Deep-learning approach to the structure of amorphous silicon

Massimiliano Comin[1,2] and Laurent J. Lewis[1]

[1]*Département de Physique and Regroupement Québécois sur les Matériaux de Pointe, Université de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, Québec, Canada H3C 3J7*

[2]*Institut Néel, CNRS, and Université Grenoble Alpes, 38000 Grenoble, France*

(Received 15 March 2019; revised manuscript received 14 August 2019; published 16 September 2019)

We present a deep-learning approach for modeling the atomic structure of amorphous silicon (*a*-Si). While accurate models of disordered systems require an *ab initio* description of the energy landscape which severely limits the attainable system size, large-scale models rely on empirical potentials, at the price of reduced reliability and a computational load that is still restricting for many purposes. In this paper, we explore an approach based on deep learning, particularly generative modeling that could reconcile both requirements of accuracy and efficiency by learning structural features from data. When trained on a set of observations, such models can generate new structures very efficiently with the desired level of accuracy, as determined by the data set. We first validate our approach by training a convolutional neural network to approximate the potential-energy surface of *a*-Si, as given by the Stillinger-Weber potential, which results in a root-mean-square error of 5.05 meV per atom—about 0.16% of the atomic energy. We then train a deep generative model, the Wasserstein autoencoder, for the generation of *a*-Si configurations. Our approach leads to models which exhibit some of the essential features of *a*-Si while possessing too much structural disorder, thus suggesting that the method is viable; we indicate avenues for improving it towards the generation of state-of-the-art structures.

DOI: 10.1103/PhysRevB.100.094107

I. INTRODUCTION

The detailed structure of *a*-Si remains unresolved: while it has been studied from many different angles over the years, the structure on the local and medium-range scales is still unclear (see, e.g., [1] for a review). The lack of long-range order is such that the structure of the material cannot be univocally extracted from diffraction experiments. In addition, different techniques are used to prepare *a*-Si in the laboratory, and these yield (slightly, if not somewhat) different structures. The most reliable preparation method, which leads to the purest, highest-quality material, is certainly ion implantation [2], where highly energetic Si ions are implanted in a *c*-Si target, causing the matrix to amorphize; the material is then annealed until most defects (vacancies, interstitials, etc.) are removed. With this method, the first coordination number has been found to be 3.88 with an average interatomic distance of 2.352 Å (vs 2.350 for *c*-Si); for second neighbors, the coordination number is 12.43, slightly more than that of the crystal (=12).

Because the structure of *a*-Si is not known precisely, studies of the physical properties of *a*-Si must rely on structural models which can in turn be validated—or discarded—by comparing calculated properties with measured properties. Because of the disorder, however, models are extremely difficult to construct. The structure of *a*-Si is akin to an ideal "continuous random network" (CRN), whose coordination is very close to 4 with a distribution of nearest-neighbor distances that is relatively narrow. It can be, and has been, modeled with varying degrees of success using different methods that have the common feature of requiring some sort of description of the energy of the interacting system. For example, the bond-switching Monte Carlo method of Wooten, Weiner, and Weaire [3,4] employs a harmonic description of the interatomic potentials, which is evidently unrealistic but ensures perfect coordination. An approach based on the activation-relaxation technique (ART) of Barkema and Mousseau [5–7] employs an empirical description of the interaction energies, the Stillinger-Weber (SW) potential [8,9] in this particular case. It could equally employ an *ab initio* formulation. However, there is a price to pay for "quality." *Ab initio* approaches (e.g., density-functional theory [10,11]) are precise and realistic [12], but terribly costly in terms of computer cycles, especially since large systems are usually needed. In fact, these need to be relaxed for long times because of the multiminima structure of the potential-energy surface (PES). Empirical potentials such as SW and Tersoff [13,14] also have limitations: (i) they lack the accuracy of first-principles potentials; (ii) they are not transferable and their parameters typically have to be adjusted "by hand" to adapt to various configurations; (iii) the computational load scales as N^3 in small systems, tending to N in large systems. In practice, the largest models of *a*-Si that have been built contain $N \sim 10^6$ atoms, which roughly translates into a cube with side length 10^2 Å—still very small for many purposes. Clearly, there is still a need for an approach for constructing *a*-Si models that is both accurate and computationally effective, and that would allow the construction of large-scale systems whose dimensions approach those of real systems. While the use of trained machine-learning (ML) potentials in MD simulations is an interesting avenue in this respect [15–19], the route we follow in this work is different.

The complexity of the description of the PES is directly related to the exponential increase of the dimension of configuration space with the number of atoms. This is known as the “curse of dimensionality” and refers to phenomena that arise in very high-dimensional spaces, such as configuration space for a disordered system. The problem is well known in the context of ML, whereby models “learn” from data. Recent advances in deep learning (DL), a branch of ML, have allowed the problem to be largely alleviated, with models able to approximate effectively high-dimensional probability distributions using deep neural networks; for an excellent review of DL, see [20]. For example, DL algorithms can approximate the energy landscape of a system of atoms when trained on a set of configurations and target energies that can be produced with the desired level of accuracy, thus providing an interesting and efficient alternative to empirical potentials [15–19,21–23].

DL models, and in particular deep generative models (see Sec. II B), can characterize and approximate the distribution of complex data sets and generate new realizations of it. This is precisely the philosophy that we adopt in the present work: model the atomic structure of complex materials as a learning task. The quality of the resulting models intimately depends on the quality of the input data set that is used to train the “learner” (algorithm). Constructing this input data set (to the desired level of accuracy) and training the algorithm on it is the most demanding part of the procedure; once the model has learned, new structures can be generated very efficiently.

The purpose of our work is to establish a “proof of concept” using *a*-Si as a test bed. We demonstrate that models exhibiting some of the essential features of the true structure can be generated, and indicate how the approach could be improved towards state-of-the-art models. We start with a description of the DL methodology required for our purpose, then discuss its application for the task of learning the PES, and finally present our model for the atomic structure of *a*-Si.

II. DEEP-LEARNING METHODOLOGY

Machine learning is a branch of artificial intelligence that pertains to the development of data-driven algorithms, whereby the instructions are not explicit but rather “learned” [24]. ML algorithms can be separated in two classes: supervised learning and unsupervised learning. In both cases, the algorithm is provided with a set of observations forming the input data set. In supervised learning, the data set is composed of a set of input and output (or target) values. The algorithm is then trained to approximate a function that makes accurate predictions of the output values when given an input. In unsupervised learning, no output values are provided, and the algorithm is trained to identify the structure and the relationships among the input data, and possibly to generate new realizations of it.

The field of DL designates a class of ML algorithms which process data via a deep cascade of layers, corresponding to multiple levels of representation and abstraction. Their topology, referred to as “architecture,” is often based on neural networks (NNs) where layers composed of several artificial neurons loosely emulate the structure of biological neurons [20,25]. These schemes have been successfully applied within material science in tasks such as predictions of phase diagrams, crystal structures, and material properties, as well as in the development of interatomic potentials and energy functionals that resulted in an increased accuracy and efficiency of materials simulations [26,27].

In this section we discuss briefly the functioning of feed-forward NNs, wherein the connections do not form a cycle (in contrast to recurrent NNs): the information circulates in only one direction (viz. forward) from the input layer to the output layer and through the so-called “hidden layers.” In particular, we examine the various architectures that will be used in this work: the multilayer perceptron (MLP) and a convolutional neural network (CNN) for the prediction of potential energies and the Wasserstein Autoencoder (WAE) for atomic structure generation.

