Data-efficient Neuroevolution with Kernel-Based Surrogate Models

Adam Gaier, Alexander Asteroth, Jean-Baptiste Mouret

Data-efficient Neuroevolution with Kernel-Based Surrogate Models

Adam Gaier
Bonn-Rhein-Sieg University of Applied Sciences
Inria / CNRS / Université de Lorraine
adam.gaier@h-brs.de

Alexander Asteroth
Bonn-Rhein-Sieg University of Applied Sciences
Sankt Augustin, Germany
alexander.asteroth@h-brs.de

Jean-Baptiste Mouret
Inria / CNRS /
Universitè de Lorraine
Nancy, France
jean-baptiste.mouret@inria.fr

ABSTRACT

Surrogate-assistance approaches have long been used in computationally expensive domains to improve the data-efficiency of optimization algorithms. Neuroevolution, however, has so far resisted the application of these techniques because it requires the surrogate model to make fitness predictions based on variable topologies, instead of a vector of parameters. Our main insight is that we can sidestep this problem by using kernel-based surrogate models, which require only the definition of a distance measure between individuals. Our second insight is that the well-established Neuroevolution of Augmenting Topologies (NEAT) algorithm provides a computationally efficient distance measure between dissimilar networks in the form of "compatibility distance", initially designed to maintain topological diversity. Combining these two ideas, we introduce a surrogate-assisted neuroevolution algorithm that combines NEAT and a surrogate model built using a compatibility distance kernel. We demonstrate the data-efficiency of this new algorithm on the low dimensional cart-pole swing-up problem, as well as the higher dimensional half-cheetah running task. In both tasks the surrogate-assisted variant achieves the same or better results with several times fewer function evaluations as the original NEAT.

KEYWORDS

NEAT; Surrogate Modeling; Neuroevolution

ACM Reference Format:
Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2018. Data-efficient Neuroevolution with Kernel-Based Surrogate Models. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3205455.3205510

1 INTRODUCTION

Neuroevolution (NE), the optimization of neural networks through evolutionary algorithms, has proven to be effective in both machine learning [1, 33] and evolutionary robotics [9, 30], and its flexibility has made it a standard approach for experiments in embodied cognition [18] and open-ended evolution [13, 14]. Recent work has shown that even in deep neural networks, where millions of weights must be optimized, evolutionary techniques are a competitive alternative to gradient descent, demonstrating a surprising ability to scale [22, 34]. With a body of existing NE research, on topics such as exploration and overcoming deception, already being leveraged on deep learning problems [7, 31], NE is poised for a surge in interest.

The main challenge for deploying NE techniques in many applications is that they require many fitness evaluations — too many for most realistic use cases. Deep neural networks, for instance, require long times to train even when large computational resources are brought to bear; in the case of robotics, there is a limit to the amount of interaction that is possible with the physical system.

A common approach to optimization in computationally expensive domains is to use approximate models of the objective function, or surrogate models [2, 8, 10, 12]. These models are created through

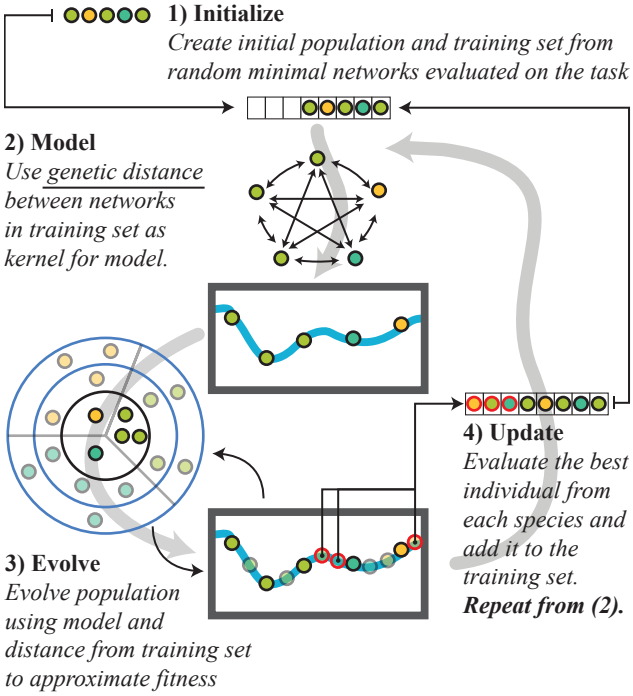

Figure 1: *Surrogate-Assisted NEAT Overview*
Surrogate-Assisted NEAT integrates the compatibility distance, used by NEAT for speciation, into a kernel for Gaussian process regression. This compatibility distance kernel allows for fitness approximation of networks whose topologies vary and grow more complex. Integrating surrogate-assistance techniques into NEAT reduce the number of samples to reach the same performance by several times.

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

an active learning process that aims at selecting points that are both promising in term of fitness and the likelihood to improve the predictions of the surrogate model. The typical loop alternates between selecting the best point to evaluate on the target system and retraining the model to take the new point into account. Machine learning techniques are then used to construct surrogate models which map the genotype space to predicted fitness values [10, 12] .

Creating such a mapping is challenging when evolving neural networks, however: In cases where the topology and weights are both evolved, the dimensionality of the input space is not constant, and the dimensions themselves carry different meanings. Put differently, the surrogate model be able to accept neural networks of varied layouts as an input, rather than a list of parameters.

