Expanding variational autoencoders for learning and exploiting latent representations in search distributions

Unai Garciarena*
Intelligent Systems Group
University of the Basque Country
Donostia-San Sebastian, Spain
unai.garciarena@ehu.es

Roberto Santana
Intelligent Systems Group
University of the Basque Country
Donostia-San Sebastian, Spain
roberto.santana@ehu.es

Alexander Mendiburu
Intelligent Systems Group
University of the Basque Country
Donostia-San Sebastian, Spain
alexander.mendiburu@ehu.es

ABSTRACT

In the past, evolutionary algorithms (EAs) that use probabilistic modeling of the best solutions incorporated latent or hidden variables to the models as a more accurate way to represent the search distributions. Recently, a number of neural-network models that compute approximations of posterior (latent variable) distributions have been introduced. In this paper, we investigate the use of the variational autoencoder (VAE), a class of neural-network based generative model, for modeling and sampling search distributions as part of an estimation of distribution algorithm. We show that VAE can capture dependencies between decision variables and objectives. This feature is proven to improve the sampling capacity of model based EAs. Furthermore, we extend the original VAE model by adding a new, fitness-approximating network component. We show that it is possible to adapt the architecture of these models and we present evidence of how to extend VAEs to better fulfill the requirements of probabilistic modeling in EAs. While our results are not yet competitive with state of the art probabilistic-based optimizers, they represent a promising direction for the application of generative models within EDAs.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; **Neural networks**; • **Theory of computation** → **Continuous optimization**;

KEYWORDS

machine learning, variational autoencoder, estimation of distribution algorithm, neural network, generative modeling

ACM Reference Format:
Unai Garciarena, Roberto Santana, and Alexander Mendiburu. 2018. Expanding variational autoencoders for learning and exploiting latent representations in search distributions. 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/10.1145/3205455.3205645

*Corresponding author.

1 INTRODUCTION

In addition to their primary goal of finding high-quality solutions in the decision space, one desideratum of efficient Evolutionary Algorithms (EAs) when applied to black-box domains is to exhibit some modeling capacity. In the single-objective scenario, this capacity is related to the question of capturing and representing the dependencies between the problem variables in the selected solutions. In multi-objective optimization problems (MOPs), modeling can also include the relationship between decision variables and objectives, particularly to accurately model this relationship in the constrained space of non-dominated solutions.

Information about interactions between decision variables and objectives can be used to design factorized distribution algorithms or gray-box optimization methods that exploit the problem structure [29]. Models can also be useful if they serve to generate solutions similar to the optimal ones, i.e., if they can be applied for *generative modeling*. In EAs, modeling of variables and objective interactions has been mainly applied in estimation of distribution algorithms (EDAs) [23].

A variational autoencoder (VAE) [17] is a neural network approach that focuses on learning a generative model of the data. This approach assumes that the information being treated has a latent representation. One of the main characteristics of the VAE is that it learns this latent representation in a process in which samples similar to the input data are incrementally generated. In this paper we investigate the suitability of VAEs as a model to capture the dependencies between variables and objectives in an EA. These dependencies are later used for generating high-quality solutions. We also introduce new variants of the VAE models that allow us to explicitly represent the relationship between the decision variables and objectives.

The main contributions of this paper are the following: 1) We apply VAE to learn probability distributions in EAs and use it as a generative model to produce new samples. 2) We introduce new types of VAE-related models that, similarly to previous generators, include encoder and decoder components, but also add a third *approximator* component. 3) We show that it is feasible to incorporate the VAE-extended models to EDAs. We have not found any reference to previous applications of VAE for modeling in EAs or the use of neural networks for explicitly capturing and modeling the dependencies between decision variables and objectives.

2 PROBABILISTIC MODELING OF SEARCH DISTRIBUTIONS

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be an n-dimensional vector of continuous variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. S will denote a set of indices in $\{1, \ldots, n\}$, and X_S (respectively x_S) a subset of the variables of $\mathbf{X}$ (respectively $\mathbf{x}$) determined by the indices in S.

The idea behind EDAs and other model-based EAs is to define search distributions that associate a probability $p(\mathbf{x})$ to the points of the search space. Points with higher fitness will have a higher probability. Two crucial steps for EDAs are to learn a compact representation of the search distribution (learning step) and use this representation to sample new solutions (sampling step).

Let us consider a set $\mathscr{X} = \{\mathbf{x}^i\}_{i=1}^N$ consisting of N solutions. We assume that there exists some unknown distribution $p(\mathbf{x})$ (optimal search distribution) that *causes* - and therefore, perfectly fits - $\boldsymbol{X}$. However, this distribution is unknown, and can only be approximated with the data in $\mathscr{X}$, $p_d(\mathbf{x})$. We are interested in generating new samples that are similar to those in $\mathscr{X}$. This can be approached as a *generative modeling* problem in which we would first approximate the search distribution $p(\mathbf{x})$ and then generate new samples from our approximation.

In some modeling problems, it is convenient to assume the existence of an unobserved (latent) continuous random variable $\mathbf{Z}$ that influences the generation process. For instance, the generation process could consist of a first step where a value $\mathbf{z}$ is generated from some distribution $p_z(\mathbf{z})$ (the prior distribution), and a second step where some conditional distribution $p(\mathbf{x}|\mathbf{z})$ (decoding distribution) is used to generate $\mathbf{x}$. In search distributions, this generation process may not have been part of the search strategy, however, addressing the modeling problem using the latent variable formalism can lead to good approximations of $p(\mathbf{x})$ and effective methods for generating samples that resemble those in $\mathscr{X}$.

