NeVAE: A Deep Generative Model for Molecular Graphs

Bidisha Samanta*
IIT Kharagpur
bidisha@iitkgp.ac.in

Abir De
MPI-SWS
ade@mpi-sws.org

Gourhari Jana
IIT Kharagpur
gour2015hari@iitkgp.ac.in

Pratim Kumar Chattaraj
IIT Kharagpur
pkc@chem.iitkgp.ernet.in

Niloy Ganguly
IIT Kharagpur
niloy@cse.iitkgp.ac.in

Manuel Gomez Rodriguez
MPI-SWS
manuelgr@mpi-sws.org

Abstract

Deep generative models have been praised for their ability to learn smooth latent representation of images, text, and audio, which can then be used to generate new, plausible data. However, current generative models are unable to work with molecular graphs due to their unique characteristics—their underlying structure is not Euclidean or grid-like, they remain isomorphic under permutation of the nodes labels, and they come with a different number of nodes and edges. In this paper, we propose NeVAE, a novel variational autoencoder for molecular graphs, whose encoder and decoder are specially designed to account for the above properties by means of several technical innovations. In addition, by using masking, the decoder is able to guarantee a set of valid properties in the generated molecules. Experiments reveal that our model can discover plausible, diverse and novel molecules more effectively than several state of the art methods. Moreover, by utilizing Bayesian optimization over the continuous latent representation of molecules our model finds, we can also find molecules that maximize certain desirable properties more effectively than alternatives.

Introduction

Drug design aims to identify (new) molecules with a set of specified properties, which in turn results in a therapeutic benefit to a group of patients. However, drug design is still a lengthy, expensive, difficult, and inefficient process with low rate of new therapeutic discovery (Paul et al. 2010), in which candidate molecules are produced through chemical synthesis or biological processes. In the context of computer-aided drug design (Merz, Ringe, and Reynolds 2010), there is a great interest in developing automated, machine learning techniques to discover sizeable numbers of plausible, diverse and novel candidate molecules in the vast ($10^{23}-10^{60}$) and unstructured molecular space (Polishchuk, Madzhidov, and Varnek 2013). In recent years, there has been a flurry of work devoted to developing deep generative models for automatic molecule design (Dai et al. 2018; Kusner et al. 2017; Gómez-Bombarelli et al. 2016; Simonovsky and Komodakis 2018; Jin, Barzilay, and Jaakkola 2018), which has predominantly followed two strategies. The first strategy (Dai et al. 2018; Kusner et al. 2017; Gómez-Bombarelli et al. 2016) consists of representing molecules using a domain specific textual representation—SMILES strings—and then leveraging deep generative models for text generation for molecule design. Unfortunately, SMILE strings do not capture the structural similarity between molecules and, moreover, a molecule can have multiple SMILES representations. As a consequence, the generated molecules lack in terms of diversity and validity, as shown in Tables 1–2 and Figure 3. The second strategy (Simonovsky and Komodakis 2018; Jin, Barzilay, and Jaakkola 2018) consists of representing molecules using molecular graphs, rather than SMILES representations, and then developing deep generative models for molecular graphs, in which atoms correspond to nodes and bonds correspond to edges. However, current generative models for molecular graphs share one or more of the following limitations, which preclude them from realizing all their potential: (i) they can only generate (and be trained on) molecules with the same number of atoms while, in practice, molecules having similar properties often come with a different number of atoms and bonds; (ii) they are not invariant to permutations of their node labels, however, graphs remain isomorphic under permutation of their node labels; (iii) their training procedure suffers from a quadratic complexity with respect to the number of nodes in the graph, which make it difficult to leverage a sizeable number of large molecules during training; and, (iv) they generate molecular graphs by combining a small set of molecular *graphlets* (or subgraphs). The above shortcomings constrain the diversity of the generated molecules, as shown in Table 1 and Figure 3.

In this paper, we develop NeVAE, a deep generative model for molecular graphs based on variational autoencoders that overcomes the above shortcomings. To do so, it relies on several technical innovations, which distinguish us from previous work (Dai et al. 2018; Kusner et al. 2017; Gómez-Bombarelli et al. 2016; Simonovsky and Komodakis 2018; Jin, Barzilay, and Jaakkola 2018):

(i) Our probabilistic encoder learns to aggregate information (*e.g.*, atom and bond features) from a different number of hops away from a given atom and then map this aggregate information into a continuous latent space, as in inductive graph representation learning (Hamilton,

*This work was partially done during B. Samanta's internship at MPI-SWS.

Ying, and Leskovec 2017; Lei et al. 2017). However, in contrast with inductive graph representation learning, the aggregator functions are learned via variational inference so that the resulting aggregator functions are especially well suited to enable the probabilistic decoder to generate new molecules rather than other downstream machine learning tasks such as, *e.g.*, link prediction. Moreover, by using (symmetric) aggregator functions, it is invariant to permutations of the node labels and can encode graphs with a variable number of atoms, as opposed to existing graph generative models, with a few the notable exception of those based on GCNs (Kipf and Welling 2016b).

(ii) Our probabilistic decoder jointly represents all edges as an unnormalized log probability vector (or 'logit'), which then feeds a single multinomial edge distribution. Such scheme allows for an efficient inference algorithm with $O(l)$ complexity, where l is the number of true edges in the molecules, which is also invariant to permutations of the node labels. In contrast, previous work typically models the presence and absence of each potential edge using a Bernoulli distribution and this leads to inference algorithms with $O(n^2)$ complexity, where n is the number of nodes, which are not permutation invariant.

(iii) Our probabilistic decoder is able to guarantee a set of local structural and functional properties in the generated graphs by using a *mask* in the edge distribution definition, which can prevent the generation of certain *undesirable* edges during the decoding process. While masking have been increasingly used to account for prior (expert) knowledge in generative models (Gómez-Bombarelli et al. 2016; Kusner et al. 2017) based on SMILES, their use in generative models for molecular graphs has been lacking.

