2b illustrates how the denoiser autoencoder is used to define a genotype-phenotype mapping.

3.2 *n*-legged Table Problem

This simple problem is used to present a proof-of-concept for the proposed AutoMap techniques. The n-legged table problem models a table design scenario. In this problem, the phenotype of a table is abstracted to a collection of continuous-valued individual leg lengths. All other details of table design are ignored. Stability is considered a highly-advantageous trait for a table. Fitness is based on the stability of a table, which is assumed to result solely from uniformity of table leg lengths. Clearly, as n grows beyond four 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 in this domain.

We chose this toy problem because it induces a straightforward rugged fitness landscape. Because unstable tables are disadvantageous, mutations to tables with uniform leg lengths tend to be deleterious. Thus, evolving between different table heights — i.e., escaping local maxima — is a tricky challenge.

For evolvability-signature experiments, evenness of leg lengths was the sole criterion for fitness in order to isolate the ability of a genotype-phenotype mapping to fulfill this constraint. 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, 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.

For response-to-selection experiments, the evenness of leg lengths 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 zero. However, the phenotypic fitness landscape remains rugged with local peaks occurring as before at level tables. Thus, the ability of a 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.

3.3 Scrabble String Problem

The Scrabble string problem provides a more challenging testbed for the AutoMap approach. In this problem domain, phenotypes are 100-character strings consisting of the letters "a" through "z" and the " " (space) character. Fitness is determined as the count of letter characters contained in valid Scrabble words within the 100-character string. First and foremost, this problem was chosen because phenotypes — and potential indirect genotype-phenotype maps — were thought likely to be easily human-understandable. This problem was chosen as a rough stand-in for linear genetic programming. Finally, working in this problem domain allowed us to leverage existing deep learning work with character-level representations of English language sentences [33]. Like the n-legged table problem, the Scrabble string problem presents a rugged fitness landscape. Many changes to components to the phenotype that are part of a valid scrabble word will invalidate the word, sharply reducing the string's count of letter characters contained in valid Scrabble words. Thus, many mutations will have severely deleterious consequences. Because the Scrabble string problem is much less regular than the n-legged table problem, predicting which types of variation will be severely deleterious in this domain is a more challenging task.

3.4 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 [10]. The artificial neural network autoencoder components of this project were implemented using PyTorch, a Python-based deep learning framework [20]. The software used to perform and analyze our experiments, our figures, and data from our experiments are available via the Open Science Framework at https://osf.io/n92c7/.

3.4.1 n-legged Table Problem. For the n-legged table problem, the denoising autoencoder consisted of a 100-to-100 fully-connected linear layer without bias. The network was trained for 2500 epochs by stochastic gradient descent with learning rate: 10^{-4}, momentum 0.9, and 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 the reconstructed phenotype.

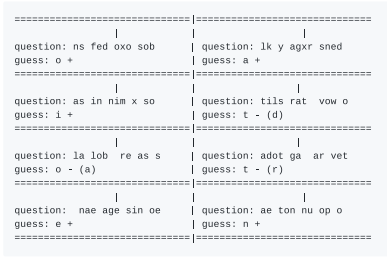


Figure 3: Input/output examples for the denoiser autencoder in the Scrabble domain.

For the n-legged table problem, the bottleneck autoencoder consisted of a 100-to-1 fully-connected linear layer with bias (encoder) and a 1-to-100 fully-connected linear layer with bias (decoder). Thus, the bottleneck consisted of 1 float value. The network was trained for 200 epochs by stochastic gradient descent with learning rate 10^{-3}, momentum 0.7, and batch size 16. Loss was defined as mean square error of the difference between the presented phenotype and the reconstructed phenotype.

Training data for both encoders used for the n-legged table problem consisted of 250 populations of 300 individuals evolved with the 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 set of 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 length values were normalized to the range $(0, 1)$ for the training process. Both encoders were easily trained on a PC (no GPU).

For all n-legged table experiments, population size 300 and tournament selection with $k = 5$ was used. With probability 0.5, offspring engaged in two-point crossover with one other individual. (Note, though, that when the bottleneck genotype-phenotype is employed the genotype is a single float value so crossover has no effect.) 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 subsequent a per-site probability of 0.2. (For all experiments with the bottleneck map, a per-site probability of 1 was employed.) These particular operators were used due to their easy accessibility in DEAP. It would be beneficial to repeat these experiments with other common operator and parameter choices to investigate the generalizability of the AutoMap approach.