2.1 Related work

There are a number of strategies applied in EDAs that implicitly assume or model hidden variables. The most extensively applied model of this type are the mixtures of distributions [5, 8]. A limitation of the mixtures models is that, in most cases, the number of components have to be fixed in advance. Also, mixture-learning algorithms such as expectation-maximization (EM) can be computationally very costly.

A number of works have proposed the use of latent representations as a more accurate way to model dependencies. In Boltzmann Machines [2] and Restricted Boltzmann Machines (RBMs) [32], latent variables are added to the model to capture a higher order representation of the patterns in the data. Recent deep neural network models such as deep belief nets [28] and deep Boltzmann machines also incorporate several layers of hidden variables to improve representation.

Nevertheless, latent models exhibit a number of drawbacks. They can be very difficult to learn, and, for certain models, exact inference is intractable. In EDAs, inference is mainly involved as a step in the generation of new solutions. For complex models with latent variables, Markov chain Monte Carlo (MCMC) algorithms would be required to generate new solutions [30, 31]. In this paper, we propose the use of the VAE, for which learning is simpler, and MCMC methods are not required. RBMs have been also applied to model binary problems in single-objective [26] and multi-objective EDAs [34]. In comparison to RBMs, VAEs are easier to train and the sampling process is much faster.

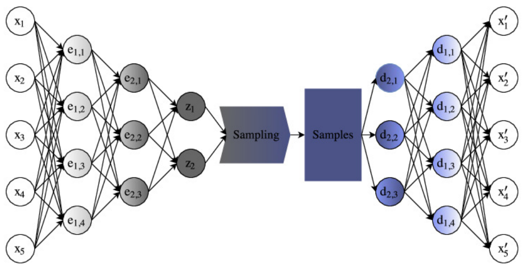

Figure 1: VAE architecture.

Generative adversarial nets (GANs) [11] and denoising autoencoders [6, 24] are exemplars of the few neural-based models used in EDAs. GAN-EDA was introduced in [25] and applied to single objective problems. However, the algorithm did not produce results competitive to other simpler approaches, neither in the number of fitness evaluations nor in the computational time [25]. Also in [3], DNNs with 5 and 10 layers are used as neural models, it is acknowledged that the learning process can be time consuming. While the Deep-Opt-GA is evaluated across a set of diverse artificial and real-world problems, it is not possible to determine the gain of the algorithm over EDAs since it is compared to a fast local optimizer and a GA.

In the area of deep neural networks, some recent works propose modifications and extensions to multi-network generative models. Although we have not found previous examples of VAE variants with three or more models, there are such multi-network models developed from GANs. In [16], an example of a GAN with two discriminators (ordinary GAN has a single discriminator) and one generator is proposed. The so-called DiscoGAN model was designed to discover relations between two unpaired, unlabeled datasets. Furthermore, recent work has developed the integration or combination of different classes of generative models in more flexible models, comprising heterogenous neural networks [21]

3 VARIATIONAL AUTOENCODERS (VAES)

VAE [17] is a generative model that explicitly represents a latent variable $\mathbf{z}$. A variational autoencoder can be split into two components: an *Encoder* and a *Decoder*. The general goal of the model is to approximate the probability distribution:

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})dz \tag{1}$$

where θ comprises all the decoder parameters.

The key idea behind the variational autoencoder is to attempt to sample values of $\mathbf{z}$ that are likely to have produced $\mathbf{x}$. For that purpose, it learns an encoding function $q(\mathbf{z}|\mathbf{x})$ that serves as an approximation of the prior distribution $p(\mathbf{z})$.

VAE is considered a *generative* model since it can be used to generate new samples that resemble those used for training it. Both the encoder (E) and the decoder (D) are implemented as neural networks, more specifically as multi-layer perceptrons (MPLs).

The encoder receives a vector $\mathbf{x}$ from the dataset as input and produces a vector $\mathbf{z}$, which is interpreted as a mean μ and a standard deviation σ coding. These μ and σ are then used to parametrize a normal distribution, before it is sampled, $\mathcal{N}(\mu, \sigma)$.

Latent variable $\mathbf{z}$ is assumed to follow a multivariate normal distribution with mean 0 and diagonal covariance matrix I, $\mathcal{N}(0, I)$.

The samples obtained from $\mathcal{N}(\mu, \sigma)$ are fed to the decoder, which is asked to reproduce, from each of the samples, the same image that was inputted to the encoder. Namely, $\tilde{\mathbf{x}}$ according to $\tilde{\mathbf{x}} \sim p_d(\mathbf{x}|\mathbf{z}; \theta)$.

Training a VAE pursues two goals: 1) Optimize E to obtain an accurate approximation of the latent variable distribution $\mathbf{z}$. 2) Optimize D to obtain an accurate reconstruction during the decoding phase. An example of the VAE architecture is shown in Figure 1.

As in other neural network models, VAE uses the back-propagation algorithm, assisted by a gradient descent optimizer, to optimize the parameters of the network (weights and biases). The optimizer attempts to find the minimum of the loss function. VAE minimizes the following upper-bound on the negative log-likelihood of $\mathbf{z}$:

$$\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[-log(p(\mathbf{x}|\mathbf{z}))] + KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] \quad (2)$$

where $\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}$ and $\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}$ respectively represent the expectations with respect to the original data distribution p_{data} and the latent variable distribution $p_{\mathbf{z}}(\mathbf{z})$. The KL function stands for the Kullback-divergence metric [18] .