We evaluate our model using molecules from two publicly available datasets, ZINC (Irwin et al. 2012) and QM9 (Ramakrishnan et al. 2014), and show that our model beats the state of the art in terms of several relevant quality metrics, *i.e.*, validity, novelty and uniqueness.

We also observe that the resulting latent space representation of molecules exhibit powerful semantics—we can smoothly interpolate between molecules—and generalization ability—we can generate (valid) molecules that are larger than any of the molecules in the datasets. Finally, by utilizing Bayesian optimization over the latent representation, we can also identify molecules that maximize certain desirable properties more effectively than alternatives. We are releasing an open source implementation of our model in Tensorflow. [1]

Background on Variational Autoencoders

Variational autoencoders (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014) are characterized by a probabilistic generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ of the observed variables $\boldsymbol{x} \in \mathbb{R}^N$ given the latent variables $\boldsymbol{z} \in \mathbb{R}^M$, a prior distribution over the latent variables $p(\boldsymbol{z})$ and an approximate probabilistic inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. In this characterization, p_θ and q_ϕ are arbitrary distributions parametrized by two (deep) neural networks θ and ϕ and one can think of the generative model as a probabilistic *decoder*, which *decodes* latent variables into observed variables, and the inference model as a probabilistic *encoder*, which *encodes* observed variables into latent variables.

Ideally, if we use the maximum likelihood principle to train a variational autoencoder, we should optimize the marginal log-likelihood of the observed data, *i.e.*, $\mathbb{E}_{\mathcal{D}}\left[\log p_\theta(\boldsymbol{x})\right]$, where $p_{\mathcal{D}}$ is the data distribution. Unfortunately, computing $\log p_\theta(\boldsymbol{x})$ requires marginalization with respect to the latent variable $\boldsymbol{z}$, which is typically intractable. Therefore, one resorts to maximizing a variational lower bound or evidence lower bound (ELBO) of the log-likelihood of the observed data, *i.e.*,

$$\max_\theta \max_\phi \mathbb{E}_{\mathcal{D}}\left[-\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right].$$

Finally, note that the quality of this variational lower bound depends on the expressive ability of the approximate inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, which is typically assumed to be a normal distribution whose mean and variance are parametrized by a neural network ϕ with the observed data $\boldsymbol{x}$ as an input.

NeVAE: A Variational Autoencoder for Molecular Graphs

In this section, we first give a high-level overview of the design of NeVAE, our variational autoencoder for molecular graphs, starting from the data it is designed for. Then, we describe more in-depth the key technical aspects of its individual components. Finally, we elaborate on the training procedure, scalability and implementation details.

High-level overview. We observe a collection of N molecular graphs $\{\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)\}_{i\in[N]}$, where $\mathcal{V}_i$ and $\mathcal{E}_i$ denote the corresponding set of nodes (atoms) and edges (bonds), respectively, and this collection may contain graphs with a different number of nodes and edges. Moreover, for each molecular graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we also observe a set of node features $\mathcal{F} = \{\boldsymbol{f}_u\}_{u\in\mathcal{V}}$ and edge weights $\mathcal{Y} = \{y_{uv}\}_{(u,v)\in\mathcal{E}}$. More specifically, the node features $\boldsymbol{f}_u$ are one-hot representations of the type of the atoms (*i.e.*, C, H, N or O), and the edge weight y_{uv} are the bond types (*i.e.*, single, double, triple). Our goal is then to design a variational autoencoder for molecular graphs that, once trained on this collection of graphs, has the ability of creating new plausible molecular graphs, including node features and edge weights. In doing so, it will also provide a latent representation of any graph in the collection (or elsewhere) with meaningful semantics.

Following the above background on variational autoencoders, we characterize NeVAE by means of:

— *Prior*: $p(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$, where $|\mathcal{V}| = |\mathcal{F}| = n \sim \text{Poisson}(\lambda_n)$

— *Inference model (encoder)*: $q_\phi(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n|\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{Y})$

— *Generative model (decoder)*: $p_\theta(\mathcal{E}, \mathcal{F}, \mathcal{Y}|\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$

In the above characterization, note that we define one latent variable per node, *i.e.*, we have a *node-based* latent representation, and the number of nodes is a random variables and, as

[1] https://github.com/Networks-Learning/nevae

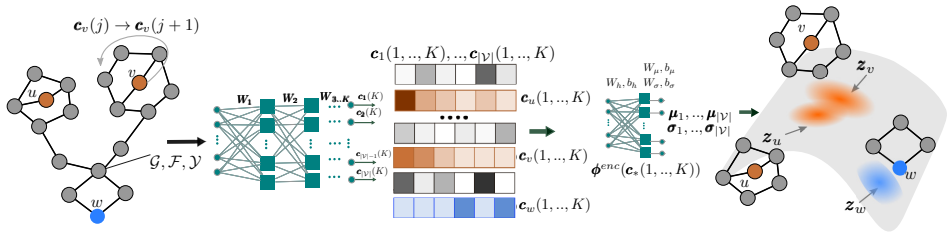


Figure 1: The encoder of our variational autoencoder for molecular graphs. From left to right, given a molecular graph $\mathcal{G}$ with a set of node features $\mathcal{F}$ and edge weights $\mathcal{Y}$, the encoder aggregates information from a different number of hops $j \leq K$ away for each node $v \in \mathcal{G}$ into an embedding vector $\mathbf{c}_v(j)$. These embeddings are fed into a differentiable function ϕ^{enc} which parameterizes the posterior distribution q_ϕ, from where the latent representation of each node in the input graph are sampled from.

a consequence, both the latent representation as well as the graph can vary in size. Next, we formally define the functional form of the inference model, the generative model, and the prior.