Our first insight is that kernel-based methods, such as Gaussian process (GP) models and support vector machines, do not require that the inputs all have the same dimensions: they only require that some distance function between samples is defined. Distance measures designed for graphs, such as graph edit distance [23] could theoretically be used to compute the distance between neural networks, but in practice are far too slow, with complexity exponential in the number of vertices. Though approximate measures of graph edit distance have been developed [15, 21], even these are too slow for use as part of every prediction in an optimization algorithm.

Our second insight is that, when evolution is used to produce neural networks, we can glean additional information into the similarity of networks through their heredity. This is already done in the Neuroevolution of Augmenting Topologies (NEAT) algorithm [30], one of the most successful neuroevolution approaches. By tracking genes as they arise in the population, it is possible to create a meaningful and computationally efficient distance measure. NEAT uses this distance to cluster similar networks into species, *here we propose its use as part of a kernel for Gaussian process regression.*

In summary, the primary idea explored in this work is that by tracing the common genes of networks as they evolve we gain a distance measure which can be used in a kernel-based surrogate model. Surrogate-assistance techniques can then be used to create a data-efficient neurevolution algorithm.

Broadly, the surrogate-assisted neuroevolution algorithm presented here proceeds as follows (Figure 1, previous page): (1) a set of minimal networks are evaluated and form the initial training set and population, (2) the distance between all individuals in the training set is computed with a compatibility distance kernel, and a GP model constructed, (3) the population is evolved with NEAT, with the fitness of individuals approximated by the compatibility distance model, (4) the best individuals in each species are evaluated and added to the training set, and the process repeats from (2).

2 RELATED WORK

2.1 Neuroevolution of Augmenting Topologies

Since its introduction in 2002 [30] NEAT has become the standard for neuroevolution. The core features of NEAT focus on overcoming the competing conventions problem of dissimilar networks. The algorithm begins with a population of minimal networks, which grow more complex through mutation. Whenever new nodes and connections are added to the network they are given unique markers. These markers allow common components of dissimilar networks to be identified, providing a basis for crossover and the clustering of networks into species. Species compete against each other for a share of the total offspring they contribute to the next population, and individuals compete within species to provide those offspring.

NEAT has seen successes in domains from video game AI [28] to particle physics [1], and forms the basis and inspiration for a host of other innovations. It is the underlying algorithm for the evolution of compositional pattern producing networks [27] which were in turn applied to the indirect encoding of large scale networks with the HyperNEAT algorithm [29]. The ability of NEAT to produce networks of increasing complexity has also made it an ideal tool for exploring open-ended evolution and novelty-based search [13, 14].

2.2 Gaussian Process Models

Surrogate models can be constructed with a variety of machine learning techniques [10, 12], but GP models [20] are most commonly used in modern approaches. GP models are accurate even with small data sets, and include a measure of uncertainty in their predictions, important for balancing exploration and exploitation.

Gaussian process models use a generalization of the Gaussian distribution: where a Gaussian distribution describes random variables, defined by mean and variance, a Gaussian process describes a random distribution of functions, defined by a mean function μ, and covariance function k.

$$f(x) \sim GP(\mu(x), k(x, x')) \tag{1}$$

GP models are based on assumptions of smoothness and locality: the intuition that similar individuals will have similar behavior. A covariance function k defines this relationship precisely in the form of a kernel. A common choice of kernel is the squared exponential function: as points x become closer in input space they become exponentially correlated in output space:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{2}$$

Given a set of observations $D = (x_{1:t}, f_{1:t})$ where $f_{1:t} = f(x_{1:t})$, we can build a matrix of covariances. In the simple noise-free case we can then construct the kernel matrix:

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \cdots & k(x_t, x_t) \end{bmatrix} \tag{3}$$

When considering a new point (x_{t+1}) we can derive the value ($f_{t+1} = f(x_{t+1})$) from the normal distribution:

$$P(f_{t+1}|D_{1:t}, x_{t+1}) = \mathcal{N}\left(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1})\right) \tag{4}$$

where:

$$\mu_t(x_{t+1}) = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{1:t} \tag{5}$$

$$\sigma_t^2(x_{t+1}) = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} \tag{6}$$

gives us the predicted mean and variance for a normal distribution at the new point x_{t+1}. When the objective function is evaluated at this point, we add it to our set of observations D, reducing the variance at x_{t+1} and at other points near x_{t+1}.