The first term of the equation represents the goal of maximizing the quality of the reconstruction of the original data given the latent variable. The second encodes the goal of accurately approximating the prior distribution $p(\mathbf{z})$.

The general question of learning the optimal VAE architecture for a given problem (i.e., the number of layers, activation functions, etc.) is open. Usually, the architecture is set by hand.

4 VAES FOR MODEL-BASED OPTIMIZATION

In this section, we introduce the models that are later tested in the experiment section. The models proposed can serve several purposes - such as image generation, or data augmentation -, however, our interest lies in optimization. Therefore, the development of this work is addressed from this perspective. When considered in the context of search distributions, we expect an "ideal" VAE model to simultaneously satisfy the following conditions:

(1) It can accurately learn a (latent) model from the best solutions. This is, the model should assign more sampling probability to those solutions that are used in the training phase.
(2) It can be efficiently used as a generator of solutions that resemble those in the original data.
(3) It learns a mapping between decision variables and the objective variable(s):
    (a) The values of the decision variables can be predicted using the values of the objective variables (Fitness-conditioned sampling).
    (b) The values of the objectives can be predicted from the values of the decision variables (Fitness-estimation).

Properties 1 and 2 are those usually required by any probabilistic model used in EDAs, and met by regular VAEs. These models are *implicitly* biased towards sampling solutions with high fitness values, since only this kind of solutions are used for training. Property 3a) would serve to *explicitly* bias the sampling of solutions to try to obtain variable configurations that produce a given fitness value. This property can be particularly useful for sampling specific areas of the Pareto front in MOPs [15, 22]. Property 3b) can be useful when the model is applied as a surrogate, or to evaluate only those solutions that are predicted by the model to be of high fitness [4].

4.1 Extending the VAE model

The original VAE model does not exhibit all the ideal mentioned properties. Therefore, we propose two extensions of the model. These extensions correspond in fact to new generative models that incorporate an additional neural network component and modified loss functions. However, the general principles of VAEs are maintained.

4.1.1 Extended VAE. The E-VAE is a variation of the regular VAE which consists of the common encoder-decoder combination network, plus a third one, the *predictor*. This third component behaves as another decoder that tries to predict the fitness value $f(\mathbf{x})$ associated to a solution $\mathbf{x}$. The input to this second decoder is the same $\mathbf{z}$ that the encoder produces. We assume that if $\mathbf{z}$ has sufficient information to reconstruct the original solution, it can also be used to approximate $f(\mathbf{x})$. The two decoders are expected to learn different parameters since they solve different decoding tasks.

The E-VAE structure is trained simultaneously by minimizing an expansion of the loss function described by Equation (2), to which a new term that specifies the optimality goal for the third network is added. This term is the mean squared error (MSE) between the fitness values predicted by the network $\hat{f}(\mathbf{x})$, and $f(\mathbf{x})$.

We expect that the inclusion of a second decoder will also work as an indirect regularizer for the latent representation that $\mathbf{z}$ encodes since now it has to be informative enough to accomplish two different tasks. The third network additionally provides another asset to the model. In cases in which the evaluation function being optimized by the EDA is excessively costly to compute, it can be used by the algorithm as a fitness surrogate. This application would discard solutions predicted to have poor fitness values and avoid evaluating them, saving computational time.

A diagram of E-VAE is shown in Figure 2b. It can be compared to the regular VAE shown in Figure 2a. Note, in the diagram, the characteristic feature of E-VAE, the same $\mathbf{z}$ samples are used to produce a new solution, and an approximation of the fitness value corresponding to that solution.

4.1.2 Conditioned, Extended VAE. The CE-VAE pursues the same goal as E-VAE, to sample solutions using the information known about the fitness of the solutions. However, in this model, this information is exploited explicitly, again, to bias the sampling process towards the highest quality solutions.

The overall structure is the same as the one for the E-VAE. Nevertheless, the input of the original decoder is a concatenation of the $\mathbf{z}$ samples and $f(\mathbf{x})$. Figure 2c illustrates the differences between

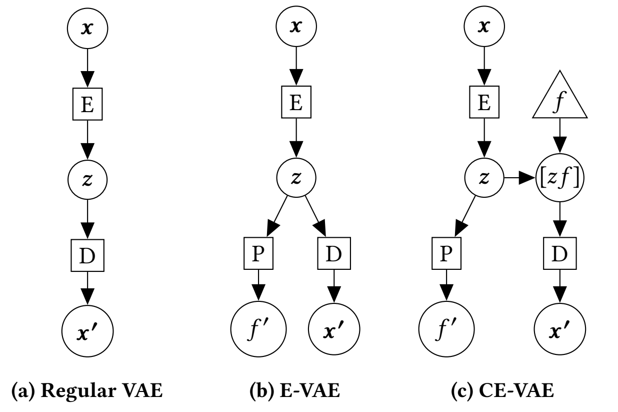

Figure 2: Diagrams of the three models tested in this work, regular VAE and its two variants. f corresponds to the value(s) the samples are biased towards.

E-VAE and CE-VAE. Notice in the figure that these differences are located in the right branches of the two models.

The random component in the procedure of sampling from $\mathbf{z}$ leaves a margin to the decoder to produce, given a single $\mathbf{z}$, various solutions. This structure aims at exploiting this characteristic, as the introduction of explicit fitness values that belong in the higher section of that margin would likely improve the input it has been given. Ideally, the introduction of these fitness values would affect the solution generation procedure, producing an improved outcome. Additionally, note that the third *predictor* network from the E-VAE structure is transferred to this one. This way, the advantages mentioned in the previous model, such as the possibility of using a surrogate fitness evaluation, are maintained.