Inference model (probabilistic encoder). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node features $\mathcal{F}$ and edge weights $\mathcal{Y}$, our inference model q_ϕ defines a probabilistic encoding for each node in the graph by aggregating information from different distances. More formally, for each node u, the inference model is defined as follows:

$$q_\phi(\boldsymbol{z}_u|\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{Y}) \sim \mathcal{N}(\boldsymbol{\mu}_u, \mathrm{diag}(\boldsymbol{\sigma}_u)) \tag{1}$$

where $\boldsymbol{z}_u$ is the latent variable associated to node u, $[\boldsymbol{\mu}_u, \mathrm{diag}(\boldsymbol{\sigma}_u)] = \phi^{enc}(\mathbf{c}_u(1), \ldots, \mathbf{c}_u(K))$, and $\mathbf{c}_u(k)$ aggregates information from k hops away from u, *i.e.*,

$$\mathbf{c}_u(k) = \begin{cases} \boldsymbol{r}(\boldsymbol{W}_k \boldsymbol{f}_u) & \text{if } k = 1 \\ \boldsymbol{r}\left(\boldsymbol{W}_k \boldsymbol{f}_u \odot \boldsymbol{\Lambda}\big(\cup_{v \in \mathcal{N}(u)} y_{uv}\, \boldsymbol{g}(\mathbf{c}_v(k-1))\big)\right) & \text{if } k > 1. \end{cases} \tag{2}$$

In the above, $\boldsymbol{W}_k$ are trainable weight matrices, which propagate information between different search depths, $\boldsymbol{\Lambda}(.)$ is a (possibly nonlinear) symmetric aggregator function in its arguments, $\boldsymbol{g}(\cdot)$ and $\boldsymbol{r}(\cdot)$ are (possibly nonlinear) differentiable functions, ϕ^{enc} is a neural network, and $\odot$ denotes pairwise product. Figure 1 describes our encoder architecture.

The above node embeddings, defined by Eq. 2, are very similar to the ones used in several graph representation learning algorithms such as GraphSAGE (Hamilton, Ying, and Leskovec 2017), column networks (Pham et al. 2017), and GCNs (Kipf and Welling 2016a), the main difference with our work is the way we will train the weight matrices $\boldsymbol{W}_k$. Here, we will use variational inference so that the resulting embeddings are especially well suited to enable our probabilistic decoder to generate new, plausible molecular graphs. In contrast, the above algorithms use non variational approaches to compute general purpose embeddings to feed downstream machine learning tasks.

The following proposition highlights several desirable theoretical properties of our probabilistic encoder (details in arxiv version),[2] which distinguishes our design from most existing generative models of graphs (Jin, Barzilay, and Jaakkola 2018; Simonovsky and Komodakis 2018):

[2]https://arxiv.org/abs/1802.05283

Proposition 1 *The probabilistic encoder defined by Eqs. 1 and 2 has the following properties:*

(i) For each node u, its corresponding embedding $\mathbf{c}_u(k)$ is invariant to permutations of the node labels of its neighbors.

(ii) The weight matrices $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_k$ do not depend on the number of nodes and edges in the graph and thus a single encoder allows for graphs with a variable number of nodes and edges.

Generative model (probabilistic decoder). Given a set of of n nodes with latent variables $\mathcal{Z} = \{\boldsymbol{z}_u\}_{u \in [n]}$, our generative model p_θ is defined as follows:

$$p_\theta(\mathcal{E}, \mathcal{Y}, \mathcal{F}|\mathcal{Z}) = p_\theta(\mathcal{F}|\mathcal{Z})\, p_\theta(\mathcal{E}, \mathcal{Y}|\mathcal{Z}), \tag{3}$$

with

$$p_\theta(\mathcal{F}|\mathcal{Z}) = \prod_{u \in \mathcal{V}} p_\theta(\boldsymbol{f}_u|\mathcal{Z}),$$

$$p_\theta(\mathcal{E}, \mathcal{Y}|\mathcal{Z}) = p_\theta(l\big|\mathcal{Z})\ .\ p_\theta(\mathcal{E}, \mathcal{Y}|\mathcal{Z}, l),$$

$$p_\theta(\mathcal{E}, \mathcal{Y}|\mathcal{Z}, l) = \prod_{k \in [l]} p_\theta(e_k|\mathcal{E}_{k-1}, \mathcal{F}, \mathcal{Z}) p_\theta(y_{u_k v_k}|\mathcal{Y}_{k-1}, \mathcal{F}, \mathcal{Z}),$$

where the ordering for the edge and edge weights is independent of node labels and hence permutation invariant, e_k and $y_{u_k v_k}$ denote the k-th edge and edge weight under the chosen order, and $\mathcal{E}_{k-1} = \{e_1, \ldots, e_{k-1}\}$ and $\mathcal{Y}_{k-1} = \{y_{u_1 v_1}, \ldots, y_{u_{k-1} v_{k-1}}\}$ denote the $k-1$ previously generated edges and edge weights respectively.

Moreover, the model characterizes the conditional probabilities in the above formulation as follows. For each node, it represents all potential node feature values $\boldsymbol{f}_u = \boldsymbol{q}$ as an unnormalized log probability vector (or 'logits'), feeds this logit into a softmax distribution and samples the node features. Then, it represents the average number of edges through as a logit, feeds this logit into a Poisson distribution and samples the number of edges. Finally, it represents all potential edges as logits and, for each edge, all potential edge weights as another logit, and it feeds the former vector into a single softmax distribution and the latter vectors each into a different softmax distribution. Moreover, the edge distribution and the corresponding edge weight distributions depend on a set of binary *masks*, which may depend on the sampled node features and also get updated every time a new edge and edge weight are sampled. By doing so,