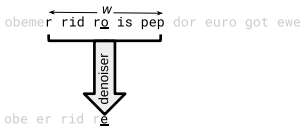


Figure 4: Illustration of denoising autoencoder in action in the Scrabble domain.

3.4.2 Scrabble String Problem. For the Scrabble string problem, the denoising autoencoder consisted of: a 9,000-channel 1-dimensional convolution layer with kernel size 3, a ReLU layer, a 100-channel 1-dimensional convolution layer with kernel size 5, a ReLU layer, a 1500-to-1500 fully-connected linear layer with bias, a Tanh layer, and a 1500-to-27 fully-connected linear layer with bias. Initial model weights were drawn from the standard normal distribution. To generate training data, 20,000 direct-encoded populations of 50 100-character strings were evolved for 40,000 generations. During training, inputs to the autoencoder were 15-character substrings of champion phenotypes with each character encoded as a one-hot vector. With probability 0.25, the middle character of the presented substring was redrawn. Over the course of two days, the network was trained for 4 epochs on a GPU-accelerated machine with the Adam optimizer and batch size 256. Loss was defined as mean square error of the difference between the true identity of the input substring's middle character (before any scrambling) and the predicted identity of that character. At the conclusion of training, the denoising autoencoder predicted the correct identity of the substring's middle character at a rate of approximately 85% on testing data. Example input/output of the denoising autoencoder during training is shown in Figure 3. For the Scrabble string problem, each site of the 100-character genotype was fed through the denoiser, yielding a new 100-character string that consisted of the denoiser's prediction at each site. Figure 4 illustrates this process. The denoiser genotype-phenotype mapping was achieved by performing four of these denoising passes on the genotype.

For all Scrabble string experiments, tournament selection with $k = 3$ was used. No crossover was performed. Mutations scrambled a single, randomly-chosen character in the genotype. We applied a per-individual mutation rate of 0.2 and a subsequent per-site mutation rate of 0.1 while evolving training data for the denoiser. For all other experiments, we used a per-individual mutation rate of 0.33 and a subsequent per-site mutation rate of 0.01.

3.5 Evolvability Signature

Tarapore et al. introduced an evolvability measure that enables simultaneous inspection of both major aspects of evolvability — viability and novelty of phenotypic outcomes under mutation [29]. They forgo use of a scalar metric to describe evolvability, instead reporting evolvability using what they term a "signature." Essentially, the signature is a two-dimensional heatmap presenting the changes in phenotypic form and fitness observed in individual offspring from a single parent. For a highly evolvable individual, we would expect to see offspring occurring with significant frequency in the corner of the heatmap indicating significant change in phenotypic form with slight or no loss of fitness. The evolvability signature provides a nuanced snapshot of evolvability, allowing for interaction between the two primary components of evolvability to be visualized. Such information can be diagnostic, for example alerting researchers to phenomena that might appear falsely promising using other metrics, such as genetic changes that alter phenotypic form substantially but at great cost to fitness or genetic changes that are beneficial to fitness but fail to uncover novel phenotypic form. In these diagrams, fitness increases left-to-right and novelty increases top-to-bottom. Thus, the parental phenotype tends be placed in the upper right of the diagram. (With the occurrence of mutations that benefit fitness, the parental phenotype ends up placed closer to the upper middle of the diagram.) Likewise, mutant phenotypes that are both novel and viable are placed in the lower right of the diagram.