A. Feed-forward neural networks

Research on biological NNs has allowed the main characteristics of the communication between neurons in the nervous system to be identified and modeled. When a neuron receives a signal from the synapses, it is propagated through the dendrites up to the kernel where the processing is done; the output is then carried through the axon. This corresponds to having an input $\mathbf{x}$ processed by a set of parameters $\mathbf{w}$, the synaptic weights (or simply the weights), and a nonlinear function ϕ, giving rise to an output $y = \phi(\mathbf{w} \star \mathbf{x} + b)$, where ϕ is the activation function, $\star$ is any linear operation and b is the threshold, or bias—indeed, a biological neuron is only activated beyond a certain value of the current received. This structure can be generalized to form neuron layers that share the same nonlinearity ϕ. For any type of tensor input $\mathbf{x}$, a neuron layer applies the transformation

$$\mathbf{y} = \phi(\mathbf{W} \star \mathbf{x} + \mathbf{b}), \tag{1}$$

where we assume the dimensions of the parameter tensors $\mathbf{W}$ and $\mathbf{b}$ and of the output tensor $\mathbf{y}$ to be consistent with the operations $\star$ and ϕ. NN layers can be stacked to form a deep neural network (DNN). The number of layers D is called depth. They can have different widths and activation functions: these are called “hyperparameters” in order to distinguish them from the parameters learned by the network (such as $\mathbf{W}$ and $\mathbf{b}$). Therefore, the topology of a feed-forward NN can be summarized as

$$\mathbf{x}^{(l+1)} = \phi_l(\mathbf{W}^{(l)} \star_l \mathbf{x}^{(l)} + \mathbf{b}^{(l)}), \tag{2}$$

where $\mathbf{x}_0$ is the input layer—a layer whose sole task is to feed the inputs to the next layer—and $\mathbf{x}_l$ is the output of layer l. Therefore, each layer is defined in terms of the ranks and dimensions of tensors $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ and by the transformations $\star_l$ and ϕ_l.

The choice of the activation function ϕ is regulated by different factors such as the desired output range or its differential properties. For example, tanh will produce values strictly in $(-1, -1)$ while $\max(0, x)$ allows for fast and efficient optimization.

The choice of the linear operation $\star$ regulates the connectivity between neurons of consecutive layers. Indeed, setting it as the standard matrix multiplication results in having all neurons belonging to a given layer to be connected with those of the previous layer. This is why such layers are also called dense layers. A network that is solely composed of dense layers is called a multilayer perceptron (MLP), and it is the archetypal structure of a DNN.

An example of how the connectivity induced by the transformation $\star$ can impact the performance of a model is the impressive success of convolutional NNs (CNNs) on discrete-topology data, such as time series, and particularly in computer vision tasks [25,28]. As the name implies, a convolution is used instead of a matrix multiplication for the linear transformation $\star$, and usually a "rectified linear unit" $\mathrm{ReLU}(x) = \max(0, x)$ for the nonlinear transformation ϕ. This approach is inspired by the functioning of the visual cortex of some mammals [29]: individual neurons of a given layer respond to stimuli coming from only a restricted region of the visual field, known as the receptive field, which corresponds to a spatially limited connectivity.

According to the universal approximation theorem, a single-layer feed-forward neural network of finite width L can approximate any continuous measurable function with arbitrary precision (for all the most commonly used activation functions) [20]. We note however that L can be in principle very large and that there is no guarantee *a priori* that the NN can learn the desired function.

Once the architecture of the network has been chosen, training is achieved by minimizing a cost function (also called objective or loss function) $\mathcal{L}(\theta)$ which depends on the optimization parameters θ (such as $\mathbf{W}, \mathbf{b}$), and whose form depends on the particular learning problem under consideration.

In the case of supervised learning, the data set is composed of pairs $\{\mathbf{x}, \mathbf{y}\}$ of inputs and targets. We wish the algorithm to find the function that maps $\mathbf{x}$ to $\mathbf{y}$, so the cost function may be any suitable metric measuring the discrepancy between the targets and the algorithm's output. For example, when training a predictive model for reproducing the PES, a suitable choice for the objective function would be the mean-squared error (MSE) between the predicted energies $\hat{y}(\theta)$ and the true energies y. Unsupervised learning focuses instead on finding a function that accurately describes the input data, as no targets are present in the data set. In this case, a suitable cost function can be derived from the maximum likelihood framework. For example, the cost function can be set to the negative log-likelihood (NLL).

The minimization of the cost function is achieved by gradient descent: the parameters θ are updated iteratively following the direction of steepest descent, i.e., $\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta)$, where η is the learning rate controlling the size of the steps. Computing gradients over the entire data set can be inefficient, and is not done in practice. Stochastic gradient descent (SGD) employs instead a stochastic approximation of the gradient by calculating it on a minibatch of inputs randomly selected from the data set. Updates of the parameters throughout the network are achieved by applying iteratively the chain rule of derivative. The back-propagation algorithm, which implements automatic differentiation [30], allows the gradients to be back-propagated from the objective function all the way to the first hidden layer. All implementations of the neural networks considered here were encoded in PYTHON [31] using Theano and Lasagne software libraries [32,33].

B. Deep generative models

Generative modeling is a branch of (unsupervised) ML which aims at approximating the distribution $p_\mathcal{X}(\mathbf{x})$ of a given data set $\mathbb{D} = \{\mathbf{x}_i\}_{i=1,\dots,N}$ defined in a high-dimensional space $\mathcal{X}$. The goal is to build a model which can easily be sampled from, with a distribution $p_G(\mathbf{x})$ that is as close as possible to $p_\mathcal{X}(\mathbf{x})$ according to some probability measure. While achieving both of these goals in the context of ML has long been a challenge, recent algorithmic developments such as variational autoencoders (VAEs) and generative adversarial networks (GANs) have opened new avenues in this regard [34,35], being extremely successful in such difficult tasks as image generation and voice synthesis. In the context of the present study, the distribution we seek to approximate is that of the atomic positions of amorphous silicon.

GANs and VAEs are called latent-variable models since the generation process is based on sampling from hidden (as opposed to observed) variables in a latent space $\mathcal{Z}$. These variables are used by the generative models to determine which features of the data are important for the description and generation processes. More formally, the model's distribution is factored as

$$p_G(\mathbf{x}) = \int_\mathcal{Z} G(\mathbf{x}|\mathbf{z}) p_\mathcal{Z}(\mathbf{z}) d\mathbf{z}, \tag{3}$$

where the latent variable $\mathbf{z}$ follows the distribution $p_\mathcal{Z}(\mathbf{z})$. The generative model then consists of a function (possibly stochastic) distributed according to $G(\mathbf{x}|\mathbf{z})$ that must be optimized in order to make p_G close to $p_\mathcal{X}$.

In this work we consider a latent variable model that similarly to the VAE introduces two networks: an *encoder* $q_\phi : \mathcal{X} \to \mathcal{Z}$ with parameters ϕ and a *decoder* $g_\theta : \mathcal{Z} \to \mathcal{X}$ with parameters θ. The encoder projects the input into a tensor of reduced dimensionality, the latent code. The decoder projects the latent code $\mathbf{z} \in \mathcal{Z}$ back into input space $\mathcal{X}$.

If the latent code follows a known distribution which is easy to sample from (e.g., a normal distribution), then the decoder becomes, at the end of training, a generator. When fed with a latent vector from the aforementioned distribution, its output will match closely the input distribution $p_\mathcal{X}(\mathbf{x})$, as if the latent vector was coming from the encoder.

More formally, the Wasserstein autoencoder (WAE) is a latent variable model whose distribution is given by Eq. (3) that aims at minimizing a specific class of divergences induced by the optimal transport problem [36] given by

$$W_c(p_\mathcal{X}, p_G) = \inf_{\gamma \in \prod(p_\mathcal{X}, p_G)} \mathbb{E}_{(x,y)\sim\gamma} c(x, y), \tag{4}$$

where $\prod(p_\mathcal{X}, p_G)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $p_\mathcal{X}$ and p_G.[1] One may see $\gamma(x, y)$ as the "mass" that needs to be transferred from x to y in order to transform $p_\mathcal{X}$ into p_G; $W_c(p_\mathcal{X}, p_G)$ then corresponds to the cost of the optimal transport plan. When $c(x, y) = ||x - y||_p$, the so-called p-Wasserstein distance is obtained, which possesses several interesting properties with respect to other divergences.