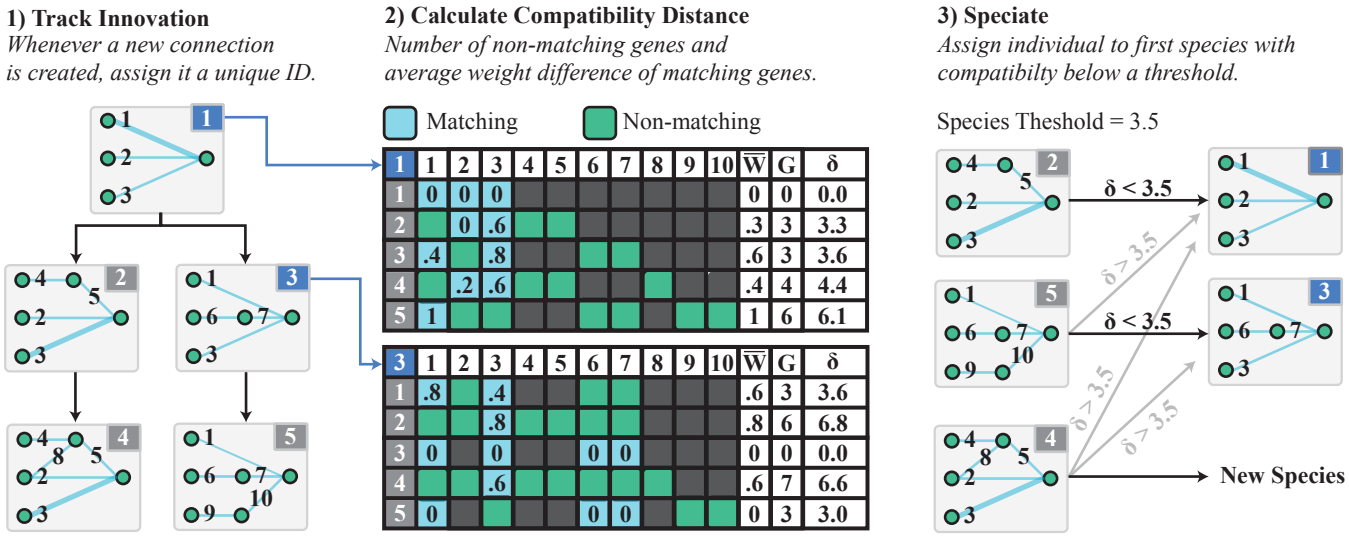

Figure 2: *Calculating Compatibility Distance in NEAT*
By tracking genes as they arise NEAT allows for the efficient comparison of dissimilar networks. The "compatibility distance" between two networks is a weighted sum of the number of genes they do not share and the weight differences between the genes they do. Each species has a representative. Individuals are compared to these representatives, and are assigned to the first species whose representatives compatibility distance is below a threshold. If none exist the individual forms a new species.

Bayesian Optimization. Modern surrogate-assisted optimization often takes place within the framework of Bayesian optimization (BO) [2, 4, 8, 11, 17, 25]. BO approaches the problem of optimization not only as one of finding the most optimal solution, but of modeling the underlying objective function in high performing regions.

BO requires a probabilistic model of the objective function, and so GP models are typically employed. This model is used to define an acquisition function, which describes the utility of sampling a given point. The objective function is evaluated at the point with maximal utility and added to the set of observations. The updated observation set is used to rebuild the model, and the process repeats.

In this work, we use the *upper confidence bound* (UCB) acquisition function [26]. A high mean (μ) and large uncertainty (σ) are both favored, with relative emphasis tuned by the parameter κ:

$$UCB(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}) \tag{7}$$

UCB performs competitively with more complex acquisition functions such as Expected Improvement (EI) and Probability of Improvement (PI) [2, 4].

3 COMPATIBILITY DISTANCE KERNEL GP

We would like to use GP models to approximate the fitness function for a surrogate-assisted neuroevolution algorithm. Though the estimates of GPs are typically based on distance between samples in parameter space, they are a kernel-based method, and as such only require some distance measure between samples.

In the case of neuroevolution, where the topology of the network is evolved along with the values of the weights, we do not have a static or consistent parameter space. As the population of networks grow and change, the genotype spaces they exist in diverge into varied dimensions with inconsistent meanings, leaving standard distance measures such as Euclidean distance unusable.

If a meaningful distance measure was found, then GP models could be used. Neural networks are a class of directed graph, and there already exist measures to compare graphs, such as graph edit distance [23]. Unfortunately, even approximate graph edit distances are too expensive to compute for every prediction [15, 21].

As we are producing the networks through an evolutionary process, however, we can track the phylogenetic links between individuals and compute a distance between them based on their common genes. NEAT introduces just such a mechanism by assigning *innovation markers* whenever a new gene arises. The genome of a neural network evolved with NEAT is composed of a list of nodes and a list of connections. Starting with a fully connected minimal network new nodes are added by splitting existing connections, adding a new node which has a connection from the source node and to the destination node. New connections can then be added to and from this node through mutation. In either case, whenever a connection is added it is assigned a unique *innovation number*, implemented simply as a running counter (Figure 2, left).

By comparing these identifiers similar structures in dissimilar genotypes can be easily and efficiently identified, allowing the distance between two individuals to be calculated (Figure 2, center). This *compatibility distance* is used by NEAT to cluster similar individuals into species, and we can use it as a kernel for our GP, allowing us to perform predictions across dissimilar structures.

The canonical NEAT [30] introduces several coefficients and normalization factors which provide additional degrees of freedom in how exactly this value is calculated, but we simplify it here to:

$$\delta(\mathbf{x_i}, \mathbf{x_j}) = c_1 \cdot G(\mathbf{x_i}, \mathbf{x_j}) + c_2 \cdot \overline{W}(\mathbf{x_i}, \mathbf{x_j}) \tag{8}$$

