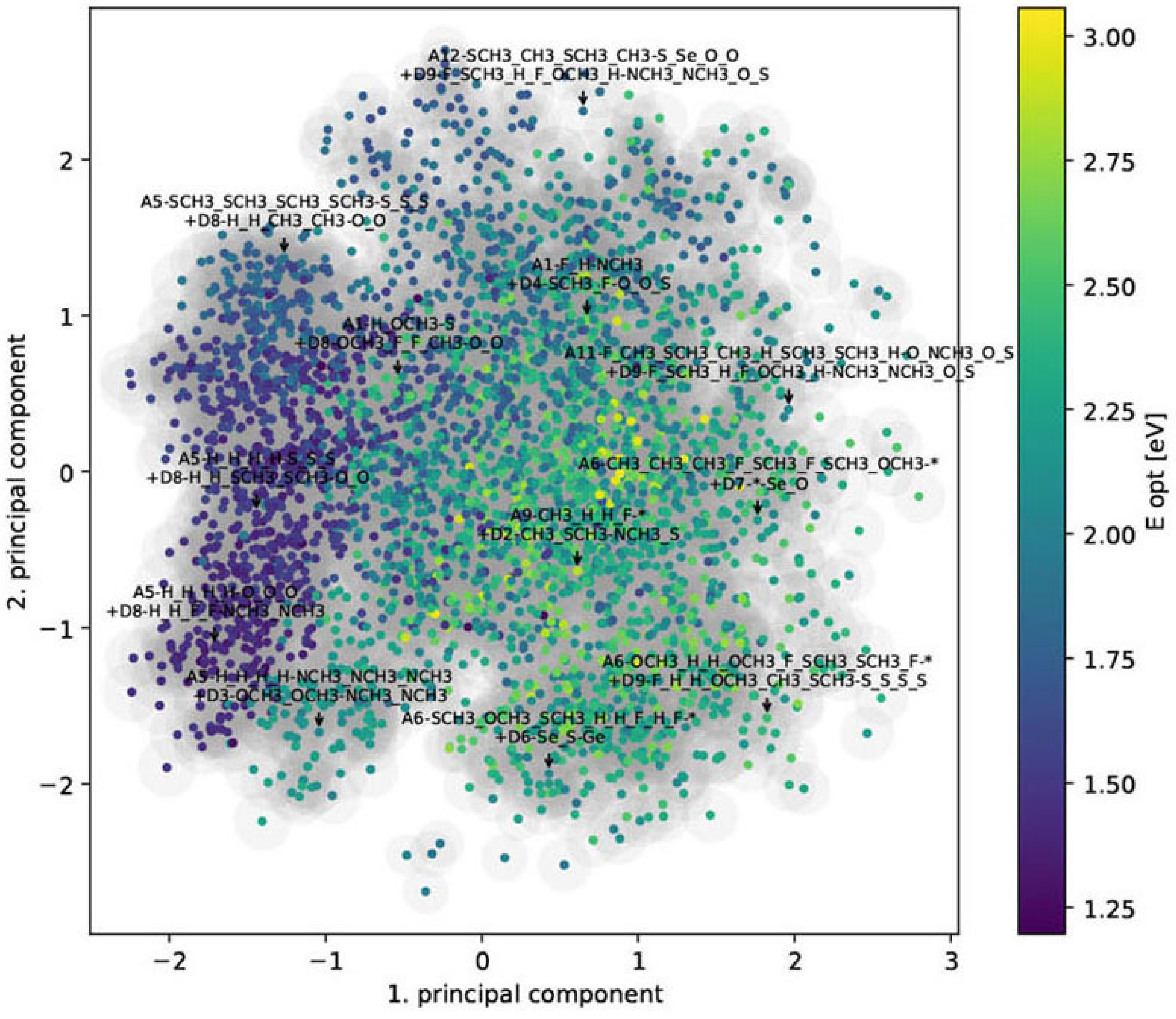
DOI: 10.1002/minf.201700133

Deep Generative Models for Molecular Science

Peter B. Jørgensen,[a] Mikkel N. Schmidt,[a] and Ole Winther*[a]



Abstract: Generative deep machine learning models now rival traditional quantum-mechanical computations in predicting properties of new structures, and they come with a significantly lower computational cost, opening new avenues in computational molecular science. In the last few years, a variety of deep generative models have been proposed for modeling molecules, which differ in both their model structure and choice of input features. We review these recent advances within deep generative models for predicting molecular properties, with particular focus on models based on the probabilistic autoencoder (or variational autoencoder, VAE) approach in which the molecular structure is embedded in a latent vector space from which its properties can be predicted and its structure can be restored.

Keywords: molecular science · deep learning · variational inference · variational auto-encoders · generative modeling

1 Introduction

Computational molecular science – the discovery and design of new molecules and analysis of their structure and properties by computer models – has traditionally involved making elaborate quantum-mechanical computations derived from first principle. In recent years, new approaches based on machine learning have shown great promise, approaching the same accuracy as first principle computations at a much lower computational cost. Machine learning is a branch of artificial intelligence concerned with making models that can learn from data by discovering patterns in the data, and generalizing these patterns to new unseen cases. In molecular science, machine learning can leverage the existing huge databases of experimental results and quantum-mechanical calculations that are currently available, to learn to predict properties and structures of new molecules at unrivaled computational speed. In this review we outline the current trends in machine learning-based computational molecular science with a particular focus on one of the most promising model classes known as *deep generative models*. Based on the latest results from the literature and from our own research, our aim is to characterize the deep generative modeling paradigm in terms of both model structure and approach to inference, in order to build intuition about the mechanisms behind its success.