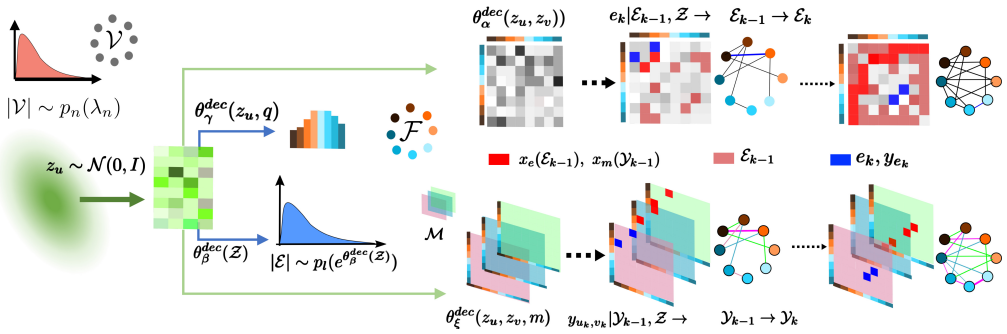


Figure 2: The decoder of our variational autoencoder for molecular graphs. From left to right, the decoder first samples the number of nodes $n = |\mathcal{V}|$ from a Poisson distribution $p_n(\lambda_n)$ and it samples a latent vector $\boldsymbol{z}_u$ per node $u \in \mathcal{V}$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Then, for each node u, it represents all potential node feature values as an unnormalized log probability vector (or 'logits'), where each entry is given by a nonlinearity θ_γ^{dec} of the corresponding latent representation $\boldsymbol{z}_u$, feeds this logit into a softmax distribution and samples the node features. Next, it feeds all latent vectors $\mathcal{Z}$ into a nonlinear log intensity function $\theta_{\boldsymbol{\beta}}^{dec}(\mathcal{Z})$ which is used to sample the number of edges. Thereafter, on the top row, it constructs a logit for all potential edges (u, v), where each entry is given by a nonlinearity θ_α^{dec} of the corresponding latent representations $(\boldsymbol{z}_u, \boldsymbol{z}_v)$. Then, it samples the edges one by one from a soft max distribution depending on the logit and a mask $x_e(\mathcal{E}_{k-1})$, which gets updated every time it samples a new edge e_k. On the bottom row, it constructs a logit per edge (u, v) for all potential edge weight values m, where each entry is given by a nonlinearity θ_ξ^{dec} of the latent representations of the edge and edge weight value $(\boldsymbol{z}_u, \boldsymbol{z}_v, m)$. Then, every time it samples an edge, it samples the edge weight value from a soft max distribution depending on the corresponding logit and mask $x_m(u, v)$, which gets updated every time it samples a new $y_{u_k v_k}$.

it prevents the generation of certain *undesirable* edges and edges weights, allowing for the generated graph to fulfill a set of predefined local structural and functional properties.

More formally, the distributions of each node feature, the number of edges, each edge and edge weight are given by:

$$p_\theta(\boldsymbol{f}_u = \boldsymbol{q}|\mathcal{Z}) = \frac{e^{\theta_{\boldsymbol{\gamma}}^{dec}(\boldsymbol{z}_u, \boldsymbol{q})}}{\sum_{\boldsymbol{q}'} e^{\theta_{\boldsymbol{\gamma}}^{dec}(\boldsymbol{z}_u, \boldsymbol{q}')}}, \quad p_\theta(l|\mathcal{Z}) = p_l(e^{\theta_{\boldsymbol{\beta}}^{dec}(\mathcal{Z})}),$$

$$p_\theta(e = (u, v)|\mathcal{E}_{k-1}, \mathcal{Z}) = \frac{x_e e^{\theta_{\boldsymbol{\alpha}}^{dec}(\boldsymbol{z}_u, \boldsymbol{z}_v)}}{\sum_{e'=(u',v') \notin \mathcal{E}_{k-1}} x_{e'} e^{\theta_{\boldsymbol{\alpha}}^{dec}(\boldsymbol{z}_{u'}, \boldsymbol{z}_{v'})}},$$

$$p_\theta(y_{uv} = m|\mathcal{Y}_{k-1}, \mathcal{Z}) = \frac{x_m(u, v) e^{\theta_{\boldsymbol{\xi}}^{dec}(\boldsymbol{z}_u, \boldsymbol{z}_v, m)}}{\sum_{m' \neq m} x_{m'}(u, v) e^{\theta_{\boldsymbol{\xi}}^{dec}(\boldsymbol{z}_u, \boldsymbol{z}_v, m')}},$$

where p_l denotes a Poisson distribution, x_e is the binary mask for edge e and $x_m(u, v)$ is the binary mask for feature edge value m, and $\theta_\bullet^{dec}$ are neural networks. Note that the parameters of the neural networks do not depend on the number of nodes or edges in the molecular graph and the dependency of the binary masks x_e and $x_m(u, v)$ on the node features and the previously generated edges $\mathcal{E}_{k-1}$ and edge weights $\mathcal{Y}_{k-1}$ is deterministic and domain dependent. Figure 2 summarizes our decoder architecture.

Note that, by using a softmax distribution, it is only necessary to account for the presence of an edge, not its absence, and this, in combination with negative sampling, will allow for efficient training and decoding, as it will become clear later in this section. This is in contrast with previous generative models for graphs (Kipf and Welling 2016b; Simonovsky and Komodakis 2018), which need to model both the presence and absence of each potential edge. Moreover, we would like to acknowledge that, while masking may be useful to account for prior (expert) knowledge, it may be costly to check for some local (or global) structural and functional properties on-the-fly.

Prior. Given a set of n nodes with latent variables $\mathcal{Z} = \{\boldsymbol{z}_u\}_{u \in [n]}$, $p_z(\mathcal{Z}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

Training. Given a collection of N molecular graphs $\{\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)\}_{i \in [N]}$, each with n_i nodes, a set of node features $\mathcal{F}_i$ and set of edge weights $\mathcal{Y}_i$, we train our variational autoencoder for graphs by maximizing the evidence lower bound (ELBO), as described in the previous section, plus the log-likelihood of the Poisson distribution p_{λ_n} modeling the number of nodes in each graph. Hence we aim to solve:

$$\underset{\phi, \theta, \lambda_n}{\text{maximize}} \ \frac{1}{N} \sum_{i \in [N]} \Big(\mathbb{E}_{q_\phi(\mathcal{Z}_i|\mathcal{V}_i, \mathcal{E}_i, \mathcal{F}_i, \mathcal{Y}_i)} \log p_\theta(\mathcal{E}_i, \mathcal{Y}_i, \mathcal{F}_i|\mathcal{Z}_i) - \mathrm{KL}(q_\phi||p_z) + \log p_{\lambda_n}(n_i) \Big) \tag{4}$$