[1]Specifically, $\gamma(x, y)$ verifies $\int \gamma(x, y)\, dx = p_\mathcal{X}(y)$ and $\int \gamma(x, y)\, dy = p_G(x)$.

Under this model, the optimal transport cost of Eq. (4) takes a simpler form:

$$W_c(p_{\mathcal{X}}, p_G) = \inf_{Q:Q(\mathbf{z})=p_{\mathcal{Z}}} \mathbb{E}_{p_{\mathcal{X}}}\mathbb{E}_{Q(\mathbf{z}|\mathbf{x})}c(\mathbf{x}, G(\mathbf{z})). \tag{5}$$

The WAE objective is obtained by relaxing the constraint on $Q(\mathbf{z})$ by adding a penalty to the cost function

$$\begin{aligned}\mathcal{L}_{\mathrm{WAE}}(p_{\mathcal{X}}, p_G) = \inf_{Q(\mathbf{z}|\mathbf{x})\in\mathcal{Q}} \mathbb{E}_{p_{\mathcal{X}}}\mathbb{E}_{Q(\mathbf{z}|\mathbf{x})}c(\mathbf{x}, G(\mathbf{z}))\\ +\lambda\mathcal{D}(Q(\mathbf{z}), p_{\mathcal{Z}}),\end{aligned} \tag{6}$$

where $\mathcal{Q}$ is any set of probabilistic encoders, $\mathcal{D}$ is an arbitrary divergence, and $\lambda > 0$ is a hyperparameter. Two possible ways of regularizing the WAE have been proposed. The first consists of setting $\mathcal{D}$ as the maximum mean discrepancy between $Q(\mathbf{z})$ and $p_{\mathcal{Z}}$, and the second in setting $\mathcal{D} = D_{JS}$, the Jensen-Shannon divergence, and estimate it through adversarial training [37]. The latter is achieved by introducing an adversary NN, the discriminator d_γ with parameters γ, which will learn to separate "true" points sampled from $p_{\mathcal{Z}}$ and "fake" ones sampled from $Q(\mathbf{z})$. This adversary procedure is originally found in GANs, with the difference that it applies here to the latent space, which may have a better structure than the input space. For simplicity, and since this is the model that will be used in this work, we refer to the WAE-GAN as WAE.

Thus, the WAE training is achieved by minimizing the following objectives with respect to the encoder, decoder, and discriminator parameters ϕ, θ, and γ:

$$\begin{aligned}\mathcal{L}_{\mathrm{AE}} &= \frac{1}{B}\sum_{i=1}^{B} c\{\mathbf{x}_i, g_\theta[q_\phi(\mathbf{x}_i)]\} - \lambda \ln f_\gamma[q_\phi(\mathbf{x}_i)]\\ \mathcal{L}_{\mathrm{D}} &= \frac{\lambda}{B}\sum_{i=1}^{B} \ln f_\gamma(\mathbf{z}_i) + \ln\{1 - f_\gamma[q_\phi(\mathbf{x}_i)]\}\ ,\end{aligned} \tag{7}$$

where the sums run over a batch of examples and B is the minibatch size.

III. DEEP GENERATIVE MODEL FOR THE ATOMIC STRUCTURE OF A-SI

As mentioned above, DNNs allow the PES of a material to be approximated efficiently [15–19,21–23]), which can in turn be used in, e.g., molecular dynamics (MD) simulations. These proceed iteratively by moving the system one time step at a time towards a global minimum; the length of the time step is a small fraction of the period of vibration of the atoms and is typically of the order of femtoseconds. Because of the variety of energy barriers found in the PES, some of which can be very large, an MD approach for simulating disordered systems rapidly becomes intractable. Methods such as ART alleviate this problem to a large extent by exploring the PES directly [5–7], but it remains a formidable task for large systems. In contrast, generative models directly approximate the probability distribution of the atomic configurations forming the input training data set, after which new configurations can be generated in a very efficient manner as neither energy calculations nor time iterations are required, new configurations being produced by forward propagating a random vector of the latent space through the generator. This is the essence of our approach.

The first step, therefore, consists in forming the data set on which the NN will be trained, which is evidently of utmost importance for the quality of the learning algorithm and the resulting configurations. We discuss this first, then move on to examine how it translates into learning the PES, by analogy with the work of Behler *et al.* In the last subsection, we present the results of training a WAE for modeling *a*-Si.

A. Construction of the input data set

The statistical learning process requires, and this is essential, a representative sample of the probability distribution that we seek to approximate, i.e., a sufficiently large set of realizations that captures the main properties of the distribution. In our case, this boils down to a set of atomic *a*-Si configurations exhibiting the desired structural properties. A configuration is considered acceptable if it corresponds to a low-lying minimum of the PES (noting that the global minimum corresponds to the crystal) and if its structural properties, such as bond length and bond angle distributions, agree with experimental observations. These configurations constitute the input data set for the NN. We also incorporate the total potential energies of the configurations into the data set, in order to learn the PES and improve the accuracy of the generative model. For the purpose of testing the procedure, and for simplicity, we employed here the modified SW potential as it provides an adequate description of *a*-Si [8,9]; nevertheless, our approach applies to any PES, empirical or *ab initio*. The SW potential is defined as

$$\begin{aligned}E = \epsilon\Bigg[\sum_{\langle ij\rangle}\phi_2\left(\frac{r_{ij}}{\sigma}\right)\Theta\left(a - \frac{r_{ij}}{\sigma}\right)\\ + \sum_{\langle ijk\rangle}\phi_3\left(\frac{r_{ij}}{\sigma}, \frac{r_{ik}}{\sigma}\right)\Bigg] \quad r_{ij}, r_{ik} < \sigma a,\end{aligned} \tag{8}$$

$$\phi_2(r_{ij}) = A\big(Br_{ij}^{-p} - 1\big)e^{1/(r_{ij}-a)},$$

$$\phi_3(r_{ij}, r_{ik}) = \lambda\left(\cos\theta_{ijk} + \frac{1}{3}\right)e^{\gamma/(r_{ij}-a)}e^{\gamma/(r_{ik}-a)},$$

where Θ is the Heaviside step function and ϵ, A, B, σ, p, a, λ, γ are parameters that have been fitted to the amorphous phase of silicon.

The data set was prepared in two steps: First, a number of 216-atom *a*-Si configurations, in a volume (16.282 Å)^3 with periodic boundary conditions, were constructed using ART [5–7]. This approach consists in exploring the PES of a system by searching transition paths between energy minima in an iterative fashion. Starting with a configuration in a local minimum, the algorithm looks for a saddle point and moves the system to it (activation) and then relaxes it to a new, local minimum (relaxation). The activation step is achieved with iterative diagonalization routines for the Hessian such as the Lanczos algorithm while the relaxation step can be achieved with any descent method such as conjugate gradients.

New minima in the PES were saved once every ten ART steps in order to reduce correlations. Structurally equivalent configurations were discarded, except one, as were configurations with positive energies. This procedure resulted in 23 075 different configurations and corresponding energies. Second, using these configurations as starting points, the data set was extended by carrying out MD simulations for a duration of 2 ps each, at various temperatures in the range 50–1200 K, new configurations being saved at 15-fs intervals; structurally equivalent and positive-energy configurations were again removed. Overall, this resulted in a data set $\mathbb{D} = \{\mathbf{x}_i, E_i\}_{i=1}^{N}$, containing $N = 121\,163$ *a*-Si structures $\mathbf{x}_i$ with corresponding potential energies E_i.