In machine learning we distinguish between discriminative and generative models. In a *discriminative* learning approach to molecular science, we would be concerned with learning a mapping from a molecule x to a property y that we are interested in predicting. Given a dataset $\{x_i, y_i\}_{i=1}^N$ that consists of N molecules and their corresponding known properties, the discriminative model is targeted at learning the probability distribution $p(y|x)$ such that predictions for a new material x^* can be computed. Although this can lead to excellent predictions, the downside of the discriminative approach is that the model does not describe the molecular structure x itself – only the relation between the structure and property – and can thus not directly be used to make inference about new molecules of interest. In a *generative* learning approach, the model is concerned with characterizing either the distribution of the molecular structure $p(x)$ or the joint distribution $p(x, y)$ of the molecular structure and the property of interest. The former case where only the molecular structure is modeled is known as *unsupervised* learning, whereas the latter case where both molecular structure and corresponding properties are modeled is known as *supervised* learning.

In this study we will focus on one particular non-linear flexible generative model, the *variational autoencoder (VAE)*.[1,2] The complementary likelihood free approach to generative modelling, *generative adversial networks (GAN)*, has also received a lot of attention recently. GAN and other approaches beyond VAE are discussed in Section 6. In our review of VAE we include recent advances in inference and implementation details. Therefore the tutorial by Doersch[3] might be an easier starting point to establish the mathematical intuition behind the model.

The remainder of the paper discusses generative modelling (Section 2), variational inference (Section 3), practical implementation (Section 4) and application to properties of molecules (Section 5).

2 Deep Generative Models

In this paper we define a generative model as a probabilistic model for the data features we have access to. We distinguish between unsupervised learning where we have a feature vector x and supervised learning where we have x and dependent variables y. For unsupervised learning the generative model is the joint distribution of x: $p(x)$. Probabilistic supervised learning involves modeling the conditional $p(y|x)$. In generative probabilistic supervised learning we decompose the joint distribution of x and y as $p(x, y) = p(y|x)p(x)$. In the molecular context x will usually be some representation of the molecule structure (for example the SMILES representation,[4] molecular fingerprints such as MACCS[5] or in principle the atomic positions and properties) and y will be a physical property of the system such as free energy, ground state energy, band gap, or crystal structure

[a] *P. B. Jørgensen, M. N. Schmidt, O. Winther*
Technical University of Denmark
E-mail: olwi@dtu.dk

or a biological response such as toxicity, cellular uptake, or drug efficacy.

The two main reasons why we want to apply probabilistic generative models are that they:

1. Allow for quantitative model comparision. Once fitted we can evaluate the density on a test data point x_{test} or a test dataset. A high value of log $p(x_{\text{test}})$ (relative to other model baselines) indicates that the model has captured essential statistical properties of x.
2. Can be used to synthesize new data. Once fitted we can simulate new data from the model $x_{\text{new}} \sim p(x)$ where ~ is notation for a draw from the distribution. The synthesised data can be used for a qualitative evaluation of the model for example an image generated from a model fitted on a training set of natural images. The generated data can also be of practical interest, for example a new molecule with potentially desirable properties.

In this paper we will consider a specific class of generative models called latent variables models. A latent variable (vector) z represents unobserved properties of the datum that can describe the observed datum x statistically through a generative process $p(x|z)$. A simple example of such a model is a linear model with additive noise ε: $x = Wz + \varepsilon$, where W is the weight matrix. Often we will assume that $\dim(x) > \dim(z)$ so that the latent representation is more compact than the observed data. The latent variable itself is assumed to be generated from a prior distribution $p(z)$. The generative model $p(x)$ discussed above is recovered by marginalizing over $z : p(x) = \int p(x|z)p(z)dz$. In a latent variable model we can draw from $p(x)$ by a two-step procedure: 1) $z \sim p(z)$ and 2) $x \sim p(x|z)$. In some cases the latent variable will have a direct physical meaning and in other cases the use of a latent variable model is a convenient way to define a flexible statistical model.

We will usually use maximum likelihood learning. For unsupervised learning this means that we have a training set $X = \{x_i\}_{i=1}^N$ that we model with a latent variable model $p_\theta(x, z) = p_\theta(x|z)p(z)$ with parameters θ. We will as a default assume independent identically distributed (iid) samples such that the likelihood for θ is written as

$$p_\theta(x) = \prod_{i=1}^{N} p_\theta(x_i) \tag{1}$$

with $p_\theta(x_1) = \int p_\theta(x_i|z)p_\theta(z)dz$. Sometimes we will omit the θ dependence for brevity. In maximum likelihood learning the objective is thus to maximize (1) with respect to the parameters θ.

In deep generative modelling we replace the simple linear relation between the observed data x and the latent variable z with a parameterized non-linear function $f_\theta(\cdot)$. For $f_\theta(\cdot)$ we use a multi-layered neural network with L layers of adaptable weights, for example

$$f_\theta(z) = W_L h_{L-1} + b_L, \tag{2}$$

$$h_l = \text{relu}(W_l h_{l-1} + b_l),\ l = 2, \ldots, L-1 \tag{3}$$

$$h_1 = \text{relu}(W_1 z + b_1), \tag{4}$$

where the (element-wise) rectified linear activation function is given by $\text{relu}(a) = \max(0, a)$. We then have

$$p_\theta(x|z) = N(f_\theta(z), \sigma^2 I). \tag{5}$$

The trainable parameters θ of the model are the L weight matrices $W_1, \ldots W_L$, the L bias vectors $b_1, \ldots b_L$ and the output noise variance σ^2. The model can be extended, for example by letting σ^2 depend on the latent variable z or by introducing a hierarchy of latent variables, for example $p(x, z_1, z_2) = p(x|z_1)p(z_1|z_2)p(z_2)$ where z_1 and z_2 are latent variables vectors.