Note that, in the above objective, computation of $\mathbb{E}_{q_\phi} \log p_\theta(\mathcal{E}_i, \mathcal{Y}_i, \mathcal{F}_i|\mathcal{Z}_i)$ requires to specify an order of edges present in the graph $\mathcal{G}_i$. To determine this order, we use breadth-first-traversals (BFS) with randomized tie breaking during the child-selection step. Such a tie breaking method makes the edge order independent of all node labels except for the source node label. Therefore, to make it completely permutation invariant, for each graph, we sample the source nodes from an arbitrary distribution. More formally, we replace $\log p_\theta(\mathcal{E}_i, \mathcal{Y}_i, \mathcal{F}_i|\mathcal{Z}_i)$ with $\log \mathbb{E}_{s \sim \zeta(\mathcal{V}_i)} p_\theta(\mathcal{E}_i, \mathcal{Y}_i, \mathcal{F}_i|\mathcal{Z}_i)$ for each graph $\mathcal{G}_i$, where s is the randomly sampled source node for the BFS, and ζ is the sampling distribution for s. Note that, the logarithm of a marginalized likelihood is difficult to compute. Fortunately, by using Jensen inequality, we can have a lower bound of the actual likelihood:

$$\log \mathbb{E}_{s \sim \zeta(\mathcal{V}_i)} p_\theta(\mathcal{E}_i, \mathcal{Y}_i, \mathcal{F}_i|\mathcal{Z}_i) \geq \mathbb{E}_{s \sim \zeta(\mathcal{V}_i)} \log p_\theta(\mathcal{E}_i, \mathcal{Y}_i, \mathcal{F}_i|\mathcal{Z}_i)$$

Therefore, to train our model, we maximize

$$\frac{1}{N}\sum_{i\in[N]}\big(\mathbb{E}_{q_\phi(\mathcal{Z}_i|\mathcal{V}_i,\mathcal{E}_i,\mathcal{F}_i,\mathcal{Y}_i),s\sim\zeta(\mathcal{V}_i)}\log p_\theta(\mathcal{E}_i,\mathcal{Y}_i,\mathcal{F}_i|\mathcal{Z}_i) - \mathrm{KL}(q_\phi||p_z)+\log p_{\lambda_n}(n_i)\big), \quad (5)$$

The following theorem points out the key property of our objective function (proven in arxiv version).[3]

Theorem 2 *If the source distribution ζ does not depend on the node labels, then the parameters learned by maximizing the objective in Eq. 5 are invariant to the permutations of the node labels.*

Scalability and implementation details. In terms of scalability, the major bottleneck is computing the gradient of the first term in Eq. 5 during training, rather than encoding and decoding graphs once the model is trained. More specifically, given a source node for a network without masks, an exact computation of the per edge partition function of the log-likelihood of the edges, *i.e.*, $\sum_{e'=(u',v')\notin\mathcal{E}_{k-1}}\exp(\theta^{dec}_{\boldsymbol{\alpha}}(\boldsymbol{z}_{u'},\boldsymbol{z}_{v'}))$, requires $O(|\mathcal{V}|^2)$ computations, similarly as in most inference algorithms for existing generative models of graphs, and hence is costly to compute even for medium networks. Fortunately, in practice, we can approximate such partition function using negative sampling (Mikolov et al. 2013) which reduces the likelihood computation to $O(l)$, where $l=|\mathcal{E}|$ is the number of (true) edges in the graph. Therefore, for S samples of source nodes, the complexity becomes $O(Sl)$. Here, note that most real-world graphs are sparse and thus $l\ll|\mathcal{V}|^2$.

Experiments on Real Data

In this section, we first show that our model beats several state of the art machine learning models for molecule design (Dai et al. 2018; Gómez-Bombarelli et al. 2016; Kusner et al. 2017; Simonovsky and Komodakis 2018; Jin, Barzilay, and Jaakkola 2018; Liu et al. 2018) in terms of several relevant quality metrics, *i.e.*, *validity*, *novelty* and *uniqueness*. Then, by applying Bayesian optimization over the latent space of molecules provided by our encoder, we also show that our model can find a greater number of molecules that maximize certain desirable properties. Finally we show that the continuous latent representations of molecules that our model finds are smooth.

Experimental setup. We sample ∼10,000 drug-like commercially available molecules from the ZINC dataset (Irwin et al. 2012) with $\mathbb{E}[n]=44$ atoms and ∼10,000 molecules from the QM9 dataset (Ramakrishnan et al. 2014; Ruddigkeit et al. 2012) with $\mathbb{E}[n]=21$ atoms. For each molecule, we construct a molecular graph, where nodes are the atoms, the node features are the type of the atoms *i.e.* $\boldsymbol{f}_u\in\{C,H,N,O\}$, edges are the bonds between two atoms, and the weight associated to an edge is the type of bonds (single, double or triple)[4]. Then, for each dataset, we train our variational autoencoder for molecular graphs using batches comprised of molecules with the same number of nodes[5]. Finally, we sample 10^6 molecular graphs from each of the (two) trained variational autoencoders using: (i) $\mathcal{G}\sim p_\theta(\mathcal{G}|\mathcal{Z})$, where $\mathcal{Z}\sim p(\mathcal{Z})$ and (ii) $\mathcal{Z}\sim p_\theta(\mathcal{Z}|G=G_T)$, where G_T is a molecular graph from the corresponding (training) dataset. In the above procedure, we only use masking on the weight (*i.e.*, type of bond) distributions both during training and sampling to ensure that the valence of the nodes at both ends are valid at all times, *i.e.*, $x_m(u,v)=\mathbb{I}(m+n_k(u)\leq m_{max}(u)\wedge m+n_k(v)\leq m_{max}(v))$, where $n_k(u)$ is the current valence of node u and $m_{max}(u)$ is the maximum valence of node u, which depends on its type $\boldsymbol{f}_u$. Moreover, during sampling, if there is no valid weight value for a sampled edge, we reject it. To assess to which extent masking helps, we also train and sample from our model without masking. Here, we would like to highlight that, while using masking during test does not lead to significant increase in the time it takes to generate a graph, using masking during training does lead to an increase of 5% in training time.