4 RESULTS AND DISCUSSION

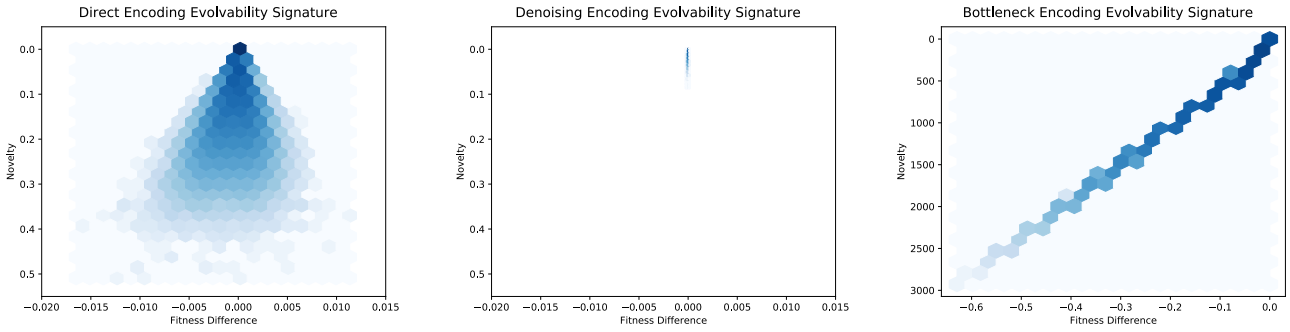
4.1 *n*-legged Table Problem

We used evolvability signature analysis to characterize the evolvability of the *n*-legged table direct, bottleneck, and denoiser encodings (Figure 5). The bottleneck mapping clearly generates more novelty per mutational step than either of the other mappings. Fitness does decline in tandem with novelty, but the absolute fitness scores of all mutant offspring under the bottleneck mapping are greater than the absolute fitness of the direct-evolved champions. Relatedly, although beneficial mutational outcomes are observed frequently under the direct encoding, this is likely affected by relatively low absolute fitness of -0.82 ($s = 0.1$) for direct-evolved champions (in comparison, for instance, to absolute fitness of -0.025, $s = 0.004$ for denoiser-evolved champions). Finally, more novelty is produced by mutation under the direct encoding compared to the denoising encoding. The denoiser's action filtering out certain phenotypic variation likely contributes to the reduction of novelty observed under mutation.

Subsequently, we assessed the ability of the three genotype-phenotype maps to facilitate traversal of the evolutionary search space in response to a selective pressure. Specifically, a selection pressure for short table height was added. Table height is calculated as the mean leg length of a table. For each map, we evolved three replicate populations of 300 individuals for 5,000 generations. We initialized these populations to have table height of approximately 1,000. Response to selection pressure for short table height was assessed by tracking mean table height of the populations generation-by-generation. Figure 6 plots mean table height by generation under selection for both zero table height and table stability. Shaded areas, hugging the table height trajectories so tight as to be difficult to discern, represent bootstrapped 95% confidence intervals ($n = 3$).

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 swift decrease in mean table height is observed after initialization. Well within 100 generations,



(a) direct map **(b) denoising map** **(c) bottleneck map**

Figure 5: Evolvability signatures for three genotype-phenotype maps in the n-legged table problem domain. Note that subfigure 5c is presented with different axis scaling than subfigures 5a and 5b.

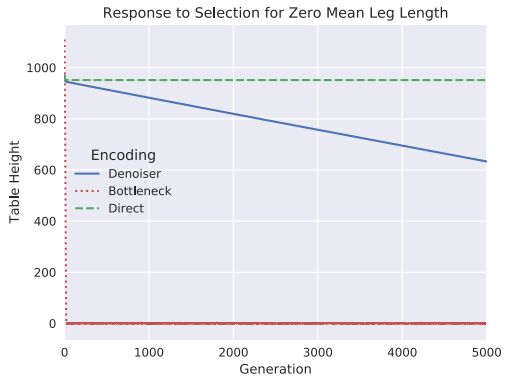


Figure 6: Response to short-table selection pressure under different genotype-phenotype maps.

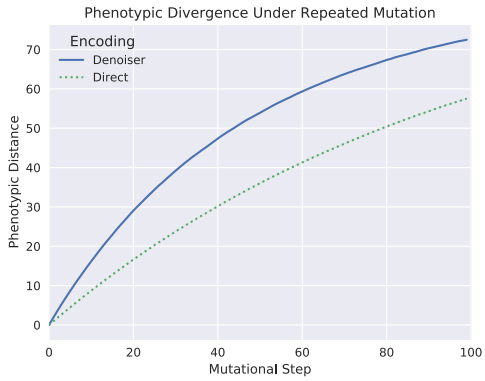


Figure 7: Phenotypic divergence by mutational step in the Scrabble string domain.

the populations approached to the global fitness peak of a perfectly level table with height zero. Populations with the bottleneck mapping were able to quickly respond to selective pressure for short table height.

Under the denoising mapping, a slight drop-off in mean table height occurs after initialization, followed by a steady decrease in table height over the remainder of the evolutionary time. These runs ended with a mean table height of approximately 630. Although not as swiftly as under the bottleneck mapping, denoiser-encoded populations were still able to respond to selective pressure for short table height.