B. Learning the PES

The atomic structure of a given configuration in the input data set $\mathbb{D}$ is described by the set of all atoms' Cartesian coordinates. However, this choice of representation is not unique and has a major drawback: the symmetries inherent to the system are not taken into account. Cartesian coordinates are not invariant under global orthogonal transformations and atomic index permutations, while the atomic structure and its potential energy are. Training a NN to approximate the PES with such a representation would certainly fail: equivalent configurations would be treated as different and the total potential energy would not be invariant to the system's symmetries. It is thus of utmost importance to represent the atomic coordinates in a suitable, invariant fashion for any learning problem. Here we follow the procedure introduced by Behler [23]; a configuration is mapped onto a set of symmetry functions describing the local environment of each atom in an invariant fashion.

We now assess the methodology by training two different NNs (a MLP and a CNN—see below) for learning the PES. This will allow us (i) to validate the representation of the atomic configurations in terms of symmetry functions, (ii) to ensure that it is indeed possible to learn from $\mathbb{D}$, and (iii) to assess two different choices of architecture.

For this supervised learning task, the data set has been divided into a training set ($\sim 80\%$), a validation set ($\sim 10\%$) to monitor the model's performance during training, and a test set ($\sim 10\%$) for estimating, in an unbiased way, the performance on configurations it has never seen. Each atomic configuration was represented by a set of 18 symmetry functions [23] of type G_1, G_2, and G_3, for a total of 54. Symmetry functions G_4 and G_5 were not used as they require the calculation of bond angles, which implies a considerable computational overhead. Hence, an atomic configuration with Cartesian coordinates $\mathbf{x} \in \mathbb{R}^{216\times 3}$ is transformed into a set of 54 symmetry functions $\mathcal{G} \in \mathbb{R}^{216\times 54}$, and periodic boundary conditions (PBCs) are used. In contrast to Behler *et al.*, the symmetry functions are not constructed with fixed parameters but, rather, are implemented as a neuron layer such that their parameters can be optimized on the fly while learning the PES, just as any other parameter of the NN. In both NNs presented in this section, the first layer maps the atomic configurations onto the symmetry functions. Moreover, each 216-atom configuration is considered as a minibatch of 216 data points: each atom is thus treated in parallel and one gradient step per configuration is performed. The total energy is then evaluated as the sum of individual atomic energies $E = \sum_{i=1}^{N} E_i$, computed on the last layer of the network.

TABLE I. Comparison of MLP and CNN performance for the prediction of the potential energies over the test set. The total number of parameters N_θ is equal for both NNs.

	N_θ	Epochs	RMSE (meV/atom)	
			train	test
MLP	34853	8	4.621	6.978
CNN	34853	8	4.496	5.095

We considered two architectures with the same number of parameters, illustrated in Fig. 1. The first is a MLP, similar to the approach of Behler *et al.* [21], viz. a five-layer MLP with ReLU activations. The second is a five-layer CNN, where the spatial dimension (the one over which the convolution is applied) corresponds to the different symmetry functions. The filter size has been set to 1, since symmetry functions do not exhibit any particular local structure, thus reducing the number of parameters and preventing overfitting. In that specific case, the convolution operation becomes a simple dyadic product $(\mathbf{w} \otimes \mathbf{x})_{ij} = w_i x_j$.

Both NNs have been trained with the adaptive moment estimation algorithm ADAM ($\eta = 10^{-4}$, $\beta_1 = 0.5$, $\beta_2 = 0.9$, $\lambda = 10^{-6}$) with weight decay [38,39], where η is the learning rate, β_1 and β_2 are exponential decay factors for the first and second moment estimates, respectively, and λ is the weight decay factor. The objective function is the MSE between the estimated energy $\tilde{E}(\mathbf{x})$ and the target energy y, that is the function

$$\mathcal{L}(\mathbf{x}, y) = [\tilde{E}(\mathbf{x}) - y]^2, \tag{9}$$

while the performance was measured by the RMS error of the energy over the entire validation/test set

$$\mathrm{RMSE}(\tilde{E}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[\tilde{E}(\mathbf{x}_i) - y_i]^2}. \tag{10}$$

The performance on the validation set has been computed at every 128 gradient steps, and the final state of the NN has been taken as that corresponding to the minimum of these evaluations. The overall performance has been measured on the test set at the end of this procedure. Figure 2 shows the evolution of the objective function vs iteration number for both models, which have the same number of parameters and have been trained for eight epochs. (One epoch corresponds to one passage over the whole training set.)

The results of training these two NNs on $\mathbb{D}$ are presented in Table I, for the parameters corresponding to the best-case validation scenario of each model. The CNN performs better than the MLP, with a RMS error on the energy of 1.101 eV (5.095 meV/atom) vs 1.507 eV (6.978 meV/atom). These values are comparable to those reported by Behler *et al.* [21] for 64-atom crystalline silicon configurations at various pressures and temperatures (RMSE over the test set was reported to be of the order of 5–6 meV/atom), in spite of the fact that we have neglected the G_4 and G_5 symmetry functions to describe the atomic configurations. These results show that the use of convolutions provides a clear advantage over dense layers.

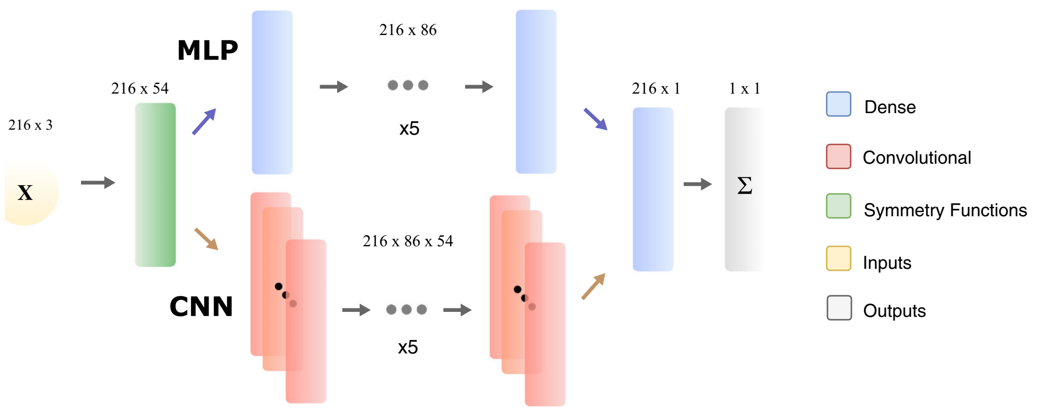

FIG. 1. Architecture of the MLP (top) and the CNN (bottom) used for learning the PES. Each block represents a neuron layer; the color indicates its type, and the dimensions of the output tensor are indicated. The first dimension corresponds to that of the minibatches: the corresponding elements are treated in parallel by the NN as different observations. The last dimension is the one where the operation is applied; the second dimension in the convolutional layers therefore corresponds to channels. The last layer performs a sum over the atomic energies and yields the total potential energy.

The performance of both NNs is quite satisfactory and both could certainly be used to approximate the SW potential. That being said, we insist that learning has been done here only on a-Si configurations, so that we cannot expect these models to transfer easily to other structures. The approach presented here is thus tailor-made for a-Si, but would easily transpose to other forms of silicon or other disordered materials inasmuch as it is possible to construct a proper input data set.