Peter B. Jørgensen received his B.Sc. (2011) in Electronics Engineering and IT and his M.Sc. (2013) in Wireless Communication Systems from Aalborg University, Denmark. He is currently working on his Ph.D. in Machine Learning at Technical University of Denmark. PBJs research interests includes (variational) Bayesian inference, statistical signal processing and deep learning.

Mikkel N. Schmidt is Associate Professor at DTU Compute, Technical University of Denmark. He is interested in probabilistic modeling and statistical machine learning with applications in both science and industry. His primary focus is the development of computational procedures for inference and validation.

Ole Winther is Professor of Data Science and Complexity at the DTU Compute, Technical University of Denmark. OWs Research interests include general machine learning methodology, deep generative modeling and applications of machine learning in health- and bioinformatics, energy and natural language.

As soon as we introduce non-linearities in the model we can not analytically marginalize out the latent variables. We therefore have to resort to approximations. A prominent method, that variational autoencoders are based on, is variational inference.

3 Variational Inference

The approach we use in variational autoencoders (VAE) consists of three steps:

Likelihood lower bound. Replace the likelihood with a more tractable lower bound. The most widely used is:

$$\log p(x_i) \geq E_{q_i(z)}\left[\log \frac{p(x_i|z)p(z)}{q_i(z)}\right], \tag{6}$$

where $E_{q_i(z)}[\ldots]$ denote average (expectation) over $q_i(z)$. We have here introduced a variational distribution $q_i(z)$ which is an approximation to the posterior distribution over the latent variables and its role in the variational autoencoder is explained in the next paragraph. This lower bound still requires intractable integration due to $E_{q_i(z)}[\ldots]$, but we can derive low variance Monte Carlo estimators for these. The bound can be decomposed into two terms

$$\log p(x_i) \geq E_{q_i(z)}[\log p(x_i|z)] - KL(q_i(z), p(z)), \tag{7}$$

where $KL(q(x), p(x)) \equiv \int q(x) \log \frac{q(x)}{p(x)} dx$ is the Kullback-Leibler divergence between q and p. The first term on the right-hand-side of (7) can be interpreted as the average reconstruction error, i.e. how well does the generative model fit the data distribution and the KL-term is a measure of how much $q(z)$ diverges from the prior $p(z)$.

An alternative tighter bound is given by Burda et al.,[6] which is often used when comparing different models on the same dataset.

$$F_K(x) \equiv \int \prod_{k=1}^{K} q(z_k) \log \left[\frac{1}{K}\sum_{k'=1}^{K} w_x(z_{k'})\right] \prod_{k=1}^{K} dz_k$$

with $w_x(z) \equiv \frac{p(x|z)p(z)}{q(z)}$. This so-called importance weighted bound coincides with the standard bound for $K=1$, obeys $F_K \geq F_L$ for $K > L$ and converges to the log $p(x)$ for $K \to \infty$ as a consequence of the law of large numbers. It is often used with moderate K for example $K \approx 3$–10 during training and large K, $K \approx 1000$–5000 for evaluating the test log likelihood.

Inference network. The lower bound (6) depends upon a variational distribution $q_i(z)$ which is an approximation to the latent posterior distribution

$$p(z|x_i) = \frac{p(x_i|z)p(z)}{p(x_i)}. \tag{8}$$

We can for example choose $q_{\phi_i}(z) = N(z|\mu_i, \sum_i)$ with variational parameters $\phi_i = \{\mu_i, \Sigma_i\}$. In VAE, instead of having a set of parameters for each x_i we will use a socalled inference network parameterization $q_\phi(z|x)$. The inference network will be specified as a deep neural network similar to the generative model but now with x as input and mean and variance of z as output. We could for example take $q_\phi(z|x) = N\ (z|\mu_\phi(x), \Sigma_\phi(x))$ where $\mu_\phi(x)$ and $\Sigma_\phi(x)$ are written as output of a deep network in the same fashion as the generative model was specified in the generative model Eqs. (2)–(5). This is actually more restrictive than the formulation with individual variational parameter for each datum but it has the advantage that we leverage information across data points. In other words it is based upon the assumption that data points that are close (in some sense derived from the data) will also have similar posterior latent distribution. The log likelihood lower bound for the training set that we want to optimize is:

$$L(\theta, \phi) = \sum_{i}^{N} E_{q_\phi(z|x_i)}\left[\log \frac{p_\theta(x_i|z)p_\theta(z)}{q_\phi(z|x_i)}\right]. \tag{9}$$

Note that even though we treat the two sets of parameters θ and ϕ in the same way they play different roles: the θ-optimization is model fit and the ϕ-optimation is for making the bound as tight as possible. Overfitting is not an issue in the latter case – we want the inference network to come as close as possible to $p(z|x)$.