Cursory inspection of input/output of the decoding component of the bottleneck autoencoder revealed that it had learned to echo the single bottlenecked float in order to uniformly populate the vector of 100 phenotypic leg lengths. Thus, the bottleneck genetic representation essentially described the table height; table leg lengths were set to this height value via the genotype-phenotype map. Evolvability signature analysis suggests that this encoding allows the generation of substantial novelty with mild deleterious consequence. Indeed, as shown in Figure 6, table populations using this encoding are able to rapidly evolve to a global fitness peak.

Similar inspection of the denoising autoencoder revealed that that neural network had essentially learned to calculate each phenotypic leg length output as the average of its inputs. Thus, the denoising encoding took genetic input for what — under the direct encoding — would otherwise be an unstable table and output a stable table nearby in phenotype space. Although this encoding does not enhance the novelty of phenotypic outcomes under mutation relative to the direct encoding, it does curb the deleterious effects of mutation. Thus, as shown in Figure 6, table populations using this encoding are able to make slow and steady progress towards a global fitness peak.

4.2 Scrabble String Problem

Evolvability signature analysis was used to characterize the evolvability of the learned denoiser encoding in the scrabble string. To generate these signatures, 250 champion genotypes were drawn from the direct-encoded populations used to train the denoiser encoding. 100-step mutational walks were then taken from the champion genotype. The phenotypes generated from those genotypes under both the direct and the denoising encoding were compared.

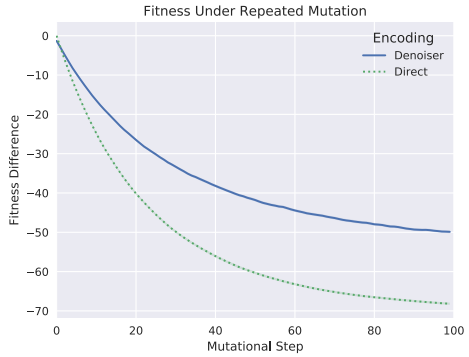


Figure 8: Fitness loss by mutational step in the Scrabble string domain.

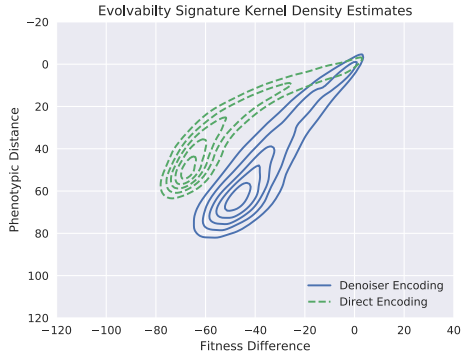


Figure 9: Gaussian kernel density estimates for evolvability signatures of direct and indirect encodings in Scrabble string domain.

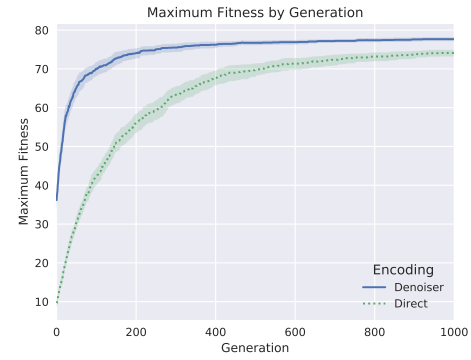


Figure 10: Maximum individual fitness by generation in populations evolving in the Scrabble string domain.

Figure 7 plots novelty (number of non-matching phenotypic characters between the original champion and its mutant offspring) against mutational step (n = 250). Novelty increases more rapidly under the denoiser encoding relative to the direct encoding. Figure 8 plots fitness relative to the original champion against mutational step (n = 250). Fitness declines more slowly under mutation with the denoiser encoding compared to the direct encoding. In both of these figures, bootstrapped 95% confidence are shaded along each curve but are tight enough to be difficult to discern. Figure 9 ties together fitness and novelty information from these random mutational walks in the format of an evolvability signature. Kernel density estimate contours of evolvability signature densities under both encodings are shown side-by-side. This comparison shows that the denoiser encoding simultaneously yields more novelty and less fitness decline under mutation.