We designed a set of Python classes that implement the different VAE models. The network architectures are defined using the `tensorflow` library [1]. The VAEs used in this work are extensions of the VAE implementation[1].

5 EXPERIMENTS

To test the validity of the proposed models to drive an EDA, two key issues need to be addressed. The first one, addressed as a preliminary step, is to evaluate the capacity of the models to fulfill two key tasks necessary to succeed when placed in an evolutionary framework: sampling both *accurate* and *diverse* points similar to the input data. This evaluation is conducted isolated from the EDA optimization context and therefore is valuable to assess the quality of the introduced models in other applications where it is possible to classify the variables in two classes: a set of independent variables (our decision variables in the optimization scenario) and dependent variables (the objective variables in the optimization scenario). After completing the assessment of the models, we evaluate their use within the context of EDAs.

[1]https://jmetzen.github.io

5.1 Preliminary experiments

Evaluating the quality of generative models is hard since the generated data should be similar but not identical to the inputs. This evaluation can be difficult to implement automatically for certain problems. For example, it is not trivial to design scores that accurately evaluate the quality of generated images since there is an important perceptual and subjective component involved.

In this paper we test the generative models in the task of generating solutions from the Pareto set (PS) of a bi-objective function. We expect that good generators will produce solutions well spread in the PS, and we will measure the quality of these solutions using the images of those points, i.e., the Pareto front (PF) approximations of the model-generated points. The suitability of this task to evaluate generative models is tudied in more detail in [9].

As bi-objective problems, we use a number of instances originally introduced in [20] to investigate the difficulty of MOPs with complex PSs. More specifically, we select functions $\{F1, \ldots, F9\} \setminus F6$ from those introduced in [20]. F6 is excluded because it is a tri-objective function.

The experiment consists of:

(1) Generate 1000 points in the Pareto set of a given function $f \in \{F1, \ldots F9\} \setminus F6$. The fitness values of the points are computed.
(2) Generate a triple of models (VAE, E-VAE, CE-VAE) whose encoder and decoder have the same architectures. To do this, the number of layers, neurons in layers, and activation functions in the encoder and decoder networks are chosen randomly, and replicated for all three models. Analogously, the architecture configuration of the *predictor* in E-VAE and CE-VAE match too.
(3) The generated models are trained with the 1000 solutions sampled from the PS of one of the functions. E-VAE and CE-VAE are provided the fitness values corresponding to these solutions too.
(4) Each of the trained models is used to sample another set of 1000 solutions.
(5) The similarity between the points in the true PS and in the approximated PSs corresponding to the samples generated by each model are computed using the inverted generational distance (IGD) [7].

The IGD metric measures the smallest distance between two sets of solutions. One of the sets is the one containing the points known to be in the Pareto set (R), while the other is composed of the points sampled from the model A. IGD measures the diversity of A, as well as its closeness to R. It is computed as:

$$IGD = \frac{1}{|R|}\left(\sum_{r \in R} min_{a \in A} d(r,a)^p\right)^{\frac{1}{p}}, \quad d(r,a) = \left(\sum_{k=1}^{m}(r_k - a_k)\right)^{\frac{1}{2}} \tag{3}$$

This experiment was performed 500 times for each function-model combination. The architectures of the networks in the models could have between one and ten hidden layers, with a maximum of 50 neurons in each layer. The other parameters that could be chosen in the random generation of the networks are:

- **Latent variable distribution**: `uniform`, `normal`.

- **Activation function**: None, relu, elu, softplus, softsign, sigmoid, tahn.
- **Loss function**: MSE, Gaussian Divergence.
- **Weights initialization**: xavier [10], random, uniform .

5.2 EDA scenario

The goal of this experiment is to investigate the behavior of the VAE and CE-VAE models when used as an EDA component. In this case we address a simplified protein-folding single-objective problem. We use three instances of this problem and compare the quality of the results obtained by the EDAs that use VAE, E-VAE, and CE-VAE as models to the Continuous Univariate Marginal Distribution Algorithm (UMDA$_c$) [19] - an algorithm that does not consider interactions between variables - and a random search.

5.2.1 *Off-lattice protein model.*

Off-lattice models are simplified protein models that do not follow a given lattice topology. The 2D or 3D coordinates in the real axis define the positions of the protein residues. Among the off-lattice models with known lowest energy states is the *AB* model [33], where *A* stands for hydrophobic and *B* for polar residues. The distances between consecutive residues along the chain are held fixed to $b = 1$, while non-consecutive residues interact through a modified Lennard-Jones potential. There is an additional energy contribution from each angle θ_i between successive bonds. The energy function for a chain of *n* residues, which is to be minimized, is shown in equation (4).

$$E = \sum_{i=2}^{n-1} E_1(\theta_i) + \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} E_2(r_{ij}, \zeta_i, \zeta_j), \tag{4}$$

where

$$E_1(\theta_i) = \frac{1}{4}(1 - cos\theta_i) \tag{5}$$

and

$$E_2(r_{ij}, \zeta_i, \zeta_j) = 4(r_{ij}^{-12} - C(\zeta_i, \zeta_j) r_{ij}^{-6}) \tag{6}$$

Here, r_{ij} is the distance between residues *i* and *j* (with $i < j$). Each ζ_i is either A or B, and $C(\zeta_i, \zeta_j)$ is $+1$, $+\frac{1}{2}$, and $-\frac{1}{2}$ respectively, for *AA*, *BB*, and *AB* pairs, giving strong attraction between *AA* pairs, weak attraction between *BB* pairs, and weak repulsion between *A* and *B* [13].