C. Model structures

We have established in the preceding section the relevance of representing the atomic configurations in terms of symmetry functions in spite of omitting the angular functions G_4 and G_5, and the advantage of using convolutional layers instead of fully connected ones. We will now exploit these observations

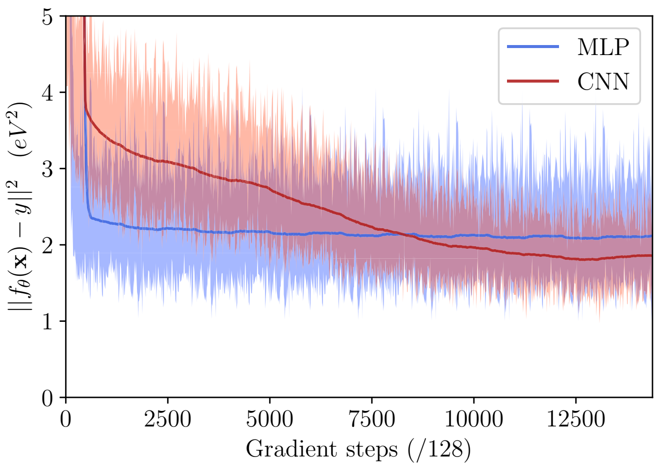

FIG. 2. Mean-square error (in eV^2) over 128 configurations between predicted and target energies during training for the two architectures considered, as indicated. The full lines are obtained by smoothing the data via a moving average.

for building a deep generative model for the atomic structure of a-Si.

We considered two architectures for this task. A Wasserstein GAN (WGAN) [40] was first considered, but while it succeeded in producing realistic structures, its performance was systematically poorer than the WAE, presented in Sec. II B. We will thus consider only the WAE in the following.

Since we are dealing with an unsupervised learning task, the data set has been separated in only two parts: a training set ($\sim$90%) and a validation set ($\sim$10%), the latter being used only to measure the distance between unseen configurations and generated configurations—see below.

It is certainly judicious to incorporate the potential energies contained in $\mathbb{D}$ to the model in order to fully exploit the available data. This can be done by conditioning the WAE over the potential energies $\mathbf{y}$, i.e., providing these to both the encoder and the decoder; their distributions then become $Q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and $G_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})$, respectively. While this is not essential—one may completely discard the potential energies and train the model only on the configurations—we observed that conditioning the model increased its accuracy.

1. WAE architecture

In order to evaluate the reconstruction cost of the WAE, we need the function $c(\mathbf{x}, \mathbf{x}')$ to be a suitable metric of the input space $\mathcal{X}$. Finding an appropriate way to measure a *distance* between two a-Si configurations is not *a priori* simple. Such a metric must be invariant under the system's symmetries, be reasonably fast to compute, and also be smooth enough to allow back-propagation of gradients. The symmetry functions $G_\mu(\mathbf{x})$ possess these properties naturally. The distance between two configurations can thus be taken as the Euclidean distance between their respective symmetry-function representations. As these are not invariant with respect to the permutation of atomic indices, we must sort the vectors

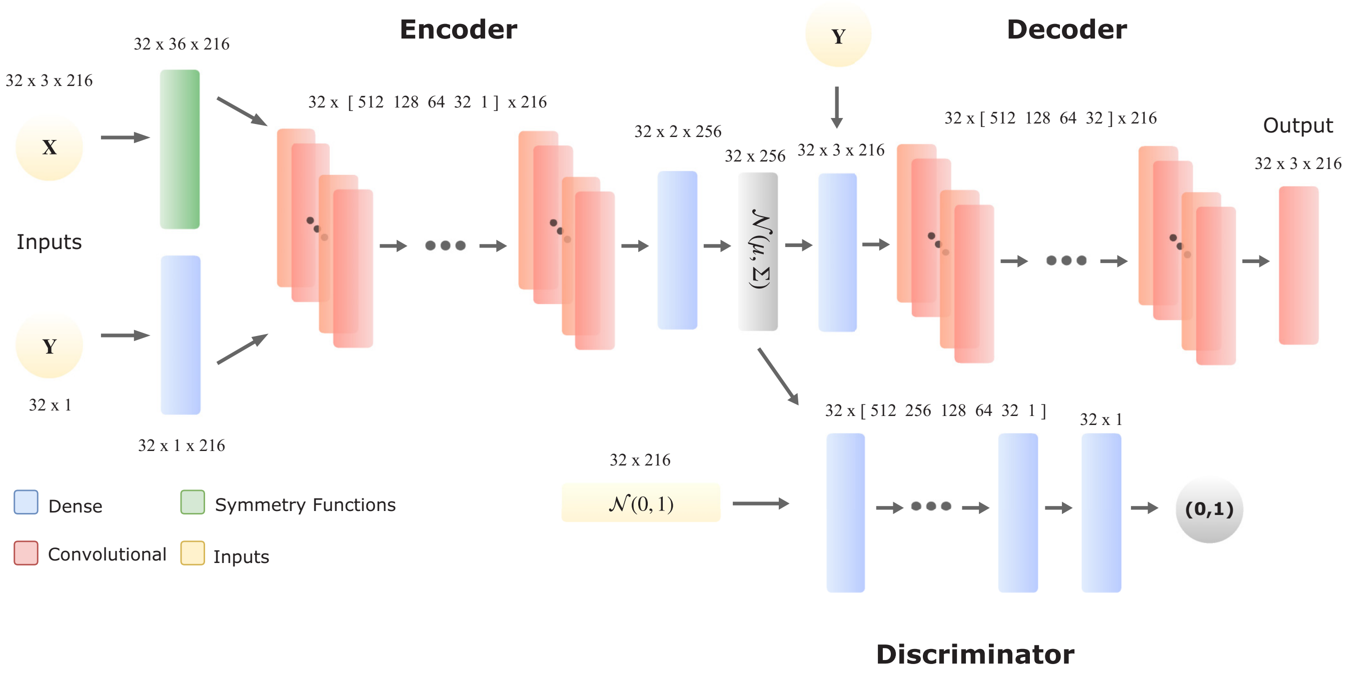

FIG. 3. Architecture of the WAE used to model the structure of *a*-Si. The color code and symbols are the same as in Fig. 1. The dimension of the output tensor of each layer is indicated. Inputs $\mathbf{x}$ and $\mathbf{y}$ stand for a minibatch of 32 data-set configurations and the corresponding total potential energies. Layers $\mathcal{N}(0, 1)$ and $\mathcal{N}(\mu, \Sigma)$ perform a draw from the normal distribution, where μ and Σ are the encoder's output vectors.

$G_\mu(\mathbf{x})$; thus we have

$$\begin{aligned} c(\mathbf{x}, \mathbf{x}') &= ||\mathcal{G}(\mathbf{x}) - \mathcal{G}(\mathbf{x}')||^2 \\ &= \sum_{i=1}^{N}\sum_{\mu=1}^{K}\left[\mathrm{sort}G_\mu^i(\mathbf{x}) - \mathrm{sort}G_\mu^i(\mathbf{x}')\right]^2. \end{aligned} \tag{11}$$

One can easily verify that this function is non-negative, symmetric with respect to its arguments, and vanishes when $\mathbf{u} \sim \mathbf{v}$ (where $\sim$ indicates the equivalence between two configurations). The objective functions of the WAE are thus given by Eq. (7), with the distance given by Eq. (11), and by feeding q_ϕ and g_θ with atomic configurations *and* potential energies. Figure 3 illustrates the architecture of the WAE used.

The autoencoder is essentially convolutional, as it allows the formation of deeper networks with fewer parameters and yields a better performance than dense layers for learning the PES. The encoder's first layers have the task of (i) calculating the symmetry functions associated with the atomic configurations and (ii) projecting the potential energies onto a vector. The output from these two layers are concatenated and fed into five convolutional layers. One last dense and linear layer, i.e., with identity activation function, is applied to produce the output vectors μ and Σ of the encoder. These are then used to produce a minibatch of stochastic latent vectors $\mathbf{z} = \mu + \epsilon\Sigma$ where $\epsilon \sim \mathcal{N}(0, 1)$. This procedure allows us to draw from $\mathcal{N}(\mu, \Sigma)$ while ensuring well-defined gradients with respect to μ and Σ, thus making it possible to backpropagate gradients through the encoder. As a technical note, we find that a stochastic (as opposed to deterministic) encoder is important to ensure the convergence of the model.