The inference network $q_\phi(z|x)$ can be interpreted as a probabilistic encoder which maps an input x to a probability distribution in the latent vector space and $p_\theta\ (x|z)$ is the corresponding probabilistic decoder. Hence the name *variational autoencoder.*

Monte Carlo estimators and reparameterization trick. Low variance Monte Carlo estimators of the log likelihood lower bound and its derivative with respect to the parameters θ and ϕ are obtained by choosing a parameterization of $q_\phi(z|x)$ that allows the use of the so-called reparameterization trick. Again we focus on the standard bound and use M samples:

$$\log p(x) \geq E_{q_\phi(z|x)}[\log w_x(z)] \approx \frac{1}{M}\sum_{m=1}^{M} \log w_x(z^m)$$

with $w_x(z) = \frac{p_\theta(x|z)p_\theta(z)}{q_\phi(z|x)}$ and $z^m \sim q(z|x)$. Since we expect $w_x(z)$ to scale exponentially in the number of dimensions of z then the logarithm appearing in the bound is important to make the Monte Carlo estimator (the right hand side above) have low variance.

We need the reparameterization trick when taking derivatives. For the Gaussian inference network it amounts to replacing an average over $q_\phi(z|x)$ with an average over $\varepsilon \sim N(0, I)$. The parameter dependence is thus shifted into the integrand. In the integrand we will then replace z with $z_\phi(\varepsilon, x) = \mu_\phi(x) + \sqrt{\sum_\phi(x)}\varepsilon$ where $\sqrt{\sum_\phi(x)}$ is to be understood as a matrix square root. A diagonal $\Sigma_\phi(x)$ is the standard choice because it is simple only requiring a network with dim(z) outputs for the (log) variances.

3.1 Bits Back

The bits back argument[7,8,9] gives theoretical insight on why variational models often will converge to a solution where relatively few latent dimensions are being used. That is in $p(x|z)$ many of the components of z have no influence on x. The bits back back argument is simply an alternative decomposition of the log likelihood bound in the limit of very large training set corresponding to averaging over the data generating distribution $p_{\text{data}}(x)$. For the variational bound using an inference network we have

$$E_{p_{data}(x)}[\log p(x)] \geq E_{p_{data}(x),q(z|x)}\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right].$$

We can rewrite this expression using two steps: multiplying by $\frac{p_{data}(x)}{p_{data}(x)}$ inside the log and replacing $p(x|z)p(z)$ by the equivalent $p(z|x)p(x)$:

$$\begin{aligned} E_{p_{data}(x)}[L] = {} & -H(p_{data}) - KL(p_{data}(x), p(x)) \\ & -E_{p_{data}(x)}[KL(q(z|x), p(z|x))], \end{aligned}$$

where $H(p) = -\int p(x) \log p(x) dx$ is the entropy of p. The first term (minus the entropy of the data generating distribution) is the irreducible lower bound on the log likelihood, the second is the model error and the third is the variational posterior approximation error. The trade-off between the two last terms determines the solution we will find. We get a more flexible model by introducing latent variables because $p(x) = \int p(x|z)p(z)dz$ will in general be more flexible than a model without the latent variables but we pay a price in terms of the variational approximation error. The components $\hat{z}$ of z not are used in the generative model will have $p(\hat{z}|x) = p(\hat{z})$ and we can get $KL(q(\hat{z}|x), p(\hat{z}|x))$ to be zero by setting $q(\hat{z}|x) = p(\hat{z})$. The limited number of active units observed empirically, see for example,[10] reflects this trade-off. At some point introducing additional latent variables gives a smaller gain in terms of model fit than the price paid in approximation error. This is motivation for using the improved variational approximations discussed in the text section.

3.2 Improving the Variational Approximation

Methods for improving the variational distribution mayroughly be divided into three categories:

Hierarchical specification of variational distribution. For generative models with two or more stochastic layers there is some freedom in choosing the connectivity of variational distribution. Consider a two layer generative model $p(x, z_1, z_2) = p(x|z_1)p(z_1|z_2)p(z_2)$. One minimal solution is to specify q with dependence in the reverse order: $q(z_1, z_2|x) = q(z_2|z_1)p(z_1|x)$. However, we get a more flexible and thus more accurate inference network if we condition the first term on x as well: $q(z_1, z_2|x) = q(z_2|z_1, x)p(z_1|x)$. We say that we introduce a skip-connection that connects x directly to the network for the mean and covariance of z_2: $\mu_\phi(z_1, x)$ and $\Sigma_\phi(z_1, x)$. The ladder VAE[10] has a more advanced version of skip connections which also includes a parameterization where the parts of the inference network are shared by the prior hierarchical prior specification for the generative model. This leads to improved generative performance.

Normalizing flows. Let f be an invertible dim(z)→dim(z) mapping. We can use distribution $z' = f(z)$ in our generative model instead of z and introduce the determinant of the Jacobian in the likelihood.[11] If we let f have adaptable parameters it is possible to learn quite flexible priors. We can also generalize this concept in a simple way by letting f be a series on invertivle transformations.[12,13]

Auxiliary latent variables. In the auxiliary variable approach[14,15,16] we introduce a new latent vector a into the inference network $q(z, a|x) = q(z|a, x)q(a|x)$. We specify the generative model as $p(x, z, a) = p(x|z)p(z)p(a|z, x)$ in order to leave the generative process unaffected that is z and x are not affected by the value of a. However, the resulting marginal inference network $q(z|x) = \int q(z|a, x)q(a|x)da$ is now more flexible than before and should therefore give a better fit to the posterior. This is also observed in practice.[15] One may also make a bits back argument for the auxiliary model that shows that we still have to make a trade-off between the exact posterior $p(a|z, x)$ and its variational approximation.