We consider Fibonacci sequences defined recursively by

$$S_0 = A, \quad S_1 = B, \quad S_{i+1} = S_{i-1} * S_i \tag{7}$$

where $*$ is the concatenation operator.

A 2D off-lattice solution of the *AB* model can be represented as a set of $n - 2$ angles. Angles are represented as continuous values in $[0, 2\pi]$. The first two residues can be fixed at positions $(0, 0)$ and $(1, 0)$. The other $n - 2$ residues are located from the angles with respect to the previous bond. We look for the set of angles that defines the optimal off-lattice configuration. As instances, we consider Fibbonacci sequences for numbers $(6, 7, 8)$. The respective sizes of these sequences, in the same order, are $n \in (8, 13, 21)$.

The EDAs driven by both VAE and CE-VAE are used to minimize the energy function, shown in Equation (4).

5.2.2 *Parameters of the algorithms.*

The pseudocode of the EDA is shown in Algorithm 1. It makes use of the following functions:

- create_model(pop[, fitnesses]): Given a set of solutions, this function returns a VAE, CE-VAE, or a set of means and variances (for the UMDA$_c$) trained with those solutions. In the case of the CE-VAE, the fitness values associated to the solutions are also inputs of the function.
- sample_model(model[, fitnesses]): Given the VAE, CE-VAE, or the set of means and variances, this function generates samples (new solutions) from it. Equivalently, arbitrary fitness values are provided when CE-VAE is the model chosen.
- best(pop, fitnesses): Given a population, and their respective fitness values, this function returns the best individuals using tournament selection, where the tournament size is $s_t = 10$.

```
pop = generate_population();
while halting condition is not met do
    fitness = evaluate_population(pop);
    selected_pop, selected_fit = select_solutions(pop, fitness);
    model = create_model(selected_pop, selected_fit);
    offs = sample_model(model);
    pop = offs + best(pop, fitnesses)
end
```

Algorithm 1: Pseudo-code for a generic EDA.

The structures chosen for this experimental phase are fixed to a reasonable number of hidden layers ($h_l = 3$), and neurons in each of these layers ($h_n = 50$). The population of the algorithm (number of solutions being evaluated in each iteration) is set to $N = 50$, 100, 200, and 300 points. The stop condition is a maximum number of $n_{gen} = 300$ generations.

Algorithm 1 is run 30 times, and the energy value of the best solution found in each run is used to evaluate the performance of the algorithm.

It is worth mentioning that we have added different shifts to the evaluation function, which ensures that there is no bias in the results. This transformation is performed to avoid any kind of positive correlation between any algorithm and the original problem that could lead to an overestimation of the optimization capacity of said algorithm.

5.3 Results for the comparison of the models

To evaluate the results of the first experiments, we will determine the rank of the three models in terms of the IGD values computed from the PS approximations they generate.

For each of the 8 functions, and each of the 500 executions of related models (with identical encoder and decoder structures), we proceed as follows: Each time the IGD generated by one model was better (lower) than the other, the winner algorithm is awarded one point. This results in a table similar to Table 1, which contains the summary of the comparison for the 500 trials for function *F*9. When read by rows, the table shows in how many pairings the model in the row has obtained a better IGD than the model in that column. For example, it can be seen in Table 1 that CE-VAE obtained a better score than VAE in 294 occasions, and performed worse in

206. The tables for all functions have been summarized in Table 2, by adding the individual tables by rows. Following the example, it can be observed in the summary table the way the cell value corresponding to CE-VAE $F9$ is computed as 294 + 304 = 598.

Additionally, in order to evaluate the margins by which a model outperforms another, we have created a similar table, in which, instead of adding points, a *performance* index is computed, as $-\log\left(\frac{IGD_i}{IGD_j}\right)$ (where i, j are the model index) is added in cell (i, j). An example for $F9$ can be seen in Table 3, and its summarized version in Table 4. Again, the larger the number, the better the performance.

Judging by the results shown in Table 2, we can see some regularities. With the exception of $F8$, CE-VAE has better IGDs overall. In some cases, such as $F1$, $F4$, and $F7$, the most complex model has the upper hand in 160-260 more occasions that its peers. For functions $F2$, $F3$, $F5$, and $F9$, this difference is lowered to 100-150. Finally, $F8$ seems to be an outlier, as it shows very close results between all models, awarding a narrow edge to the E-VAE. The comparisons between E-VAE and VAE do not display as much margin as the CE-VAE showed. In some cases, as in $F1$, $F3$, $F8$ and $F9$, the differences where not superior to 20, while in the rest, they did not exceed 65.

From this comparison, we can extract that the sole addition of the *approximator* does not improve the sampling capability of the model. However, the explicit addition of arbitrary (in this case, chosen from the Pareto front) fitness values seems to be key to obtain better samples, as in all but one function CE-VAE obtained clearly better results.

Regarding Table 4, it can be seen how the superiority of the CE-VAE is maintained regularly across all comparisons. The largely unbalanced values in this table denote that, not only was CE-VAE better in most cases, but when its IGD was worse, it was similar to the superior one. However, a contradiction between the two tables is found when comparing the results achieved for $F8$. When comparing the instances in which one model produced better results than the other, $F8$ showed abnormally balanced results. When comparing the quality of all the solutions, however, this balance is broken, and results more similar to those generated from the rest of the functions were found.