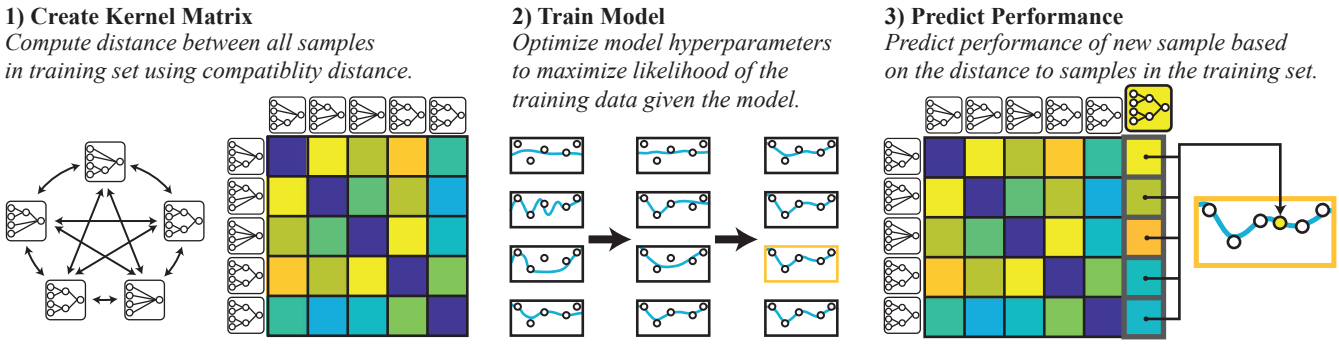

Figure 3: *Predicting Performance from Compatibility Distance*
To predict performance based on compatibility distance a kernel matrix must first be created by comparing every individual in our training set to every other with the kernel function. We then train the model hyperparameters to maximize the likelihood of the training set given the model. Fitness for an unknown individual can then be predicted using the compatibility distance from each individual in the training set and the found model hyperparameters.

where the compatibility distance δ between two individuals x_i and x_j is the weighted sum of the number of non-matching genes G and the average weight differences of matching genes W. The compatibility distance is used by NEAT to cluster individuals into species. New individuals are compared to representatives of each species found in the previous generation, and join the first whose distance is below a certain threshold (Figure 2, right).

To produce the kernel matrix of the GP we use a compatibility distance kernel function which returns the squared exponential compatibility distance between samples:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{1}{2}\delta(\mathbf{x}_i, \mathbf{x}_j)^2\right) \tag{9}$$

While precision of the predictions when only matching connections and weight differences may be limited, the underlying assumption, that the more similar two individuals are the more similarly they can be expected to behave, holds. The rough predictions produced by the predictive model provide enough information to ensure that higher fitness individuals produce more offspring.

To train a GP model, its hyperparameters are tuned to make the known observations most likely given the model, balancing accuracy and simplicity. We tune two hyperparameters of our kernel: the characteristic length scale (ℓ), which can be thought of as the distance you can move in input space before the output value changes significantly, and the variance (η), how far the output signal varies from the function's mean. Integrating these hyperparameters give us a kernel of the form:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \eta \exp\left(-\frac{1}{2\ell}\delta(\mathbf{x}_i, \mathbf{x}_j)^2\right) \tag{10}$$

These hyperparameters θ are optimized by maximizing the log likelihood of the fitness values $\mathbf{y}$ given the individuals in the population $\mathbf{x}$ and compatibility kernel matrix $\boldsymbol{K}$:

$$\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} - \boldsymbol{\sigma_n}\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{1}{2}\log|(\boldsymbol{K} - \boldsymbol{\sigma_n}\boldsymbol{I})| - \frac{n}{2}\log 2\pi \tag{11}$$

Typically, gradient-based optimization is used to maximize the likelihood, but this is not possible here because the compatibility distance is not differentiable. Instead, we use the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which has been proven as effective at optimizing Gaussian process model parameters as gradient-based methods in other contexts [5]. In addition to the kernel specific parameters ℓ and η the mean (μ) and signal noise (σ) are also tuned.

The training and prediction process can be summarized as follows: all individuals in the training set are compared using NEAT's compatibility distance metric to produce a covariance matrix of their similarity, the hyperparameters of the model are then optimized with CMA-ES to maximize the likelihood of the data given the model, and finally a prediction can be calculated based on the model and distance to the individuals in the training set. This training and prediction process is illustrated in Figure 3.

4 SURROGATE-ASSISTED NEAT

Predictions based on a GP model with a compatibility distance kernel can identify the most promising individuals to test and the most promising genotype regions to explore. By judiciously sampling these individuals we can improve the accuracy of our models in optimal regions and perform the same simultaneous topology search and weight optimization as NEAT, with a focus on data-efficiency. The core algorithmic machinery of NEAT is maintained, with the adjustments needed to place NEAT into a surrogate-assisted framework outlined below and illustrated in Figure 4.

Initialization. This surrogate-assisted variant of NEAT begins just as the original version of NEAT, by initializing a set of minimal networks and testing them. These initial samples and their fitness form the training set of our first model. The distance between all samples is computed and a Gaussian process model trained.

Surrogate-Assisted Evolution. The population is evolved according to the standard mechanisms of NEAT: individuals are grouped into species, a number of offspring are assigned to each species based on their fitness, and finally tournament selection and variation is performed within each species to produce a new population. The compatibility distance between the newly produced individuals