We compare the quality of the molecules generated by our trained models and the molecules generated by several state of the art competing methods: (i) GraphVAE (Simonovsky and Komodakis 2018), (ii) GrammarVAE (Kusner et al. 2017), (iii) CVAE (Gómez-Bombarelli et al. 2016), (iv) SDVAE (Dai et al. 2018), (v) JTVAE (Jin, Barzilay, and Jaakkola 2018) and (vi) CGVAE (Liu et al. 2018). Among them, GraphVAE, JTVAE and CGVAE use molecular graphs, however, the rest of the methods use SMILES strings, a domain specific textual representation of molecules. We use the following evaluation metrics for performance comparison:

(i) *Novelty*: we use this metric to evaluate to which degree a method generates novel molecules, *i.e.*, molecules which were not present in the (training) dataset, *i.e.* Novelty $=1-|\mathcal{C}_s\cap\mathcal{D}|/|\mathcal{C}_s|$, where $\mathcal{C}_s$ is the set of generated molecules which are chemically valid, $\mathcal{D}$ is the training dataset, and Novelty $\in[0,1]$.

(ii) *Uniqueness*: we use this metric to evaluate to what extent a method generates unique chemically valid molecules. We define, Uniqueness $=|\text{set}(\mathcal{C}_s)|/n_s$ where n_s is the number of generated molecules and Unique $\in[0,1]$.

(iii) *Validity*: we use this metric to evaluate to which degree a method generates chemically valid molecules[6]. That is, Validity $=|\mathcal{C}_s|/n_s$ where n_s is the number of generated molecules, $\mathcal{C}_s$ is the set of generated molecules which are chemically valid, and note that Validity $\in[0,1]$.

Quality of the generated molecules. Tables 1–2 compare our trained models to the state of the art methods above in terms of novelty, uniqueness, and validity. For GraphVAE and CGVAE we report the results reported in the paper and, for SDVAE, since there is no public domain implementation

[3] https://arxiv.org/abs/1802.05283

[4] We have not selected any molecule whose bond types are others than these three.

[5] We batch graphs with respect to the number of nodes for efficiency reasons since, every time that the number of nodes changes, we need to change the size of the computational graph in Tensorflow.

[6] We used the opensource cheminformatics suite RDkit (http://www.rdkit.org) to check the validity of a generated molecule.

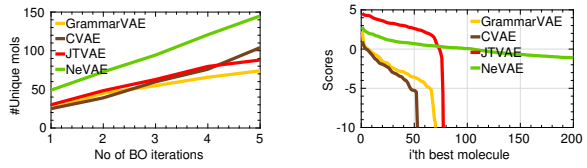
	Novelty							
Dataset	**NeVAE**	**NeVAE***	**GraphVAE**	**GrammarVAE**	**CVAE**	**SDVAE**	**JTVAE**	**CGVAE**
ZINC	1.000	1.000	-	1.000	0.980	1.000	0.999	1.000
QM9	1.000	1.000	0.661	1.000	0.902	-	1.000	0.943
	Uniqueness							
Dataset	**NeVAE**	**NeVAE***	**GraphVAE**	**GrammarVAE**	**CVAE**	**SDVAE**	**JTVAE**	**CGVAE**
ZINC	0.999	0.588	-	0.273	0.021	1.000	0.991	0.998
QM9	0.998	0.676	0.305	0.197	0.031	-	0.371	0.986

Table 1: Novelty and Uniqueness of the molecules generated using NeVAE and all baselines. The sign * indicates no masking. For both the datasets, we report Novelty (Uniqueness) over valid (10^6) sampled molecules.

		Validity							
Dataset	**Sampling type**	**NeVAE**	**NeVAE***	**GraphVAE**	**GrammarVAE**	**CVAE**	**SDVAE**	**JTVAE**	**CGVAE**
ZINC	$\boldsymbol{Z} \sim P(\boldsymbol{Z})$	1.000	0.590	0.135	0.440	0.021	0.432	1.000	1.000
	$\boldsymbol{Z} \sim P(\boldsymbol{Z}\|G_T)$	1.000	0.580	-	0.381	0.175	-	1.000	-
QM9	$\boldsymbol{Z} \sim P(\boldsymbol{Z})$	0.999	0.682	0.458	0.200	0.031	-	0.997	1.000
	$\boldsymbol{Z} \sim P(\boldsymbol{Z}\|G_T)$	0.999	0.660	-	0.301	0.100	-	0.965	-

Table 2: Validity the molecules generated using NeVAE and all baselines. The sign * indicates no masking. For both the datasets, we report the numbers over 10^6 sampled molecules.

of these methods at the time of writing, we have used the sampled molecules from the prior provided by the authors for the ZINC dataset. For CVAE, GrammarVAE and JTVAE, we run their public domain implementations in the same set of molecules that we used. We find that, in terms of novelty, both our trained models and all competing methods except for the GraphVAE, which assumes a fixed number of nodes, are able to (almost) always generate novel molecules. However, we would also like to note that novelty is only defined over chemically valid molecules. Therefore, despite having (almost) perfect novelty scores, all baselines except JTVAE generate significantly fewer novel molecules than our method. In terms of uniqueness, which is defined over the set of sampled molecules, we observe that all baseline methods, except CGVAE (for ZINC and QM9) and JTVAE (for ZINC), perform very poorly in both datasets in comparison with NeVAE. In terms of validity, our trained model significantly outperform four competing methods—GraphVAE, GrammarVAE, CVAE and SDVAE—even without the use of masking, and achieve a comparable performance to JTVAE and CGVAE. In contrast to our model, GrammarVAE,

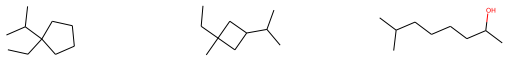


(a) Uniqueness (b) Score

Figure 3: Property maximization using Bayesian optimization. Each plot shows the values of $y(m)$ in decreasing order for unique molecules. Panel (a) shows the variation of Uniqueness with the no. of BO iterations. Panel (b) shows the values of $y(m)$ sorted in the decreasing order.