The combination of the results in both tables for $F8$ shows that, even though the scoring was paired, when VAE or E-VAE beat CE-VAE the results were close. However, when it was CE-VAE that produced better results, the difference was much larger. This means that the top solutions produced by the CE-VAE are much better than those generated by VAE.

To back up these conclusions, we have subjected these results to a statistical test. The selected test was the Wilcoxon signed rank test [35], and was applied pairwise. The results back up what is seen in the tables, as the test shows p-values near 10^{-7}, when comparing the CE-VAE with the other models, and only shows statistical difference for $F1$ between the VAE and E-VAE.

$F8$ seems to be the most complicated function to be optimized, as the IGD values range from fractional numbers up to numbers larger than 10. The rest of the functions rarely obtained IGD values above 4, and none above 5. This can be clearly seen in Figure 3. This figure shows the density function of the whole set of IGDs generated in the experiments. It is apparent that $F2$, $F7$ and $F9$ have similar difficulties, as well as $F1$, $F3$, $F4$ and $F5$. $F8$ however clearly poses another kind of difficulty for the algorithm. In any case, it can be seen how the line representing the CE-VAE has sampled more solutions in the section where the top points are found. Meanwhile, the lines representing the sampling by VAE and E-VAE, stay close to each other. It can also be observed how CE-VAE has sampled more poor solutions in $F8$ compared to its competitors, which resulted in the abnormal behavior displayed in previous tables.

As a summary, we can state that CE-VAE produces better results than VAE regularly, and its top results are much better than the best ones produced by the VAE. Taking into consideration these results, we would expect that the CE-VAE would have a good performance when being used as a model running an EDA.

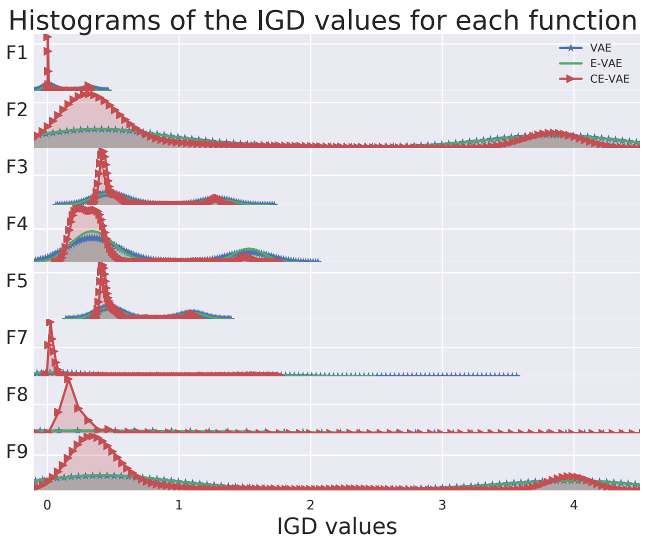

Figure 3: Density function of all the 500 IGDs generated for each function, by each variation of the VAE. For $F8$, a reduced number of configurations that produced an IGD value near 100 were generated. These have been omitted in the representation for a more detailed perception.

Structure	VAE	E-VAE	CE-VAE
VAE	0	247	206
E-VAE	253	0	196
CE-VAE	294	304	0

Table 1: Example of score computation in the comparison of the three models for function $F9$. Each time a IGD generated by a model x is lower than any of those generated by any of its counters y, 1 is added in cell (x, y). A large positive number denotes better performance.

5.4 Results in the optimization scenario

We now present and discuss the results for the second experiment. Firstly, we have empirically set the number of iterations used to learn the VAE models to 5. Table 5 presents the results of the algorithms. It shows the mean of the best solutions for a subset of the

Str.	F1	F2	F3	F4	F5	F7	F8	F9
VAE	403	502	461	462	480	449	488	453
E-VAE	413	437	465	414	427	411	508	449
CE-VAE	678	560	568	621	589	635	499	598

Table 2: Scores for all functions. These scores were computed by adding all the scores in each row of the score matrix for each function. See an example of these matrices in Table 1. These numbers were rounded to integers. A larger value denotes a better performance.

Structure	VAE	E-VAE	CE-VAE
VAE	0	45.2308	-89.7701
E-VAE	-45.2308	0	-135.001
CE-VAE	89.7701	135.001	0

Table 3: Performance index example for $F9$. The lower the number, the better the performance by the model

Str.	F1	F2	F3	F4	F5	F7	F8	F9
VAE	-274	43	25	14	54	-272	-129	-44
E-VAE	-648	-231	-169	-272	-224	-378	-324	-180
CE-VAE	923	187	143	257	169	651	453	224

Table 4: Performance index for all functions. These scores were computed by adding all the indices in each row of the index matrix for each function. See an example of these matrices in Table 3. These numbers were rounded to integers. A larger value denotes a better performance.

30 runs. Results are organized by problem dimension and search algorithm in the columns, and population sizes in the rows. Runs were filtered to remove those that, very early in the optimization, got stuck in local optima whose fitness was very distant from the optimal values. We discarded all runs for which the fitness of the best solution has value above 30. Table 6 displays the number of rejected runs.

From this table, we can firstly see the effects of the applied threshold, since some algorithms could not overcome that threshold for the problem with dimension 34, with low population size.