The decoder's input consists in the concatenation of a minibatch of latent vectors with a minibatch of potential energies. The first layer is dense and up-scales its input into a vector of size 3×216 that is subsequently reshaped and fed to five convolutional layers. The last layer of the decoder outputs reconstructed (generated) configurations during (after) training. In particular, it allows us to generate configurations that automatically satisfy periodic boundary conditions by using the following activation function:

$$\mathrm{PBC}(x) = (x - L)\mathrm{mod}(2L) - L, \tag{12}$$

where L is half the simulation box (8.141 Å here). The encoder and the decoder are constructed so as to be as symmetric and balanced as possible, a heuristic often used for ensuring more stable training.

The discriminator is an MLP with a number of parameters similar to that of the encoder and decoder; this allows for similar capacities of the different NNs involved. Finally, an approach used in Ref. [41] consists in helping the discriminator to discern the latent distribution of the model from $\mathcal{N}(0, 1)$, knowing that the optimal discriminator between the distributions $P_\mathcal{Z}$ and Q_ϕ is $D^* = \ln dP_\mathcal{Z}(\mathbf{z}) - \ln dQ(\mathbf{z}|\mathbf{x}, \mathbf{y})$. Since $p_\mathcal{Z}$ is known, the discriminator need only learn $\ln dQ(\mathbf{z}|\mathbf{x}, \mathbf{y})$, which can be done by subtracting the term

$$\ln dP_\mathcal{Z} = \frac{1}{2}\left(\ln(2\pi) + \sum_{i=1}^{\dim z} z_i^2\right) \tag{13}$$

from its output.

The WAE we consider here thus consists of (i) a stochastic encoder with 305 585 parameters formed of five convolutional layers and a linear, dense output layer; (ii) a deterministic decoder with 343 891 parameters, five convolutional layers and a dense input layer; and (iii) a discriminator with 306 179 parameters, six dense layers, and a sigmoid output. Unless

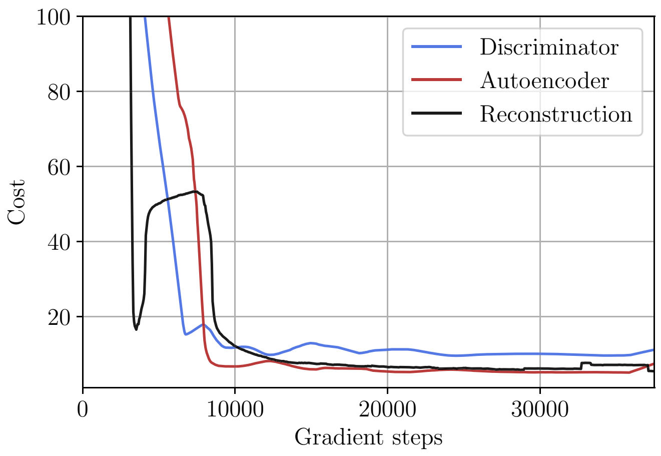

FIG. 4. Separate contributions to the training error of the WAE with respect to gradient iterations (unitless). The discriminator cost (blue) is computed from $\mathcal{L}_\text{D}$ in Eq. (6). The reconstruction cost corresponds to the first term of $\mathcal{L}_\text{AE}$ in Eq. (6) while the autoencoder cost corresponds to the second term, i.e., the adversarial part. The reconstruction cost has been evaluated on the validation set. The curves are obtained by smoothing the raw data via a moving average.

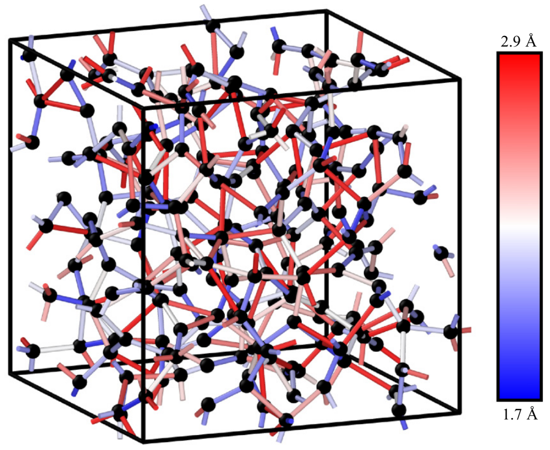

FIG. 5. Atomic configuration (of energy −302.38 eV) generated using the WAE algorithm. The color code indicates the distribution of distances in the range 1.7–2.9 Å. The image was produced using the OVITO software [42].

otherwise noted, all transfer functions are ReLUs. Thus, in total, the model contains 777 621 optimization parameters. It was trained over 20 epochs using the ADAM stochastic gradient algorithm with $\eta_{AE} = 5 \times 10^{-4}$, $\eta_D = 10^{-3}$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\omega = 10^{-8}$, $\lambda = 1$, and $\dim z = 216$. The gradient step was subject to the schedule

$$\eta(t) = \frac{\eta}{t^{1/4}}, \tag{14}$$

where t is the number of iterations in ADAM; this schedule turned out to be excellent for the convergence of the model.

The evolution of the different components of the objective function during the training phase is presented in Fig. 4. Training took 160 min on a Tesla K80 GPU. We note that training becomes stable after five epochs (one epoch = 1893 iterations), which means that the encoder has correctly learned to project the configurations from $\mathbb{D}$ onto a latent Gaussian representation. This is very important because it demonstrates the capacity of the encoder to properly decorrelate latent variables representing the system. In addition, if the encoder would fail, the decoder would not have learned to reconstruct configurations from a $\mathcal{N}(0, 1)$ latent vector, and thus to generate realistic configurations at the end of training.

The reconstruction error, evaluated over the validation set, decreases monotonically, apart from a few isolated peaks, then converges to a value comparable to the typical distance between configurations in $\mathbb{D}$. The generator is thus able to construct realistic configurations that are close, according to the distance measure in Eq. (11), to configurations it has never seen.

However, convergence heavily depends on the hyperparameters of the model, as poor choices lead to instabilities or divergences. For instance, a learning rate schedule that is too fast or too slow can prevent convergence by causing the system to either escape from local minima or struggling to reach it. Training is penalized if the first inertia factor β_1 is too large and, in practice, values in the range [0.5,0.7] are optimal. Likewise, β_2 should lie between 0.9 and 0.999. The gradient step of the autoencoder and the discriminator were found to be best in the range $[10^{-4}, 10^{-3}]$. Finally, a latent dimension $\dim z$ smaller than ∼200 leads to instabilities in the first few epochs, after which the model stabilizes and proper configurations can be generated down to $\dim z = 150$.

At the end of the training phase, the generative model consists of only the decoder, or generator, g_θ. The process for generating new configurations amounts to feeding into g_θ a minibatch of latent vectors $\mathbf{z} \sim \mathcal{N}(0, 1)$ as well as energies $\mathbf{y} \sim \mathcal{N}(\mu_E, \sigma_E^2)$, where μ_E and σ_E are the average and standard deviation for the whole data set, respectively.

2. *Models of a-Si*

Following the procedure described above, 1600 new configurations were generated, which took 6.1 s on a local machine. Figure 5 shows a selected, low-energy, configuration from this set. A visual inspection indicates that the configuration is already quite satisfactory—the structure is clearly disordered and has no major deficiencies, in particular no atom pairs at unrealistically short distances.