Objective	NeVAE	GrammarVAE	CVAE	JTVAE
LL	**-1.45**	-1.75	-2.29	*-1.54*
RMSE	**1.23**	1.38	1.80	*1.25*
Fraction of *valid* molecules	**1.00**	0.77	0.53	**1.00**
Fraction of *unique* molecules	**0.58**	0.29	0.41	0.32

Table 3: Property prediction performance (LL and RMSE) using Sparse Gaussian processes (SGPs) and property maximization using Bayesian Optimization (BO).

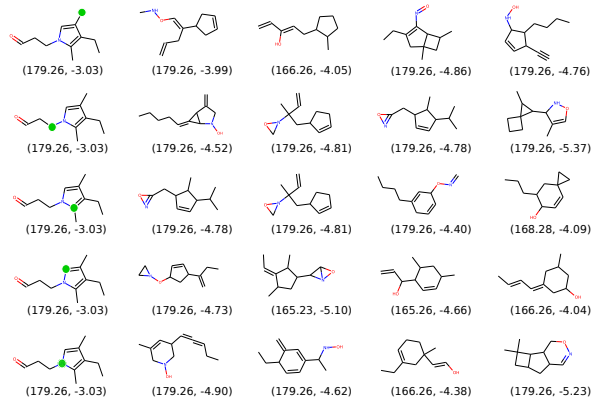


$y(m) = 2.826$ (1st) $y(m) = 2.477$ (2nd) $y(m) = 2.299$ (3rd)

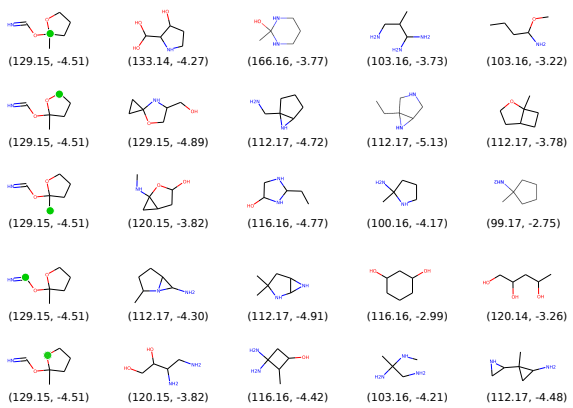
Figure 4: Best molecules found by Bayesian Optimization (BO) using our model.

CVAE and SDVAE use SMILES, a domain specific string based representation, and thus they may be constrained by its limited expressiveness. Among them, GrammarVAE and SDVAE achieve better performance by using a grammar to favor valid molecules. GraphVAE generates molecular graphs, as our model, however, its performance is inferior to our method because it assumes a fixed number of nodes, it samples edges independently from a Bernoulli distribution, and is not permutation invariant.

Bayesian optimization. Here, we leverage our model to discover novel molecules with desirable properties. Similarly as in previous work (Gómez-Bombarelli et al. 2016; Kusner et al. 2017; Jin, Barzilay, and Jaakkola 2018), we use Bayesian optimization (BO) to identify novel molecules m with a high value of the octanol-water partition coefficient (logP) $y(m)$, penalized by synthetic accessibility (SA) score and number of long cycles. More specifically, we first sam-



(a) ZINC dataset



(b) QM9 Dataset

Figure 5: Molecules sampled using the probabilistic decoder $\mathcal{G} \sim p_\theta(\mathcal{G}|\mathcal{Z})$, where $\mathcal{Z} = \{\boldsymbol{z}_i + a_i \boldsymbol{z}_i | \, \boldsymbol{z}_i \in \mathcal{Z}_0, a_i \geq 0\}$ and a_i are given parameters. In each row, we use same molecule set $a_i > 0$ for a single arbitrary node i (denoted as ●) and set $a_j = 0, j \neq i$ for the remaining nodes. Under each molecule we report its molecular weight and synthetic accessibility score.

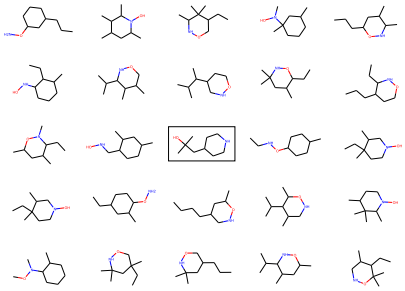


Figure 6: Molecules sampled using probabilistic decoder, *i.e.* $\mathcal{G}_i \sim p_\theta(\mathcal{G}|\mathcal{Z})$, given the (sampled) latent representation $\mathcal{Z}$ of a given molecule $\boxed{\mathcal{G}}$ from the ZINC dataset. The sampled molecules are topologically similar to each other as well as the original.

ple 3,000 molecules from our ZINC dataset, which we split into training (90%) and test (10%) sets. Then, for our model and each competing model with public domain implementations, we train a sparse Gaussian process (SGP) (Snelson and Ghahramani 2006) with the latent representations and $y(m)$ values of 100 inducing points sampled from the training set. The SGPs allow us to make predictions for the property values of new molecules in the latent spaces. Then, we run 5 iterations of batch Bayesian optimization (BO) using the expected improvement (EI) heuristic (Jones, Schonlau, and Welch 1998), with 50 (new) latent vectors (molecules) per iteration. Here, we compare the performance of all models using several quality measures: (a) the predictive performance of the trained SGPs in terms of log-likelihood (LL) and root mean square error (RMSE) on the test set and (b) the average value $\mathbb{E}[y(m)]$, fraction of valid molecules and fraction of *good* molecules, *i.e.*, $y(m) > 0$, among the molecules found using EI.