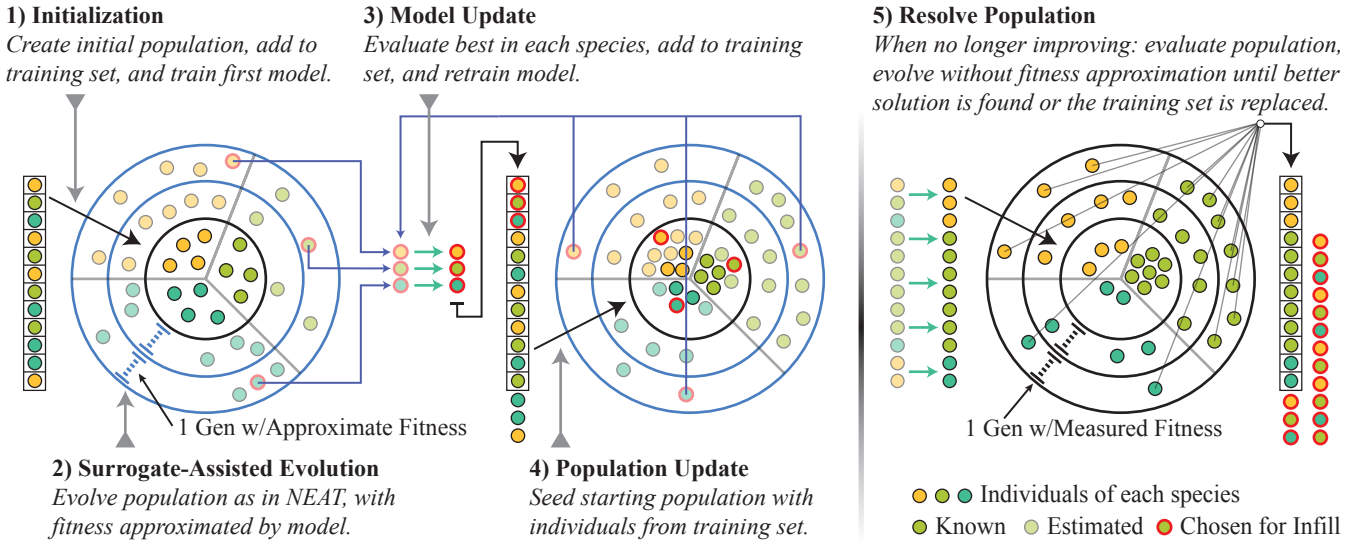

Figure 4: *Surrogate-Assisted NEAT*
1) An initial population of minimal networks is used as a training set, serving as the basis for the initial surrogate model.
2) The population evolves according to the NEAT algorithm, with fitness replaced by the approximations of the model.
3) From each species, the individual with the highest predicted performance is selected and evaluated on the task. These new samples are placed in the training set, replacing the oldest samples if the training set has reached its maximum size.
4) The training set and population are combined to form a new population, and the process repeats from step 2.
5) When fitness is no longer improving, the entire population is evaluated and added to the training set. If none of these individuals are an improvement, a new generation is produced with NEAT, which is evaluated and added to the training set. Evolution continues in this way until a) a better individual is found, or b) the entire training set has been replaced with new individuals. The model is then retrained and the algorithm resumes from (2) with this new population and training set.

and all individuals in the training set is then calculated. Based on the model and this distance, we calculate the utility of sampling each new individual. We reward individuals with high predicted fitness and high uncertainty, using the UCB acquisition function (see Section 2.2). We then repeat the evolutionary process, grouping the new individuals into species, and using this utility value in place of fitness when assigning offspring to species and determining the winners of intra-species tournaments.

Model Update. Surrogate-assisted evolution is repeated a number of times before new samples are added to the model. When selecting these new samples we take advantage of NEAT's concept of species. The species clustering in NEAT ensures that a diversity is maintained, and new complexity nurtured. Species are clustered using the same measure of similarity as our model, and so by sampling one individual in a species we improve the prediction accuracy on other individuals in the same species. To improve our model across species the best performing individual in each is evaluated on the task, added to the training set, and the model retrained.

The training set is limited to a maximum size, and if adding new samples would extend it beyond that size the oldest samples are replaced. This sliding window approach to our training data serves dual purposes. The first is to keep our models relevant to the current individuals being evolved. As the genotypes become more complex the distance from older, simpler individuals becomes less relevant. Older individuals will not only have lower fitnesses, but as the population explores new spaces older individuals will contain many genes which do not exist in the current population, providing little benefit to prediction.

There is also a computational advantage in a smaller training set. New individuals must be compared to every individual in the training set, and the matrix of distances between the training set samples must be inverted when creating the Gaussian process model, an operation with complexity cubed in the number of samples [20]. A limited training set of recent individuals ensures a computationally efficient model focused on relevant and high performing regions.

Population Update. The training set serves another purpose, doubling as a store of known starting points for evolution. As generations of surrogate-assisted evolution repeat, the population drifts farther away from known solutions where reasonable predictions can be made. In typical cases of surrogate-assisted optimization this is not a concern: all solutions occupy the same space and predictions become more accurate as the solution space is explored. With a complexifying genome, however, new dimensions are introduced faster than they can be efficiently explored.

To contain this explosion of genotypic complexity we reintroduce known samples back into the population. Whenever we update the model, we also add one member of the training set for every member of the population, with the most recent added first, effectively doubling the breeding pool for that generation. This larger collection of individuals is divided into species and recombined