Subsequently, we tested the capacity of the denoiser encoding to speed up adaptive evolution. We set up twenty replicate Scrabble string populations to use the denoiser genotype-phenotype map and twenty Scrabble string populations to use the direct map. For each population, maximum fitness was recorded over 1,000 generations of evolution. Figure 10 shows that the populations using the denoiser encoding gain fitness more rapidly than the direct encoding over the first few hundred generations (n = 20). (Again, bootstrapped 95% confidence are shaded along each curve.) Clearly, the evolvability properties of the denoiser encoding translate to evolutionary dynamics that differ strongly from the direct encoding.

As shown in Figures 3 and 4, the denoising autoencoder essentially learns to spell-check a genotype to generate the phenotype. Generally, though not always, characters contributing to valid words are left undisturbed. Interestingly, the change that the denoiser makes to correct a spelling error doesn't always reflect the original state of the string. There are often several possible repairs that will yield valid spelling outcomes. We hypothesize that this dynamic explains the rapid rate of novelty increase under mutation — often, instead of correcting the mutated site in the phenotype corrections are applied to one or several *other* sites. Thus, more than one character in the phenotype may change due to mutation of a single site in the genome. In particular, based on informal analysis of example sentences fed to it, the denoising autoencoder seems to favor fixing spelling mistakes by strategically replacing a letter character with a space. This tendency might occur because the denoiser encountered spaces more often than any other single character during training.

5 CONCLUSION

We introduced AutoMap, a scheme to learn evolvable genotype-phenotype encodings via autoencoder artificial neural networks trained on champion phenotypes from a preliminary evolutionary search. We presented two variants of the AutoMap approach. One is built around a denoising autoencoder that yields a more neutral evolutionary search space through phenotypic robustness under mutation. The other employs a bottleneck autoencoder to yield a more compact representation. Both approaches were demonstrated to increase evolvability relative to the direct encoding in the context of a proof-of-concept test problem, the n-legged table problem. The denoiser approach was further demonstrated to increase evolvability in the more challenging Scrabble string problem.

Much work is left to be done. As immediate next steps, the bottleneck approach should be demonstrated in the Scrabble string

domain and the capacity of both approaches to facilitate evolution towards higher-value fitness peaks should be explored, perhaps by defining high-level criteria to be rewarded in addition to the low-level spelling criteria. In addition, this approach should be put in explicit conversation with recent thinking about parallels between the principles of machine learning and the evolution of evolvability [15, 32].

Indeed, this exploratory work only scratches the surface of the possible applications of the AutoMap approach. Such artificial neural network autoencoders can, in principle, be employed with any phenotype that can be represented as a vector. AutoMap might 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.

Unfortunately, autoencoder design and training itself typically requires skilled human input; AutoMap cannot entirely sidestep the need for manual labor. Indeed, the question of how to adapt autoencoder architecture and training to a less well-understood domain is nontrivial. 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 in any application of AutoMap. It should also be noted that the success of AutoMap in any problem domain depends on the computational capacity to generate large amounts of training data through direct evolution and a willingness to accept the computational cost of performing a forward pass through the autoencoder component for each fitness evaluation when evolving with a learned genotype-phenotype map. Domains where evolution with pre-existing encodings generate poor solutions, repeated cycles of autoencoder training and generation of new training data via evolution might be necessary to yield satisfactory performance. Despite these limitations, we believe that the AutoMap approach to learning evolvable genotype-phenotype maps will prove a useful component of the computational evolution toolbox.

REFERENCES

[1] Pere Alberch. 1991. From Genes to Phenotype: Dynamical Systems and Evolvability. *Genetica* 84, 1 (1991), 5–11.

[2] Peter Bentley and Sanjeev Kumar. 1999. Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. Morgan Kaufmann Publishers Inc., 35–43.

[3] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. 2013. Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 167–174.

[4] Jeff Clune, Benjamin E Beckmann, Robert T Pennock, and Charles Ofria. 2009. HybrID: A Hybridization of Indirect and Direct Encodings for Evolutionary Computation. In *European Conference on Artificial Life*. Springer, 134–141.

[5] 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.

[6] Jeff Clune, Kenneth O Stanley, Robert T Pennock, and Charles Ofria. 2011. On the Performance of Indirect Encoding Across the Continuum of Regularity. *IEEE Transactions on Evolutionary Computation* 15, 3 (2011), 346–367.

[7] Jerry A Coyne. 1987. Lack of Response to Selection for Directional Asymmetry in Drosophila Melanogaster. *Journal of Heredity* 78, 2 (1987), 119–119.