4 VAE Inference in Practice

Variational autoencoder practice follows standard modern deep learning practice. Mini-batch stochastic gradient descent optimization is employed with parameter updates based upon gradients calculated on mini-batches of the order of 100 training examples. Thousands of epochs (complete sweeps through the training set) are often needed for convergence. The objective is usually evaluated with $M = 1$ samples for the expectation:

$$\sum_{i}^{\text{mini-batch}} \log \frac{p_\theta(x_i|z_i)p_\theta(z_i)}{q_\phi(z_i|x_i)}$$

with $z_i \equiv z_\phi(\varepsilon_i, x_i) = \mu_\phi(x_i) + \sigma_\phi(x_i) \otimes \varepsilon_i$, $\mu_\phi(x_i)$ and $\log \sigma_\phi(x_i)$ are the output of the inference network (both dim(z) dimensional), $\otimes$ denotes component-wise multiplication and $\varepsilon_i \sim N(\varepsilon|0, I)$. Setting $M = 1$ is the favored choice because averaging over mini-batches will be a more efficient way to decrease the variance of the estimator than to use the same example $M > 1$ in the same mini-batch. Many variants of adaptive step-size stochastic gradient descent with momentum have been proposed recently. Adam[1] is a popular choice. Differentiation of the objective involves applying the chain rule of differentiation. This is performed automatically within modern deep learning software packages such as TensorFlow and PyTorch. Some derivate terms in the objective cancels exact on expectation.[17] These terms have to removed explicitly from the gradient calculation. In practice it turns out that it does not make a big difference to take this into account or not.[17]

The effect of removal of latent variables discussed in Section 3.1 can be partially mitigated by employing "warm-up"[10,18] or the "free bits" surrogate objective.[13] We rewrite the log likelihood lower bound objective in (9) and introduce a "temperature" parameter α:

$$L(\theta,\phi,\alpha)=\sum_{i}^{N}E_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)+\alpha\log\frac{p_\theta(z)}{q_\phi(z|x_i)}\right]. \qquad (10)$$

The original objective is restored by setting $\alpha=1$. Warmup[10,18] amounts to starting with a traditional map *x*-to-*x* autoenocder objective corresponding to $\alpha=0$ and slowly increasing α, say linearly during the first 200 epochs, to the variational objective $\alpha=1$. In some cases for example in the sequence encoder and decoder models employed in Section 5 it turns out that it is necessary to set the final value of α below one in order to make the model use any latent variables. The learned latent structure is still useful but the likelihood lower bound interpretation is no longer valid.

5 Applications to Molecular Sciences

Using VAEs with recurrent neural network encoder and decoder was first employed by Bowman et al.[18] to generate English written sentences. The first application in molecular science is demonstrated by Gómez-Bombarelli et al.,[19] where a similar encoder and decoder model is used to generate SMILES strings character by character from the latent space. The method is applied to a dataset of approximately 250,000 drug-like molecules from the ZINC database[20] and 100,000 organic light emitting diode (OLED) molecules.[21] One of the problems of the character-based VAE is that it often produces invalid molecules. In the experiments by Gómez-Bombarelli et al.[19] from 70% to less than 1% of the generated samples are valid molecules. The errors can be syntax errors (the generated string is not a valid SMILES string) or semantic errors (the SMILES string is syntactically valid but the molecule corresponding to the SMILES string is physically impossible). Despite these problems, using SMILES strings is an appealing approach because it gives a full description of the molecule without making assumptions about which features are important for the task at hand. If we for example modeled fingerprints we would have to search for the molecule that corresponds to the generated fingerprint (which might not exist if it is generated erroneously) and we have already made a choice of which features are important by selecting or designing the fingerprint.

5.1 Using a Grammar

The number of syntax errors can be significantly reduced by replacing the character-based encoder and decoder with a syntax-aware model as done in the Grammar VAE.[22] The SMILES syntax can approximately[1] be described by a so-called context free grammar. Rather than constructing a string character by character we can be represent the string as a sequence of production rules. Not all possible sequences of production rules are valid and the formulation as a context free grammar allows us to enforce this restriction upon the decoder such that all generated sequences are syntactically valid. Kusner et al.[22] also applies the Grammar VAE model to the drug-like molecules from the ZINC database and the results indicates improved smoothness in the latent space representation in comparison to the character based VAE.

In many molecular screening applications we are not interested in the set of all possible molecules, but for example want to limit our search to a smaller set of molecules that are easy to synthesize. In this case the SMILES grammar formulation may be unnecessarily complex and can be replaced with a simpler application specific grammar that is easier to handle for the grammar VAE. This approach is used by Jørgensen et al.[23] for screening of materials for polymer solar cells where each material is composed from a library of acceptor, donor and side group substructures. If the application allows it, the grammar can be formulated such that a syntactically valid string implies a semantically valid molecule, such that the model only generates valid molecules.

5.2 Example: Screening of Polymer Solar Cells Using Grammar VAE

The problem of interest in[23] is to find new materials for polymer solar cells. The polymer units are composed by one of 13 acceptor units, one of 10 donor units and a number of side groups. The crucial properties are the Lowest Unoccupied Molecular Orbital (LUMO) and the optical band gap energy. These properties can be estimated with computationally costly DFT calculations and from a dataset of approximately 4000 DFT calculations we seek to propose new candidate molecules and determine their properties with machine learning. An example polymer solar cell

[1] Not all elements of SMILES are context free, e.g. opening and closing of ring-bonds where the same digit must be used for opening and closing the bond as seen in benzene "c1ccccc1".

molecule (monomer) is shown in Figure 1 and the context free grammar that describes all the possible strings is shown in Figure 2. A model based on this grammar might generate invalid molecules because the number of side groups for each acceptor/donor is not defined by the grammar, but the grammar rules are simpler this way.