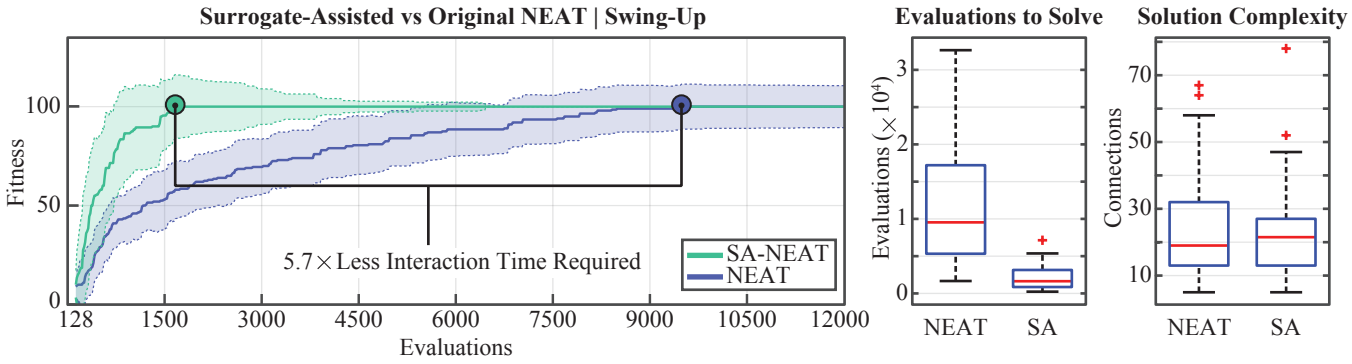

Figure 5: *Cart-Pole Swing-Up: Evaluations to Solve*
Comparison of median best fitness found over 50 trials of SA-NEAT and NEAT on the cart-pole swing-up task. Shaded region indicates one standard deviation from the median. SA-NEAT solved the task in a median of 1664 evaluations, while NEAT required 9488, nearly six times the number of evaluations. Even the most data-efficient quartile of of NEAT runs required as many evaluations as the least data-efficient SA-NEAT runs. The complexity of solutions found in each case are similar, demonstrating that the same variable dimensional space explored by NEAT is also explored in the surrogate-assisted variant.

to form a new population of the standard size. Much of the new population will have known samples as one or both parents, pulling the population back towards known genetic dimensions, allowing more accurate predictions of their fitness.

Resolve Population. In cases where the parameter space is fixed, surrogate-assisted methods will reliably converge on the optima as more samples are obtained, but in an open-ended space this is not the case. In the event that the algorithm converges on a local optimum and stagnates we "resolve" the population, replacing fitness approximations with true fitness values.

If enough newly added individuals are added to the training set without improvement, the entire population is evaluated on the task, revealing any individuals in the population which would achieve higher fitness but were never chosen for evaluation. If no better solutions are found then the speciation and recombination of NEAT repeats, and the entire population again is evaluated on the task. Every individual evaluated is added to the training set, and this NEAT evolution continues until either a better solution is found or the entire training set is replaced with new individuals. At that point the GP model is reconstructed and the algorithm begins again with a diverse, complex, but known population. With the search space once again well-modeled the process of surrogate-assisted evolution, model update, and population update resumes.

5 EXPERIMENTAL RESULTS

5.1 Cart-Pole Swing-Up

Setup and Hyperparameters. We test our approach to surrogate-assisted neuroevolution first on a classic benchmark control problem, the cart-pole swing-up. The system begins with a cart on a two dimensional track with a pole hanging below it, with the objective of swinging the pole into an upright position and maintaining it in a balanced state. This task is more difficult than benchmarks used in many evolutionary computation publications, such as pole-balancing, and cannot be solved with a linear controller [19], requiring networks to grow beyond their initial minimal state.

The known state of the system is the position and velocity of the cart, and the angle and angular velocity of the pole. Inclusion of a bias node results in a total of 5 inputs, with a single output node specifying a command to the system as a percentage of the maximum force. The cart-pole system used here is composed of a 2 kg cart and a 0.5 m pole weighing 0.5 kg. The maximum force which can be applied to the cart is 10 N, with control signals sent to the system at every 0.25 seconds, for a total of 5 seconds.

Controllers are rewarded for the most consecutive time steps in which the pole is held upright. If, for example, the pole is held upright for 25 time steps, falls, then is swung back up and held upright for an additional 15 time steps the controller is only awarded a fitness of 25, not 40. Fitness is only awarded for time steps in the second half of the trial, for a maximum fitness of 100.

NEAT has a large number of hyperparameters, too many to test and tune exhaustively. Instead we conducted preliminary testing with different levels of variation per generation, based on the hyperparameters presented in the canonical NEAT article [30]. The probabilities to add nodes and connections, reenable nodes, perform crossover, and mutate weights were scaled by 2, 1, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$: preliminary tests showed NEAT's best performance when variation was scaled by $\frac{1}{2}$ and so these hyperparameter values were used.

In runs of SA-NEAT, 4 generations of evolution took place before selecting 4 infill individuals to add to the population. These were taken from the top 4 species, except in the case where less then 4 species were present, in which case the highest performing individuals were taken in their place. A training set of 512 individuals was maintained, and the population "resolved" if 128 individuals were added to the training set without improvement. To keep the same amount of variation in one sampling iteration of SA-NEAT as would occur in a single generation of NEAT, the rates of variation were decreased by $\frac{1}{4}$ ($\frac{1}{8}$ of the hyperparameters in [30]). Table 1 outlines the hyperparameters and their relationship.

Results. The comparison of performance between NEAT and SA-NEAT on the swing-up task over 50 replicates is shown in Figure 5.