The average SW potential energy was −1.336 eV/atom, as compared to −3.130 eV/atom for the data-set configurations. However the distances between generated and input samples [as defined in Eq. (11)] are comparable to the distances between samples in $\mathbb{D}$. Similar distances are thus attributed to configurations that differ by as much as 1.8 eV/atom, suggesting that this criterion is not able to properly distinguish configurations with different structures. We suspect that this is a consequence of the neglect of the G_4 and G_5 symmetry functions, but further investigation is needed to ascertain this.

Thus, the model has converged up to the best of the G_1, G_2, and G_3 symmetry-function representation capability. We stress that no overfitting has taken place: the algorithm has properly learned, in an unsupervised manner, to generate configurations that are near the data-set ones, i.e., the generator is clearly tending towards the distribution $p_{\mathcal{X}}$.

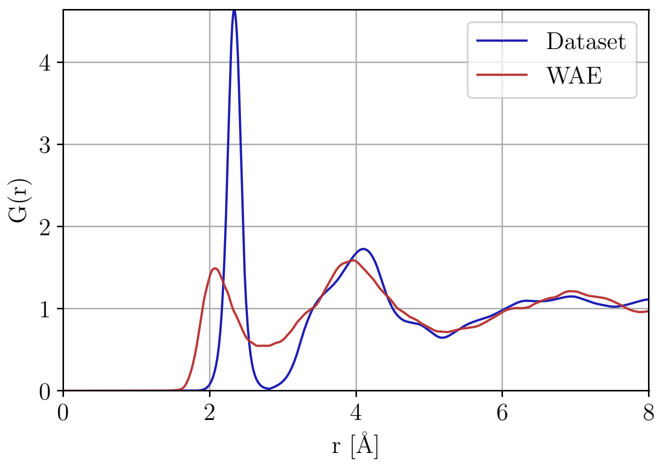

FIG. 6. Average radial distribution functions of generated samples (blue) and data-set samples (red), normalized by the number density (5.00×10^{-2} Å^{3}). The averages have been taken over 1600 samples (WAE) and 20 000 samples (data set), and the curves have been smoothed via a moving average.

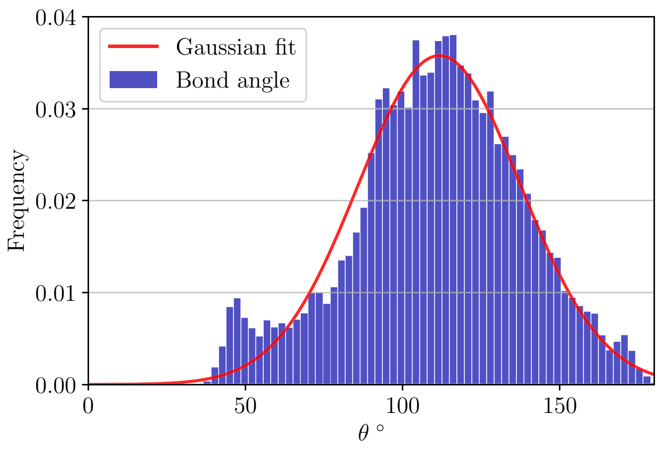

FIG. 7. Average nearest-neighbor bond-angle distribution over the 1600 generated configurations. The distribution is approximately Gaussian, while presenting a small peak at its low tail.

To be more quantitative, we present in Fig. 6 the average radial distribution function (RDF) of the generated and the input data-set configurations. We observe that the RDF presents a well-defined local order, as no atom pairs are found at distances shorter than 1.8 Å, confirming that the model has properly learned this crucial aspect of the structure of *a*-Si. The two RDFs coincide at large distances while we note a downward shift and broadening of the first peak for the generated structures; this is manifest of too strong a structural disorder, thus explaining the observed energy difference.

Nevertheless, we may set a nearest-neighbor cutoff distance r_c at the minimum between the first and second peaks of the RDF, yielding $r_c = 2.892$ Å. This can then be used to calculate various properties such as coordination numbers, average positions, and standard deviations of the RDF's peaks. These and other structural properties, averaged over the 1600 samples, are presented in Table II and compared with experimental data, average properties of the data set, and a state-of-the-art structure from [4].

The first-neighbor shell is centered at 2.323 Å which is slightly under the experimental value of 2.352 Å and the average value in the data set of 2.362 Å. The first coordination number $C_1 = 3.278$ is also below the experimental and data-set value, which are comparable. A bond-length deviation of 12.7% indicates excessive structural disorder, as noted above, when compared to the data-set value of 3.13%, which is close to the experimental value of 2.76%.

Figure 7 presents the average bond-angle distribution over the generated structures. It is centered at 109.67°, which is very near the crystalline value $\arccos(-\frac{1}{3}) = 109.47°$, but significantly too wide with a standard deviation of $\sigma = 27.387°$ vs 10–15° in traditional models, depending on the method used. Of course, the numbers depend on the particular choice of r_c which, as we have seen, is not well defined. We note also the presence of a small number of 60° angles, which reveals the presence of three-membered rings. (The latter are sometimes removed artificially from *a*-Si models as they are unrealistic, a procedure which is not possible in a automatic learning context.)

The second neighbor shell, here defined as the set of first neighbors to first neighbors, lies at 3.73 Å with a standard deviation of 0.654 Å and a coordination $C_2 = 8.560$. These values indicate an undercoordination of the shell and a strong disorder in bond angles, as seen previously in Fig. 7.

3. Discussion

Overall, our model yields configurations presenting a structure which is qualitatively similar to that expected while possessing too much disorder. The WAE exhibits a stable learning curve after five epochs, and the distance between

TABLE II. Average structural properties of (i) configurations generated by our model (WAE), (ii) the input data set ($\mathbb{D}$), (iii) a 4096-atom model of [4] (BM), and (iv) experimental data of Laaziri *et al.* [2] (Expt.). Presented are the density ρ, the coordination numbers C_1 and C_2, the first and second peak positions, and standard deviations r_1 and r_2 and σ_1 and σ_2. The SW total potential energy E and its standard deviation σ_E are also presented when available.

	ρ (g/cm^3)	C_1 (at.)	r_1 (Å)	σ_1 (Å)	C_2 (at.)	r_2 (Å)	σ_2 (Å)	E (eV)	σ_E (eV)
WAE	2.334	3.278	2.323	0.297	8.560	3.73	0.654	−288.56	13.30
$\mathbb{D}$	2.334	3.926	2.362	0.074	11.97	3.82	0.206	−676.15	12.016
BM	2.450	4.000	2.397	0.075	12.00	3.74	0.272		
Expt.	2.285	3.881	2.352	0.065	12.43	3.81	0.238		

generated and data-set configurations decreases monotonically in this regime, down to the typical distance between data-set configurations. This indicates that the model has converged, to the best of the capabilities of the given distance measure, towards the data set's distribution. We believe that the lack of angular description (which cannot easily be resolved computationally) provided in the symmetry-function representation, and the resulting distance, to be the cause of the discrepancies between generated and data-set configurations.

Some other limitations may be invoked. First, convergence of the learning process is very sensitive to the choice of hyperparameters. Generative models are known to be less stable than predictive models, which is why unsupervised learning is notably more difficult than supervised learning. Second, the model is not invariant with respect to the number of particles, i.e., it cannot learn from, or generate structures of, different sizes—a feature we believe is achievable in principle and constitutes an interesting research avenue.

Nevertheless, we have demonstrated that it is feasible to train a WAE to generate structures that are qualitatively close to those expected for *a*-Si, based on a learning data set of representative configurations and the corresponding potential energies, using the Euclidean distance between the symmetry functions for the reconstruction cost.