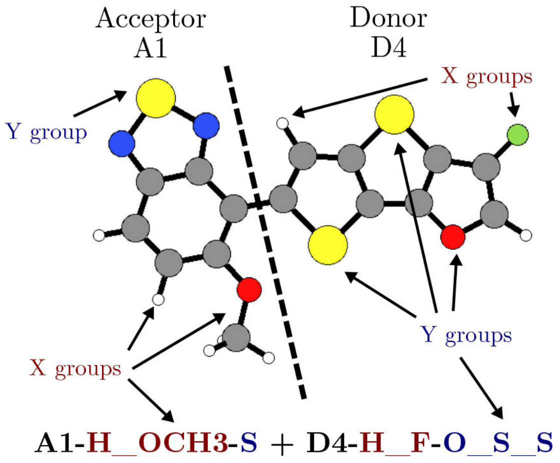


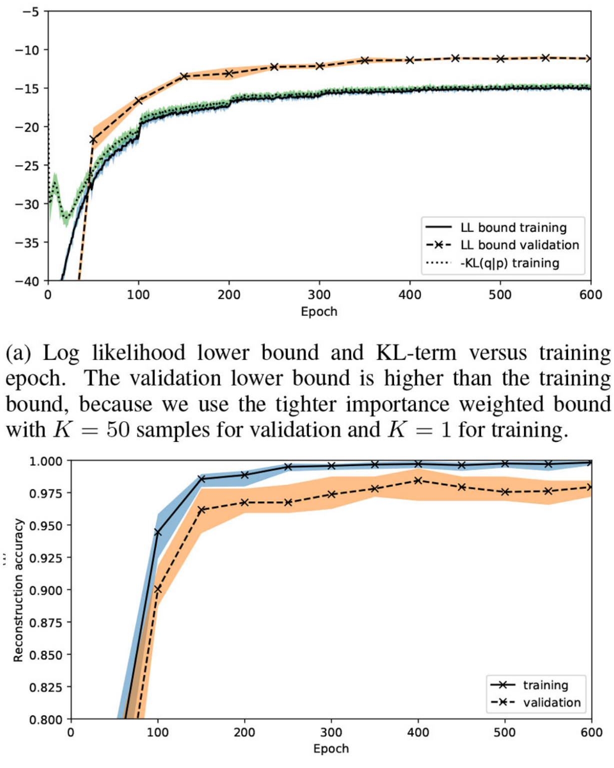
Figure 1. An example molecule from the polymer solar cell dataset and its simplified string representation.

m	→	A – OPTG – OPTG + D – OPTG – OPTG
A	→	A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A1
D	→	D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10
OPTG	→	⋆ | GS
GS	→	G | GS _ G
G	→	Ge | CH3 | OCH3 | H | C | O | SCH3 | NCH3 | S | F | Si | Se

Figure 2. Context free grammar for the polymer solar cell dataset.

We train a grammar VAE using the log likelihood lower bound objective (10) with a constant temperature parameter $\alpha < 1$. Initial studies showed that using $\alpha < 0.1$ was necessary in order to yield good reconstruction error and we use $\alpha = 0.08$ for the training objective and $\alpha = 1$ when evaluating the bound. We use Adam[24] with initial learning rate 0.001 and the learning rate is divided by 2 after every 100 epochs. The log likelihood lower bound for the training and validation set is shown in Figure 3a and the reconstruction accuracy[2] is shown in Figure 3b.

The latent space dimension is set to 32, but as discussed in Section 3.1 it might not be beneficial to use all the latent variables. Because the prior $p(z)$ and $q(z|x)$ factorize across the dimensions of z we can compute the KL-term of the objective function for each of the latent dimensions as shown in Figure 4 for the polymer solar cell grammar VAE. In this example the "effective" dimension of the latent space is around 17, the KL-term for the remaining dimensions is close to zero. Depending on the subsequent application of the model, it may be beneficial to "prune" these dimensions from the model.



(a) Log likelihood lower bound and KL-term versus training epoch. The validation lower bound is higher than the training bound, because we use the tighter importance weighted bound with $K = 50$ samples for validation and $K = 1$ for training.

(b) Average reconstruction accuracy.

Figure 3. Training curves for grammar VAE on the polymer solar cell dataset. We use 5-fold cross-validation to estimate the model's generalisation error. The lines are averages across the 5 folds and the shaded area shows the maximum and minimum.