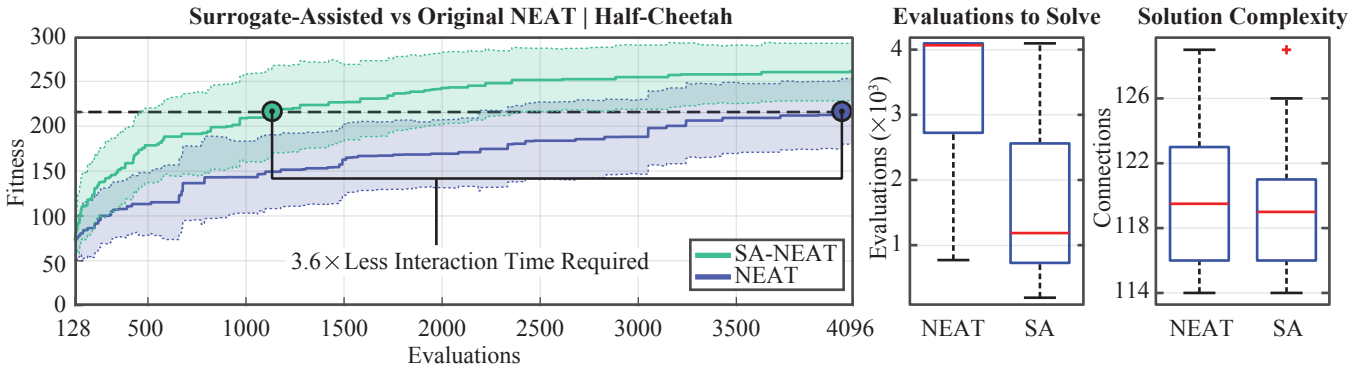

Figure 6: *Half Cheetah: Best Fitness Per Evaluation*
Comparison of median best fitness found over 30 trials of SA-NEAT and NEAT on the half-cheetah running task. Shaded region indicates one standard deviation from the median. With no solve state, data-efficiency was compared by the evaluations required to reach the highest median fitness reached by NEAT. SA-NEAT reached the median level of fitness achieved by NEAT in 4096 evaluations in a median of 1136 evaluations, a difference of more than three and a half times. Even the most data-efficient quartile of NEAT runs required as many evaluations as the least data-efficient SA-NEAT runs. Again, the complexity of the networks produced by the two approaches was equivalent.

Table 1: SA-NEAT Hyperparameter Values and Derivation

Hyperparameter	Relative Value	Absolute Value
# of Species	-	4
Gens Per Infill	-	4
Population Size	-	128
Variation	Base / Gens Per Infill	(Published / 8)
Inds Per Infill	# of Species	4
Training Set	Inds Per Infill × Pop Size	512
Stagnation	Population Size	128

We compare only the number of fitness evaluations performed on the cart-pole simulator: fitness predictions using the surrogate model are not counted. A dramatic speed-up can be observed: by the time NEAT exhausted evaluations equal to 13 generations, half of SA-NEAT runs had already solved the task. This represents a gain in data-efficiency of nearly six times. The acceleration is made even more stark when the full distribution of results is examined. Even the most data-efficient quartile of NEAT replicates require as many evaluations as the least data-efficient runs of SA-NEAT. It should also be noted that the complexity of the produced networks matches those found by NEAT, illustrating that SA-NEAT is indeed exploring the same variable dimensional space as NEAT.

5.2 Half-Cheetah

Setup and Hyperparameters. To test the SA-NEAT approach on a higher dimensional problem, we compare its performance in the half-cheetah running task. The half-cheetah system described in [32] is one half of a quadruped robot with a front and back leg, with each leg consisting of 3 joints. The system has a state space of 17 values: the velocity and angles of each joint, the position and velocity of the body in the x (forward) and z (up) directions, and the angle and angular velocity of the body in y (side). Each of the 6 joints are controlled by sending a torque command, for a total 108 weights in the initial minimal networks (including a bias input).

To encourage smoother gaits we prevent rapid direction shifts in joint direction by applying the output of the neural network not as raw joint torques, but as adjustments to the existing torque levels. Torques on each joint at a time step t is applied as:

$$\mathbf{torque}_{(t)} = \mathbf{torque}_{(t-1)} + \mathbf{y} \tag{12}$$

where $\mathbf{y}$ is the output vector of the neural network.

The OpenAI gym implementation [3] of the half-cheetah is run for 150 time steps[1] , with fitness awarded for moving forward with minimal effort:

$$\text{fitness} = \sum_{t=1}^{150} (\text{pos}_{t+1} - \text{pos}_t)/dt - 0.1 \times \mathbf{torque}_t^2 \tag{13}$$

The same hyperparameters used for NEAT and SA-NEAT in the swing-up task were used here. As there is no half-cheetah solve state, and is much more expensive to simulate than the cart-pole, we limit the number of evaluations to 4096 and run only 30 replicates.

Results. Even in this more complex problem, SA-NEAT outperforms NEAT (Figure 6), reaching the same levels of fitness as NEAT at the end of the trial using only a third of the needed evaluations. This not only confirms our earlier experiment, but also shows that SA-NEAT is able to navigate a high dimensional weight space as well as searching the space of possible topologies.

While the swing-up benchmark is not a trivial task the space of solutions, even in the complexifying case, is relatively limited. With a minimal topology of five inputs and one output it begins as only a five dimensional problem. The half-cheetah, on the other hand, begins in a space that is more than 100 degrees of freedom. As the compatibility distance kernel is independent of the dimensionality of the underlying genotype, our models are still able to make useful predictions in this space with only a few hundred samples.

[1] This is significantly fewer time steps than is typical for this task in reinforcement learning experiments, and so results are not directly comparable to those in the literature. The purpose of these experiments is only to establish the benefits w.r.t. NEAT: more thorough comparisons with previous work will be presented in a future publication.

6 CONCLUSION AND DISCUSSION

We introduced a surrogate-assisted variant of NEAT, SA-NEAT, as a data-efficient method of performing neuroevolution in computationally expensive problems. By taking advantage of the phylogenetic information produced as a byproduct of the evolutionary process, we created a new kernel to judge similarity of neural networks based on their shared genes. Using GP models built with this kernel we are able approximate the performance of individuals, allowing us to achieve similar results with several times fewer evaluations. Fewer evaluations does not guarantee a faster experiment: when the fitness function is evaluated in simulation, there is always a trade-off between the cost of modeling and evaluation. By limiting the model to recently evaluated samples we boundnd its complexity, and it is possible that both accuracy and performance could be further improved with even more purposefully constructed models.