Focusing only on the smallest variable dimension, 13, it can be seen how a reduced population size clearly harms the performance of the UMDA$_c$, as the random search and all the VAE-based models obtained much better results. As the number of individuals in each generation is increased, the UMDA$_c$ keeps gaining relative to the other searches, to the point of obtaining the best results of all searches, even though closely followed by the VAE-based EDAs. The random search, however, fails to scale from 100 individuals per population onward. This shows how difficult this problem is.

Going on with the next variable dimension, the UMDA$_c$ starts with a clear advantage over the other searches, but fails to improve as the population size is increased. Again, the random search offers a relatively competitive performance with few evaluations per generation, but is unsuccessful when scaling. The VAE and CE-VAE based EDAs follow a similar regular progression, even though shifted in favor of the VAE-EDA. It is the E-VAE based search, however, the one producing top results with the exception of the case in which the population size is 50.

Finally, for the largest variable size, 34, results are poor in general. Only for the larges population size (300) the number of runs over the threshold is significant, which excludes other configurations from producing valid conclusions. The VAE and CE-VAE guided searches offer the best performances.

Table 6 shows the number of runs that could not perform better than the threshold for dimensions 21 and 34. For dimension 13, all but 4 UMDA$_c$ runs performed better than the threshold. This table reflects the difficulty of the problem being treated, as the UMDA$_c$, in a large number of runs for dimension 21, was unable to produce competitive results. Judging by the number of rejected runs with the larges variable dimension, it is safe to assert that larger variable sizes require larger population sizes.

6 CONCLUSIONS

This paper has investigated the convenience of using VAEs as models to drive EDAs. More specifically, two variations of the common VAE are proposed in this work, and all three of them are evaluated in a two-step analysis to prove its effectiveness. The two proposed models, E-VAE (implicitly) and CE-VAE (explicitly), exploit extra information about the fitness function when sampling new solutions.

The first preliminary evaluation determined that the presented model that explicitly uses the extra information was able to produce more distributed samples in the set of top performing solutions, compared to the regular VAE. This is a key feature that is necessary in any model intended to be used in an evolutionary algorithm.

The results provided by the models when placed in an evolutionary framework showed that can improve the outcome of an basic evolutionary algorithm that assumes no interaction between the variables of the problem. However, the stability of the proposed techniques is yet to be achieved, since reduced searches tend to get stuck in local optima very far away from the real optimum value.

6.1 Future work

While we have focused on problems with continuous representations, there are some proposals that extend the application of VAEs to discrete problems [27]. Similarly, variants of VAE, such as the one that adds noise to the original data could be investigated [14].

More work investigating the training technique of the VAE models in evolutionary framework needs to be carried out. The results in the preliminary section suggest that these models have the capacity to perform well in evolutionary methods, and therefore, the issues that caused their poor performance in determined scenarios are worth investigating. Additionally, the VAEs generated at each generation were randomly initialized. Running a search algorithm to optimize the structures of the neural networks within the VAEs would help the sampling quality. Simple local searches could be the best option, as over-complicating the optimization with a costly internal search at each generation would make the EDA infeasible.

Finally, the covariance matrix adaptation [12] is another evolutionary approach in which the validity of the VAEs in optimization problems could be tested.

Dimension	13					21					34				
Pop. size	Rnd	UMDA$_c$	VAE	E-VAE	CE-VAE	Rnd	UMDA$_c$	VAE	E-VAE	CE-VAE	Rnd	UMDA$_c$	VAE	E-VAE	CE-VAE
50	0.201	1.381	0.207	0.237	0.246	4.614	2.027	4.143	5.632	3.510	-	-	10.453	8.418	-
100	0.014	0.503	0.033	0.073	0.019	2.891	2.587	1.530	1.672	2.727	-	11.713	12.114	17.319	-
200	-0.068	-0.181	-0.203	-0.149	-0.206	1.491	4.073	0.368	0.293	0.436	23.942	1.537	5.788	8.214	-
300	-0.108	-0.423	-0.408	-0.420	-0.396	1.135	2.003	-0.261	-0.399	-0.074	23.942	4.623	1.522	2.748	1.353

Table 5: Overall mean results for the 5 search algorithms, for 3 problem dimensions, with 3 different population sizes.

Dim.	21					34				
Pop.	Rnd	U$_c$	V	E-V	CE-V	Rnd	U$_c$	V	E-V	CE-V
50	1	22	1	2	18	30	30	29	29	30
100	0	18	0	0	2	30	27	28	29	30
200	0	14	0	0	0	28	27	6	14	30
300	0	9	0	0	0	28	25	0	2	1

Table 6: Number of runs that did not pass the threshold for dimensions 21 and 34.

ACKNOWLEDGMENTS

U. Garciarena acknowledges a predoctoral grant from UPV/EHU (ref. PIF16/238). This work has received support from the IT-609-13 (Basque Government) and TIN2016-78365-R (Spanish Ministry of Economy, Industry and Competitiveness) programs. We acknowledge the support of NVIDIA Corporation with the donation of a Titan X Pascal GPU also used for this work.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR* abs/1603.04467 (2016). http://arxiv.org/abs/1603.04467

[2] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. 1987. A learning algorithm for Boltzmann machines. In *Readings in Computer Vision*. Elsevier, 522–533.