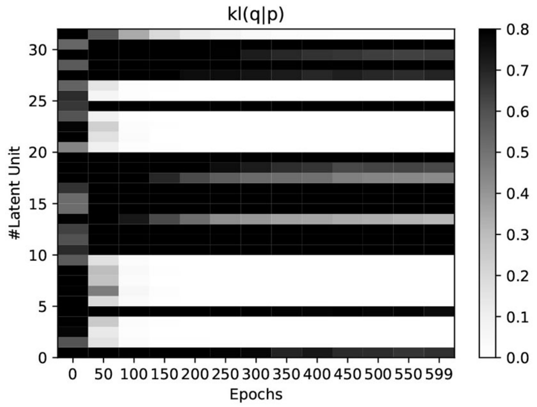
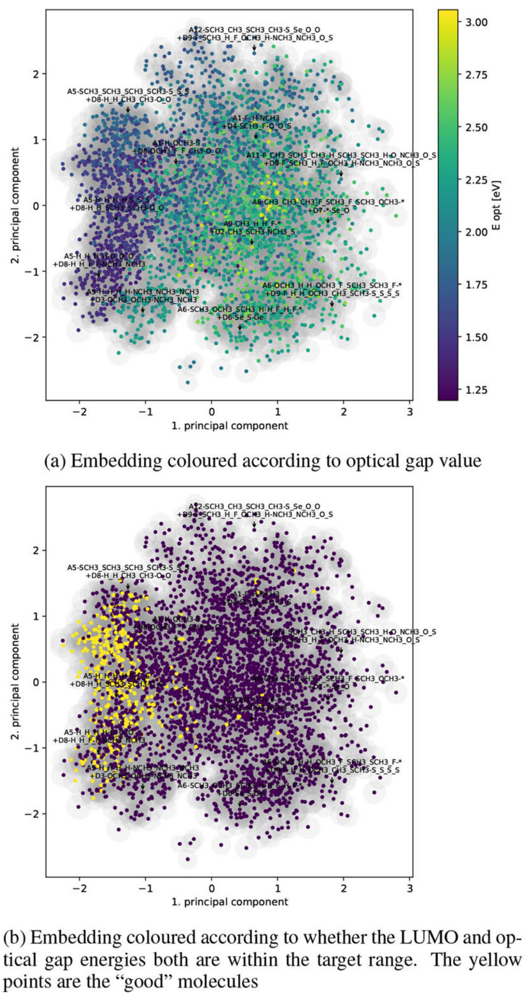


Figure 4. Average (over the training set) KL-divergence

The embedding is visualized using principal component analysis in Figure 5. Even though the grammar VAE is

[2] The reconstruction accuracy is measured in the following way: Each string x is encoded as the mean of the encoding distribution $q_\phi(z|x)$ and decoded using $p_\theta(x|z)$ where the most probable (according to the decoder model) production rule is selected at each step and the encode/decode is considered successful if the output of the decoder matches the encoder input exactly. The accuracy is the success rate across the data set.



(a) Embedding coloured according to optical gap value

(b) Embedding coloured according to whether the LUMO and optical gap energies both are within the target range. The yellow points are the "good" molecules

Figure 5. Visualization of embedding using principal component analysis on the mean value of the embedding corresponding to each data point. The dots marks the mean value and the shaded ellipses are the contours corresponding to half the standard deviation of each point.

trained without using label information we notice some structure when coloring the embedded data points according to the optical band gap energy. The target range for the optical gap is from 1.4 eV to 1.7 eV and the LUMO target range is −3.2 eV to −2.9 eV.

5.3 Generating New Molecules

In our description above we have only discussed the problem of encoding to and decoding from a latent space representation of the discrete input, but often we would like to use this representation to search for molecules with a given property. Since the latent space is a continuous vector space we can then employ our favourite regression/classification on top of this representation. Kusner et al.[22] and Gómez-Bombarelli et al.[19] use a Gaussian process regression model that is trained on the latent space representation and new points in the latent space are selected based on Bayesian optimization, see Brochu et al.[25] for a tutorial on Bayesian optimization. The Bayesian optimization procedure tries to avoid to sample new points that are close to the training data, so we can avoid sampling a point in the latent space that decodes to a molecule that is already in the training set. However, this also means we might sample from areas in the latent space that decodes to invalid molecules.[19] Instead they train a feedforward neural network on the latent representation and then optimize the position by taking a few gradient steps starting from the latent representation of a molecule with good properties.[19] Jørgensen et al.[23] continuing the example above) sample a large number of random points from the latent space and this set of points is decoded and encoded repeatedly to find well-behaved regions of the latent space. After a number of iterations the best points according to the regression model are selected for further studies.

A more advanced approach is to train a regression model in conjunction with the VAE model, as done for classification in semi-supervised learning with deep generative models.[26] In this class of models we can condition the generation of new samples on a specific class label. This is yet to be tried with the SMILES representation, but has been successfully done for drug efficiency classification trained on preand post-treatment gene expression vectors.[27]

6 Other Generative Models

Another popular deep generative model class is Generative adversarial networks (GAN).[28] In this framework the generative model is pitted against a discriminative adversarial model, which is typically also implemented as a neural network. During training the discriminator is optimized to classify whether a sample comes from the data distribution or it is generated by the generative model and the generative model is optimized to fool the discriminator. Some advantages compared to VAEs is that we do not need to specify a variational distribution $q_\phi(z|x)$ and the model can represent sharper data distributions.[28]

The GAN framework has been applied to generation of molecular fingerprints in the DruGAN Adversarial Autoencoder.[29] They effectively replace the KL-term of a VAE with a discriminator that discriminates between the encoder and a sample from $N(0, \mathbf{I})$. Apart from generating new molecular fingerprints they use the VAE as pretraining for an aqueous solubility regression problem.

It is also possible to formulate generative models based on (deep) reinforcement learning in which a software agent builds a data point through a series of actions, e.g. selecting the characters for a SMILES string representation. The agent is trained to maximize a given notion of reward for the generated data. The advantage of this approach is that the reward function can be any function of the taken actions as opposed to VAE where we need to specify a differentiable $p(x|z)$ and for a GAN the discriminator must be differentiable.

Olivecrona et al.[30] trains an autoencoder to reconstruct SMILES strings. Then they replace the decoder with a reinforcement-learning trained decoder to generate molecules with a desired property. They use a trade-off parameter to trade off between generating from the autoencoder (prior) or follow the RL cost function. In objective-reinforced generative adversarial network (ORGAN),[31] the GAN framework is combined with reinforcement learning to generate SMILES strings. They also introduce a trade-off parameter that weights between GAN training (to make molecules look like training data) and RL training (to make molecules with desired property).