IV. CONCLUSION

In summary, we have trained a deep generative model, the Wasserstein autoencoder, to generate *a*-Si configurations that feature a *qualitatively* correct structure although presenting excessive disorder. The model was trained on a data set generated by means of ART and MD simulations. Configurations were represented by symmetry functions, a mapping that also served to define a suitable distance between samples. We validated this approach by first training a convolutional neural network to predict the SW potential energies, which featured a very good performance, comparable to that of previous work. Next, the WAE has been trained. The average RDF of the generated samples has been analyzed, showing that the key features of *a*-Si are captured. Even if the lack of angular resolution in the symmetry function representation was not problematic for learning the PES, it seems to be the case for the generative task. And indeed, generative modeling is notoriously more demanding than predictive modeling.

The present work establishes the viability of the DL approach for generating model structures of disordered systems. The method clearly requires improvements, however; when implemented, we expect that it may lead to deep generative models combining both realism and efficiency, especially when trained, e.g., on *ab initio* input structures, thus removing the human bias encountered in current models. Moreover, the approach presented here paves the way to a general framework for overcoming size limitations inherent to computationally demanding techniques for simulating the structure of disordered systems.

ACKNOWLEDGMENTS

We are grateful to N. Mousseau for useful discussions and ART models, and to A. Courville for valuable advice on deep learning. This work was supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) Grant No. RGPIN-2015-06682 and the Fonds de Recherche du Québec - Nature et Technologies (FRQ-NT). We are indebted to Calcul Québec and Calcul Canada for generous allocations of computer resources. The calculations have been done on the Helios cluster of Université Laval, overwatched by Calcul Québec and Compute Canada. The exploitation of this supercomputer was funded by the Canada Foundation for Innovation (CFI) and the Ministère de l'Économie et de l'Innovation du Québec (MEI).

[1] R. Zallen, *The Physics of Amorphous Solids* (John Wiley & Sons, New York, 2008).
[2] K. Laaziri, S. Kycia, S. Roorda, M. Chicoine, J. L. Robertson, J. Wang, and S. C. Moss, High-energy x-ray diffraction study of pure amorphous silicon, Phys. Rev. B **60**, 13520 (1999).
[3] F. Wooten, K. Winer, and D. Weaire, Computer Generation of Structural Models of Amorphous si and ge, Phys. Rev. Lett. **54**, 1392 (1985).
[4] G. T. Barkema and N. Mousseau, High-quality continuous random networks, Phys. Rev. B **62**, 4985 (2000).
[5] G. T. Barkema and N. Mousseau, Event-Based Relaxation of Continuous Disordered Systems, Phys. Rev. Lett. **77**, 4358 (1996).
[6] N. Mousseau and G. T. Barkema, Traveling through potential energy landscapes of disordered materials: The activation-relaxation technique, Phys. Rev. E **57**, 2419 (1998).
[7] R. Malek and N. Mousseau, Dynamics of lennard-jones clusters: A characterization of the activation-relaxation technique, Phys. Rev. E **62**, 7723 (2000).
[8] F. H. Stillinger and T. A. Weber, Computer simulation of local order in condensed phases of silicon, Phys. Rev. B **31**, 5262 (1985).
[9] R. Vink, G. Barkema, W. van der Weg, and N. Mousseau, Fitting the stillinger–weber potential to amorphous silicon, J. Non-Cryst. Solids **282**, 248 (2001).
[10] P. Hohenberg and W. Kohn, Inhomogeneous electron gas, Phys. Rev. **136**, B864 (1964).
[11] W. Kohn and L. J. Sham, Self-consistent equations including exchange and correlation effects, Phys. Rev. **140**, A1133 (1965).
[12] A. Pedersen, L. Pizzagalli, and H. Jónsson, Optimal atomic structure of amorphous silicon obtained from density functional theory calculations, New J. Phys. **19**, 063018 (2017).
[13] J. Tersoff, New empirical approach for the structure and energy of covalent systems, Phys. Rev. B **37**, 6991 (1988).
[14] J. Tersoff, Empirical interatomic potential for silicon with improved elastic properties, Phys. Rev. B **38**, 9902 (1988).

[15] N. Artrith, A. Urban, Y. Wang, and G. Ceder, Atomic-scale factors that control the rate capability of nanostructured amorphous Si for high-energy-density batteries, arXiv:1901.09272.

[16] V. L. Deringer, N. Bernstein, A. P. Bartók, M. J. Cliffe, R. N. Kerber, L. E. Marbella, C. P. Grey, S. R. Elliott, and G. Csányi, Realistic atomistic structure of amorphous silicon from machine-learning-driven molecular dynamics, J. Phys. Chem. Lett. **9**, 2879 (2018).

[17] M. A. Caro, V. L. Deringer, J. Koskinen, T. Laurila, and G. Csányi, Growth Mechanism and Origin of High sp^3 Content in Tetrahedral Amorphous Carbon, Phys. Rev. Lett. **120**, 166101 (2018).

[18] N. Bernstein, B. Bhattarai, G. Csányi, D. A. Drabold, S. R. Elliott, and V. L. Deringer, Quantifying chemical structure and machine-learned atomic energies in amorphous and liquid silicon, Angew. Chem. **131**, 7131 (2019).

[19] D. Igram, B. Bhattarai, P. Biswas, and D. A. Drabold, Large and realistic models of amorphous silicon, J. Non-Cryst. Solids **492**, 27 (2018).

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT, Cambridge, MA, 2016).

[21] J. Behler and M. Parrinello, Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces, Phys. Rev. Lett. **98**, 146401 (2007).

[22] J. Behler, Atom-centered symmetry functions for constructing high-dimensional neural network potentials, J. Chem. Phys. **134**, 074106 (2011).

[23] J. Behler, Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations, Phys. Chem. Chem. Phys. **13**, 17930 (2011).

[24] T. M. Mitchell, *Machine Learning*, 1st ed. (McGraw-Hill, New York, 1997).

[25] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[26] T. Mueller, A. G. Kusne, and R. Ramprasad, Machine learning in materials science: Recent progress and emerging applications, *Reviews in Computational Chemistry*, edited by A. L. Parrill and K. B. Lipkowitz, Vol. 29 (John Wiley & Sons, Inc., New York, 2016), pp. 186–273.

[27] A. P. Bartók, S. De, C. Poelking, N. Bernstein, J. R. Kermode, G. Csányi, and M. Ceriotti, Machine learning unifies the modeling of materials and molecules, Sci. Adv. **13**, e1701816 (2017).

[28] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, Object recognition with gradient-based learning, in *Shape, Contour and Grouping in Computer Vision* (Springer-Verlag, London, 1999), p. 319.

[29] D. H. Hubel and T. N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, J. Physiol. **195**, 215 (1968).

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (MIT, Cambridge, MA, 1986), Vol. 1, pp. 318–362.

[31] P. L. Reference, Python Software Foundation. Available at http://www.python.org, version 3.5.3.

[32] Theano Development Team, Theano: A Python framework for fast computation of mathematical expressions, arXiv:1605.02688.

[33] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri *et al.*, Lasagne: First release, Aug. 2015.

[34] D. P. Kingma and M. Welling (unpublished).

[35] A. Radford, L. Metz, and S. Chintala (unpublished).

[36] C. Villani, *Topics in Optimal Transportation*, Graduate Studies in Mathematics, Vol. 58 (American Mathematical Society, Rhode Island, 2003).

[37] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Vol. 2 (MIT Press, Cambridge, MA, USA, 2014), pp. 2672–2680.

[38] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[39] I. Loshchilov and F. Hutter, Fixing weight decay regularization in adam, arXiv:1711.05101.

[40] M. Arjovsky, S. Chintala, and L. Bottou (unpublished).

[41] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf (unpublished).

[42] A. Stukowski, Visualization and analysis of atomistic simulation data with ovito the open visualization tool, Modell. Simul. Mater. Sci. Eng. **18**, 015012 (2010).