[3] Shumeet Baluja. 2017. Deep Learning for Explicitly Modeling Optimization Landscapes. *CoRR* abs/1703.07394 (2017). http://arxiv.org/abs/1703.07394

[4] A. E. I. Brownlee, J. A. McCall, and S. K. Shakya. 2012. The Markov Network Fitness Model. In *Markov Networks in Evolutionary Computation*, S. Shakya and R. Santana (Eds.). Springer, 125–140.

[5] D. Cho and B. Zhang. 2004. Evolutionary Continuous Optimization by Distribution Estimation with Variational Bayesian Independent Component Analyzers Mixture Model. In *Parallel Problem Solving from Nature (PPSN VIII)*, Vol. 3242. Springer, 212–221.

[6] Alexander W Churchill, Siddharth Sigtia, and Chrisantha Fernando. 2014. A denoising autoencoder that guides stochastic search. *CoRR* abs/1404.1614 (2014). http://arxiv.org/abs/1404.1614

[7] C.A.C. Coello, G.B. Lamont, and D.A. Van Veldhuizen. 2007. *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer-Verlag New York Inc.

[8] Marcus Gallagher, Marcus Frean, and Tom Downs. 1999. Real-valued Evolutionary Optimization using a Flexible Probability Density Estimator. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, Vol. I. Morgan Kaufmann Publishers, San Francisco, CA, Orlando, FL, 840–846.

[9] Unai Garciarena, Roberto Santana, and Alexander Mendiburu. 2018. Evolved GANs for generating Pareto set approximations. In *Proceedings of the 2018 on Genetic and Evolutionary Computation Conference Companion*. ACM. Accepted for publication.

[10] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 249–256.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[12] Nikolaus Hansen. 2006. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*. Springer, 75–102.

[13] Hsiao-Ping Hsu, Vishal Mehra, and Peter Grassberger. 2003. Structure optimization in an off-lattice protein model. *Physical Review E* 68, 2 (2003), 4 pages. article number 037703.

[14] Daniel Jiwoong Im, Sungjin Ahn, Roland Memisevic, Yoshua Bengio, and others. 2017. Denoising Criterion for Variational Auto-Encoding Framework. In *Proceedings of 31st National Conference on Artificial Intelligence AAAI-2017*. 2059–2065.

[15] H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. 2014. Multiobjective Estimation of Distribution Algorithm Based on Joint Modeling of Objectives and Variables. *IEEE Transactions on Evolutionary Computation* 18, 4 (2014), 519–542.

[16] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. *CoRR* abs/1703.0519 (2017). http://arxiv.org/abs/1703.05192

[17] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *CoRR* abs/1312.6114 (2013). http://arxiv.org/abs/1312.6114

[18] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[19] P. Larrañaga and J. A. Lozano (Eds.). 2002. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers.

[20] H. Li and Q. Zhang. 2008. Multiobjective Optimization Problems with Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation* 13, 2 (2008), 284–302.

[21] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *CoRR* abs/1511.05644 (2015). http://arxiv.org/abs/1511.05644

[22] Marcella SR Martins, Mohamed El Yafrani, Myriam RBS Delgado, Markus Wagner, Belaïd Ahiod, and Ricardo Lüders. 2017. HSEDA: a heuristic selection approach based on estimation of distribution algorithm for the travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 361–368.

[23] H. Mühlenbein and G. Paaß. 1996. From recombination of genes to the estimation of distributions I. Binary parameters. In *Parallel Problem Solving from Nature - PPSN IV (Lectures Notes in Computer Science)*, Vol. 1141. Springer, Berlin, 178–187.

[24] Malte Probst. 2015. Denoising autoencoders for fast combinatorial black box optimization. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 1459–1460.

[25] Malte Probst. 2015. Generative Adversarial Networks in Estimation of Distribution Algorithms for Combinatorial Optimization. *CoRR* abs/1509.09235 (2015). http://arxiv.org/abs/1509.09235

[26] Malte Probst, Franz Rothlauf, and Jörn Grahl. 2017. Scalability of using Restricted Boltzmann Machines for combinatorial optimization. *European Journal of Operational Research* 256, 2 (2017), 368–383.

[27] Jason Tyler Rolfe. 2016. Discrete variational autoencoders. *CoRR* abs/1609.02200 (2016). http://arxiv.org/abs/1609.02200

[28] Ruslan R Salakhutdinov and Geoffrey E Hinton. 2008. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in neural information processing systems*. 1249–1256.

[29] Roberto Santana. 2017. Gray-box optimization and factorized distribution algorithms: where two worlds collide. *CoRR* abs/1707.03093 (2017). https://arxiv.org/abs/1707.03093

[30] S. Shakya and R. Santana (Eds.). 2012. *Markov Networks in Evolutionary Computation*. Springer.

[31] S. K. Shakya, A. E. I. Brownlee, J. McCall, W. Fournier, and G. Owusu. 2009. A fully multivariate DEUM algorithm. In *Proceedings of the 2009 Congress on Evolutionary Computation CEC-2009*. IEEE Press, Norway, 479–486.

[32] Paul Smolensky. 1986. *Information processing in dynamical systems: Foundations of harmony theory*. Technical Report CU-CS-321-86. Colorado University at Boulder. Dept. of Computer Science.

[33] F.H. Stillinger, T. Head-Gordon, and C. Hirshfeld. 1993. Toy Model for Protein Folding. *Physical Review E* 48 (1993), 1469–1477.

[34] H. Tang, V.A. Shim, K.C. Tan, and J.Y. Chia. 2010. Restricted Boltzmann machine based algorithm for multi-objective optimization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 1–8.

[35] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1, 6 (1945), 80–83.