[8] Jesús Devesa, Cristina Almengló, and Pablo Devesa. 2016. Multiple Effects of Growth Hormone in the Body: Is It Really the Hormone for Growth? *Clinical Medicine Insights: Endocrinology and Diabetes* 9 (2016), CMED–S38201.

[9] Keith L Downing. 2015. *Intelligence Emerging: Adaptivity and Search in Evolving Neural Systems*. MIT Press.

[10] 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.

[11] Ting Hu and Wolfgang Banzhaf. 2010. Evolvability and Speed of Evolutionary Algorithms in Light of Recent Developments in Biology. *Journal of Artificial Evolution and Applications* 2010 (2010), 1.

[12] Nadav Kashtan and Uri Alon. 2005. Spontaneous Evolution of Modularity and Network Motifs. *Proceedings of the National Academy of Sciences of the United States of America* 102, 39 (2005), 13773–13778.

[13] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).

[14] Marc Kirschner and John Gerhart. 1998. Evolvability. *Proceedings of the National Academy of Sciences* 95, 15 (1998), 8420–8427.

[15] 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.

[16] Joel Lehman and Kenneth O Stanley. 2013. Evolvability Is Inevitable: Increasing Evolvability Without the Pressure to Adapt. *PloS One* 8, 4 (2013), e62186.

[17] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. 2014. Autoencoder for Words. *Neurocomputing* 139 (2014), 84–96.

[18] 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.

[19] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. 2015. Innovation Engines: Automated Creativity and Improved Stochastic Optimization via Deep Learning. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 959–966.

[20] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. PyTorch. (2017).

[21] Massimo Pigliucci. 2008. Is Evolvability Evolvable? *Nature Reviews Genetics* 9, 1 (2008), 75.

[22] Massimo Pigliucci. 2010. Genotype–Phenotype Mapping and the End of the Genes as Blueprint Metaphor. *Philosophical Transactions of the Royal Society B: Biological Sciences* 365, 1540 (2010), 557–566.

[23] Joseph Reisinger and Risto Miikkulainen. 2007. Acquiring Evolvability Through Adaptive Representations. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. ACM, 1045–1052.

[24] 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.

[25] Maud Rimbault, Holly C Beale, Jeffrey J Schoenebeck, Barbara C Hoopes, Jeremy J Allen, Paul Kilroy-Glynn, Robert K Wayne, Nathan B Sutter, and Elaine A Ostrander. 2013. Derived Variants at Six Genes Explain Nearly Half of Size Reduction in Dog Breeds. *Genome Research* 23, 12 (2013), 1985–1995.

[26] Eric O Scott and Jeffrey K Bassett. 2015. Learning Genetic Representations for Classes of Real-Valued Optimization Problems. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 1075–1082.

[27] Luís F Simões, Dario Izzo, Evert Haasdijk, and Agoston Endre Eiben. 2014. Self-Adaptive Genotype-Phenotype Maps: Neural Networks as a Meta-Representation. In *International Conference on Parallel Problem Solving from Nature*. Springer, 110–119.

[28] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. 2009. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life* 15, 2 (2009), 185–212.

[29] Danesh Tarapore and Jean-Baptiste Mouret. 2015. Evolvability Signatures of Generative Encodings: Beyond Standard Performance Benchmarks. *Information Sciences* 313 (2015), 43–61.

[30] EJ Tuinstra, G De Jong, and W Scharloo. 1990. Lack of Response to Family Selection for Directional Asymmetry in Drosophila Melanogaster: Left and Right Are Not Distinguished in Development. In *Proc. R. Soc. Lond. B*, Vol. 241. The Royal Society, 146–152.

[31] Günter P Wagner and Lee Altenberg. 1996. Perspective: Complex Adaptations and the Evolution of Evolvability. *Evolution* 50, 3 (1996), 967–976.

[32] Richard A Watson and Eörs Szathmáry. 2016. How Can Evolution Learn? *Trends in ecology & evolution* 31, 2 (2016), 147–157.

[33] Tal Weiss. 2016. Deep Spelling. (Mar 2016).

[34] Brian G Woolley and Kenneth O Stanley. 2010. Evolving a Single Scalable Controller for an Octopus Arm with a Variable Number of Segments. In *International Conference on Parallel Problem Solving from Nature*. Springer, 270–279.