Table 3, Figure 3 and Figure 4 summarize the results. In terms of log-likelihood and RMSE, the SGP trained using the latent representations provided by our model outperforms all baselines. In terms of the property values $\mathbb{E}[y(m)]$ of the discovered molecules and fraction of valid and good molecules, BO under NeVAE also outperforms all baselines. Here, we would like to highlight that, while BO under JT-VAE is able to find a few molecules with larger property value than BO under NeVAE, it is unable to discover a sizeable set of unique molecules with high property values.

Smooth latent space of molecules. In this section, we first demonstrate (qualitatively) that the latent space of molecules inferred by our model is smooth. Given a molecule, along with its associated graph $\mathcal{G}$, node features $\mathcal{F}$ and edge weights $\mathcal{Y}$, we first sample its latent representation $\mathcal{Z}$ using our probabilistic encoder, *i.e.*, $\mathcal{Z} \sim q_\phi(\mathcal{Z}|G, \mathcal{F}, \mathcal{Y})$. Then, given this latent representation, we generate various molecular graphs by sampling from our probabilistic decoder, *i.e.*, $\mathcal{G}_i \sim p_\theta(\mathcal{G}|\mathcal{Z})$. Figure 6 summarizes the results for one molecule from ZINC dataset, which show that the sampled molecules are topologically similar to the given molecule.

Next, we show that our encoder, once trained, creates a latent space representation of molecules with powerful semantics. In particular, since each node in a molecule has a latent representation, we can make fine-grained changes to the structure of a molecule by perturbing the latent representation of single nodes. To this aim, we proceed by first selecting one molecule with n nodes from the ZINC dataset. Given its corresponding graph, node features and edge weights, $\mathcal{G}$, $\mathcal{F}$ and $\mathcal{Y}$, we sample its latent representation $\mathcal{Z}_0$. Then, we sample new molecular graphs $\mathcal{G}$ from the probabilistic decoder $\mathcal{G} \sim p_\theta(\mathcal{G}|\mathcal{Z})$, where $\mathcal{Z} = \{\boldsymbol{z}_i + a_i \boldsymbol{z}_i | \, \boldsymbol{z}_i \in \mathcal{Z}_0, a_i \geq 0\}$ and a_i are given parameters. Figure 5 provides several examples across both datasets, which show that the latent space representation is smooth and, as the distance from the initial molecule increases in the latent space, the resulting molecule differs more from the original. Here, note that the interpolation is smooth both in terms of graph structure and relevant chemical properties, e.g., synthetic accessibility score and molecular weight.

Conclusions

In this work, we have introduced a variational autoencoder for molecular graphs, that is permutation invariant of the nodes labels of the graphs they are trained with, and allow for graphs with different number of nodes and edges. Moreover, the decoder is able to guarantee a set of local structural and functional properties in the generated graphs through masking. Finally, we have shown that our variational autoencoder can also be used to discover valid and diverse molecules with certain desirable properties more effectively than several state of the art methods.

Our work also opens many interesting venues for future work. For eg. in the design of our variational autoencoder, we have assumed graphs to be static, however, it would be interesting to augment our design to dynamic graphs by, *e.g.*, incorporating a recurrent neural network. We have performed experiments on a single real-world application, *e.g.*, automatic chemical design, however, it would be interesting to explore other applications *e.g.* an end-to-end generative modeling of molecules with specified properties.

Acknowledgements. B. Samanta was supported by a Google India Ph.D. Fellowship and the "Learning Representations from Network Data" project sponsored by Intel. P. K. Chattaraj would like to thank DST, New Delhi for the J.C.Bose National Fellowship.

References

Dai, H.; Tian, Y.; Dai, B.; Skiena, S.; and Song, L. 2018. Syntax-directed variational autoencoder for structured data. In *ICLR*.

Gómez-Bombarelli, R.; Duvenaud, D.; Hernández-Lobato, J. M.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2016. Automatic chemical design using a data-driven continuous representation of molecules. *arXiv preprint arXiv:1610.02415*.

Hamilton, W.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *NIPS*.

Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; and Coleman, R. G. 2012. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling* 52(7):1757–1768.

Jin, W.; Barzilay, R.; and Jaakkola, T. 2018. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*.

Jones, D. R.; Schonlau, M.; and Welch, W. J. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13(4):455–492.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kipf, T. N., and Welling, M. 2016a. Semi-supervised classification with graph convolutional networks.

Kipf, T. N., and Welling, M. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kusner, M. J.; Paige; Brooks; and Hernández-Lobato, J. M. 2017. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*.

Lei, T.; Jin, W.; Barzilay, R.; and Jaakkola, T. 2017. Deriving neural architectures from sequence and graph kernels. *ICML*.

Liu, Q.; Allamanis, M.; Brockschmidt, M.; and Gaunt, A. L. 2018. Constrained graph variational autoencoders for molecule design. *arXiv preprint arXiv:1805.09076*.

Merz, K. M.; Ringe, D.; and Reynolds, C. H. 2010. *Drug design: structure-and ligand-based approaches*. Cambridge University Press.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Paul, S. M.; Mytelka, D. S.; Dunwiddie, C. T.; Persinger, C. C.; Munos, B. H.; Lindborg, S. R.; and Schacht, A. L. 2010. How to improve r&d productivity: the pharmaceutical industry's grand challenge. *Nature reviews Drug discovery* 9(3):203.

Pham, T.; Tran, T.; Phung, D. Q.; and Venkatesh, S. 2017. Column networks for collective classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2485–2491.

Polishchuk, P. G.; Madzhidov, T. I.; and Varnek, A. 2013. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design* 27(8):675–679.

Ramakrishnan, R.; Dral, P. O.; Rupp, M.; and Von Lilienfeld, O. A. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data* 1:140022.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Ruddigkeit, L.; Van Deursen, R.; Blum, L. C.; and Reymond, J.-L. 2012. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling* 52(11):2864–2875.

Simonovsky, M., and Komodakis, N. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*.

Snelson, E., and Ghahramani, Z. 2006. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*.