7 Conclusion

This paper has given an introduction to variational autoencoders (VAE) and given a few examples of their application for modeling molecule properties. VAE are flexible non-linear latent variables generative models. At the time of writing the VAE have only been around for four years. During these years variants of VAE have pushed the state-of-the-art performance in many unsupervised and semi-supervised benchmarks and found its way into many application areas. Un- and semi-supervised learning are arguably areas where we will see much more research focus in the coming years because supervised learning is much more explored and better understood and because having access to good unsupervised models will enable researchers to explore the vast amounts of unlabeled data available. There are still many open issues around inference with VAE such as how to construct better variational approximations (inference networks) such that we can learn better model for for example sequence data. When these current shortcomings have been dealt with we will likely see even more applications in many data abundant areas such as computational materials science and biomedicine.

Conflict of Interest

None declared.

References

[1] M. Kingma, Diederik P; Welling, *arXiv preprint arXiv:1312.6114* **2013**, 1312.6114.

[2] D. J. Rezende, S. Mohamed, D. Wierstra, *arXiv preprint arXiv:1401.4082* **2014**, 1401.4082.

[3] C. Doersch, *arXiv preprint arXiv:1606.05908* **2016**, 1606.05908.

[4] D. Weininger, *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36.

[5] MDL Information Systems. MACCS Keys.

[6] Y. Burda, R. Grosse, R. Salakhutdinov, In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

[7] G. E. Hinton, D. Van Camp, In *Proceedings of the sixth annual conference on Computational learning theory*, ACM, pp. 5–13.

[8] A. Honkela, H. Valpola, *IEEE Transactions on Neural Networks* **2004**, *15*, 800–810.

[9] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, P. Abbeel, In *International Conference on Learning Representations*.

[10] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, O. Winther, In *Advances in Neural Information Processing Systems 29*, **2016**.

[11] D. J. Rezende, S. Mohamed, *arXiv preprint arXiv:1505.05770* **2015**.

[12] L. Dinh, D. Krueger, Y. Bengio. *arXiv preprint arXiv:1410.8516* **2014**.

[13] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, M. Welling, *arXiv preprint, arXiv:1606.04934* **2016**, 1606.04934.

[14] F. Agakov, D. Barber, In *Neural Information Processing, vol. 3316 of Lecture Notes in Computer Science*, Springer Berlin Heidelberg, **2004**, pp. 561–566.

[15] L. Maaløe, C. K. Sønderby, S. K. Sønderby, O. Winther. In *Proceedings of the International Conference on Machine Learning*.

[16] R. Ranganath, D. Tran, D. M. Blei. *arXiv preprint arXiv:1511.02386* **2015**, 1511.02386.

[17] G. Roeder, Y. Wu, D. Duvenaud. *arXiv preprint arXiv:1703.09194* **2017**.

[18] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, S. Bengio. *arXiv preprint arXiv:1511.06349* **2015**.

[19] R. Gómez-Bombarelli, D. Duvenaud, J. M. Hernndez-Lobato, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, A. Aspuru-Guzik, *arXiv preprint, arXiv:1610.02415* **2016**, 1610.02415.

[20] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, R. G. Coleman, *J. Chem. Inf. Model.* **2012**, *52*, 1757–1768.

[21] R. Gómez-Bombarelli, J. Aguilera-Iparraguirre, T. D. Hirzel, D. Duvenaud, D. Maclaurin, M. A. Blood- Forsythe, H. S. Chae, M. Einzinger, D.-G. Ha, T. Wu, G. Markopoulos, S. Jeon, H. Kang, H. Miyazaki, M. Numata, S. Kim, W. Huang, S. I. Hong, M. Baldo, R. P. Adams, A. Aspuru-Guzik, *Nat. Mater.* **2016**, *15*, 1120–1127.

[22] M. J. Kusner, B. Paige, J. M. Hernndez-Lobato, In *International Conference on Machine Learning*. pp. 1945–1954.

[23] P. B. Jørgensen, M. Mesta, S. Shil, J. M. G. Lastra, K. W. Jacobsen, K. S. Thygensen, M. N. Schmidt. Machine learning-based screening of complex molecules for polymer solar cells, **2017**. Unpublished, to be submitted to Journal of Chemical Physics: Special Issue on Data-Enabled Theoretical Chemistry.

[24] D. Kingma, J. Ba, *arXiv preprint arXiv:1412.6980* **2014**. 1412.6980.

[25] E. Brochu, V. M. Cora, N. de Freitas, *arXiv preprint: arXiv:1012.2599* **2010**, 1012.2599.

[26] D. P. Kingma, D. J. Rezende, S. Mohamed, M. Welling, In *Proceedings of the International Conference on Machine Learning*.

[27] L. Rampasek, D. Hidru, P. Smirnov, B. Haibe-Kains, A. Goldenberg, *arXiv preprint, arXiv:1706.08203* **2017**, 1706.08203.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. In *Advances in Neural Information Processing Systems.* **2014**.

[29] A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, A. Zhavoronkov, *Mol. Pharm.* **2017**, *14*, 3098–3104.

[30] M. Olivecrona, T. Blaschke, O. Engkvist, H. Chen, *J. Cheminform.* **2017**, *9*, 48.

[31] G. L. Guimaraes, B. Sanchez-Lengeling, P. L. C. Farias, A. Aspuru-Guzik, *arXiv preprint, arXiv:1705.10843* **2017**, *1705*, 10843.

Received: November 3, 2017
Accepted: January 13, 2018
Published online on February 6, 2018