In the presented approach, though species diverged into varied and distant genealogies, a single training set and model were used for prediction. Due to the squared exponential relationship in the kernel, individuals in more distant species have little if any effect on the predicted performance, yet are still considered in the comparison. Producing surrogate models with individuals only from within a single species would reduce the needed number of comparisons. Apart from computational concerns, species specific models could also have more predictive power, as the hyperparameters of the model could more accurately reflect the particular genotype region, rather than their likelihood over the entire training set.

NEAT is used to evolve compositional pattern producing networks (CPPNs) [27] , indirect encodings used to produce neural networks [29], images [16, 24], and solid objects [6]. Whether our approach is as successful in evolving indirect encodings remains to be seen, but as many engineering domains rely on expensive simulations, a data-efficient method of evolving CPPNs would allow their application in real world design problems.

As neuroevolution gains increased attention from industry for its capabilities in large scale problems, the tasks it is charged with will only grow in complexity. Despite the continued growth in computing power there will always be demand for more, and this is especially true for population-based approaches. Combining data-efficient machine learning with neuroevolution ensures that the diversity preserving, novelty seeking, deception avoiding abilities of evolutionary approaches can still be applied, regardless of the complexity of the challenges presented.

ACKNOWLEDGMENTS

This work received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637972, project "ResiB bots") and the German Federal Ministry of Education and Research under the Forschung an Fachhochschulen mit Unternehmen programme (grant agreement number 03FH012PX5 project "Aeromat").

REFERENCES

[1] T Aaltonen, J Adelman, T Akimoto, B Álvarez González, S Amerio, D Amidei, A Anastassov, A Annovi, J Antos, G Apollinari, et al. 2009. Observation of electroweak single top-quark production. *Physical review letters* (2009).

[2] E Brochu, VM Cora, and N De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599* (2010).

[3] G Brockman, V Cheung, L P, J Schneider, J Schulman, J Tang, and W Zaremba. 2016. OpenAI Gym. (2016). arXiv:arXiv:1606.01540

[4] R Calandra, A Seyfarth, J Peters, and M P Deisenroth. 2016. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence* 76, 1-2 (2016), 5–23.

[5] K Chatzilygeroudis, R Rama, R Kaushik, D Goepp, V Vassiliades, and J-B Mouret. 2017. Black-Box Data-efficient Policy Search for Robotics. In *Proc. of IROS*.

[6] J Clune and H Lipson. 2011. Evolving three-dimensional objects with a generative encoding inspired by developmental biology.. In *ECAL*. 141–148.

[7] E Conti, V Madhavan, F P Such, J Lehman, K O Stanley, and J Clune. 2017. Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. *arXiv:1712.06560* (2017).

[8] A Cully, J Clune, D Tarapore, and J-B Mouret. 2015. Robots that can adapt like animals. *Nature* (2015).

[9] S Doncieux, N Bredeche, J-B Mouret, and A E G Eiben. 2015. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI* 2 (2015), 4.

[10] A I J Forrester and AJ Keane. 2009. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences* (2009).

[11] A Gaier, A Asteroth, and J-B Mouret. 2017. Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination. In *Proc. of GECCO*. ACM.

[12] Y Jin. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing* (2005).

[13] J Lehman and K O Stanley. 2008. Exploiting open-endedness to solve problems through the search for novelty.. In *Proc. of ALIFE*. 329–336.

[14] J Lehman and K O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

[15] M Neuhaus, K Riesen, and H Bunke. 2006. Fast suboptimal algorithms for the computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer.

[16] A Nguyen, J Yosinski, and J Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 427–436.

[17] R Pautrat, K Chatzilygeroudis, and J-B Mouret. 2018. Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search. In *Proc. of ICRA*.

[18] R Pfeifer and J Bongard. 2006. *How the body shapes the way we think: a new view of intelligence*. MIT press.

[19] T Raiko and M Tornio. 2009. Variational Bayesian learning of nonlinear hidden state-space models for model predictive control. *Neurocomputing* (2009).

[20] C Rasmussen and C Williams. 2006. Gaussian Process for Machine Learning. *Gaussian Process for Machine Learning* (2006).

[21] K Riesen and H Bunke. 2009. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing* (2009).

[22] T Salimans, J Ho, X Chen, and I Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).

[23] A Sanfeliu and K-S Fu. 1983. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics* 3 (1983), 353–362.

[24] J Secretan, N Beato, D B D'Ambrosio, A Rodriguez, A Campbell, and K O Stanley. 2008. Picbreeder: evolving pictures collaboratively online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1759–1768.

[25] B Shahriari, K Swersky, Z Wang, R Adams, and N de Freitas. 2016. Taking the human out of the loop: A review of bayesian optimization. *Proc. IEEE* (2016).

[26] N Srinivas, A Krause, S Kakade, and M Seeger. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*.

[27] K Stanley. 2007. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines* (2007).

[28] K Stanley, B Bryant, and R Miikkulainen. 2005. Evolving neural network agents in the NERO video game. *Proc. IEEE* (2005), 182–189.

[29] K Stanley, DB D'Ambrosio, and J Gauci. 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life* (2009).

[30] K Stanley and R Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* (2002).

[31] F P Such, V Madhavan, E Conti, J Lehman, K O Stanley, and J Clune. 2017. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv:1712.06567* (2017).

[32] P Wawrzynski. 2007. Learning to control a 6-degree-of-freedom walking robot. In *EUROCON: The International Conference on" Computer as a Tool"*.

[33] S Whiteson and P Stone. 2006. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research* 7, May (2006), 877–917.

[34] X Zhang, J Clune, and K O Stanley. 2017. On the Relationship Between the OpenAI Evolution Strategy and Stochastic Gradient Descent. *arXiv:1712.06564* (2017).