CONDITIONAL GENERATION OF MOLECULES FROM DISENTANGLED REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Though machine learning approaches have shown great success in estimating properties of small molecules, the inverse problem of generating molecules with desired properties remains challenging. This difficulty is in part because the set of molecules which have a given property is structurally very diverse. Treating this inverse problem as a conditional distribution estimation task, we draw upon work in learning disentangled representations to learn a conditional distribution over molecules given a desired property, where the molecular structure is encoded in a continuous latent random variable. By including property information as an input factor independent from the structure representation, one can perform conditional molecule generation via a “style transfer” process, in which we explicitly set the property to a desired value at generation time. In contrast to existing approaches, we disentangle the latent factors from the property factors using a regularization term which constrains the generated molecules to have the property provided to the generation network, no matter how the latent factor changes.

1 INTRODUCTION

Conditional molecule generation is far from being solved. The main challenge is the enormous and discrete nature of the molecules space and the fact that molecule properties are highly sensitive to molecular structure (Kirkpatrick & Ellis, 2004). Approaches to conditional generation are typically two-step, either using a model or genetic algorithm to generate candidates which are later filtered, or learning a continuous embedding of the discrete molecules and optimizing in a real-valued representation space. The former is computationally expensive, the latter performs conditional generation only very obliquely.

We propose a conditional generative model that produces candidate molecules which targeting a desired property in a single step. This approach builds on work in structured deep generative models (Kingma et al., 2014; Siddharth et al., 2017), which aim to learn a disentangled representation that factors into observed properties we want to control for, and latent factors that account for the remaining features which are either hard to annotate or irrelevant to the properties we wish to optimize.

We derive a regularizer for supervised variational autoencoders which exploits property information that we provide as supervision, ensuring that produced molecules adhere to target properties they are conditioned on. We demonstrate the ability of our model to perform accurate conditional molecule generation and a sort of “style transfer” on molecules, where a latent representation for a single molecule can have its target properties perturbed independently of its learnt structural characteristics, allowing direct and efficient generation of candidates for local optimization of molecules.

2 BACKGROUND

Molecule discovery tasks come in two flavors. Global optimization seeks to find molecules that have a particular target property. Local optimization starts from some initial molecule and searches for molecules which have a desired property while not straying too far from the prototype. There is some overlap in methods used in the two approaches.
2.1 DEEP GENERATIVE MODELS FOR MOLECULES

Virtual screening methods start from a large database of possible molecules and retain the promising ones (Eckert & Bajorath, 2007), as measured by some quality function \( f(\cdot) \). Machine learning approaches expand on this by dynamically generating additional candidate molecules; Segler et al. (2017) uses a stacked LSTM to produce large numbers of novel molecules which have similar characteristics to an existing database.

For properties which are expensive to evaluate, generating large sets of candidate molecules is not particularly useful. More sample-efficient global search can be achieved using Bayesian optimization methods, which use a generative model with a latent space that functions as a continuous representation of molecules (Gómez-Bombarelli et al., 2016; Kusner et al., 2017). Optimization is then carried out over this continuous representation space to find candidates which are expected to have the desired property. Local gradient-based search can also be applied on continuous latent spaces to optimize the latent representation with respect to a target property (Jin et al., 2018; Liu et al., 2018).

A challenge for these latent variable models is to reliably produce valid molecules. Character variational autoencoders (CVAEs) (Gómez-Bombarelli et al., 2016) generate molecules one character at a time, and are prone to syntactic and semantic errors; the grammar-based variational autoencoder (GVAE) (Kusner et al., 2017) and syntax-directed variational autoencoder (SD-VAE) (Dai et al., 2017) instead operate in the space of context-free and attribute grammars, respectively, to ensure syntactic validity. Other work generative models that operates on graph representations (Simonovsky & Komodakis, 2018; De Cao & Kipf, 2018; Jin et al., 2018; You et al., 2018; Liu et al., 2018), largely improving the ability to generate valid molecules.

Suppose we are given a training set of pairs \( \mathcal{D} = \{ (x_i, y_i) \}, i = 1, \ldots, N \), where \( x \) corresponds to molecules and \( y \) represents a value of some properties of the molecule \( x \). Assume the molecules represent an i.i.d. sample from some unknown distribution \( p(x) \), which assigns high probability to molecules believed to be useful for a given task. Aside from Segler et al. (2017), which has no latent space and thus directly trains via maximum likelihood, these latent variable models are trained by optimizing a standard ELBO objective for variational autoencoders (Kingma & Welling, 2014). This entails learning a stochastic encoder \( q_\phi(z|x) \) which maps molecules into a latent space, and a stochastic decoder \( p_\theta(x|z) \) for reconstructing molecules, by maximizing

\[
\mathcal{L}(\theta, \phi) = \sum_{i=1}^{N} \left\{ E_{q_\phi(z|x_i)}[\log p_\theta(x_i|z_i)] - D_{KL}(q_\phi(z|x_i)||p(z_i)) \right\}.
\]

Notably, the objective is not a function of \( y \): most existing generative models with latent variables do not perform direct conditional generation, and approaches for targeted molecule discovery are bolted on to the learnt model. Some, e.g. Kusner et al. (2017), are trained in an “unsupervised” manner, agnostic to any property which later may need to be optimized. Others, e.g. Gómez-Bombarelli et al. (2016; Liu et al., 2018), train the autoencoder jointly alongside a function to predict \( y \) from \( z \), hoping to guide the latent space to be also good for predicting the desired property. A recent exception is Assouel et al. (2018), which learns a deterministic autoencoder where the decoder takes the latent code and the desired property as input, using a mutual information term in training to steer the model towards generating molecules whose target properties match the input. Guimaraes et al. (2017; De Cao & Kipf, 2018; You et al., 2018) instead learn generation models optimized towards specific metrics, such as drug-likeliness and solubility; the major downside is that these models must be retrained each time for a new property. In contrast, the autoencoder-based methods can be re-used to optimize towards any particular value of the property.

2.2 STYLE TRANSFER WITH SUPERVISED VAEs

While the latent representations learned through standard VAE models perform well on the task of molecule reconstruction they do not necessarily provide interpretable factorised representations. A disentangled representation gives us additional control on the molecule generation process, allowing us to modify a single property leaving the remaining unaffected (Bengio et al., 2013a). In many cases important variation in the data is easy to annotate. For example in the case of molecule datasets we have access to different functional descriptors of the molecules obtained by chemoinformatics software such as RDKit (Landrum). Particularly useful to us here are supervised methods for learning disentangled representations (Kingma et al., 2014; Siddharth et al., 2017). These are distinct from
unsupervised disentangling approaches such as InfoGAN (Chen et al., 2016) or β-VAE (Higgins et al., 2017), which encourages the latent factor to learn a disentangled representation by modifying the objective to promote component independence.

We will learn representations that specifically disentangle molecular properties of interest which we may later want to modify. [Kingma et al., 2014] demonstrates how disentangling can be used to take two MNIST images of different digits, written in different styles, and independently change the digit while holding the style constant. An analogous operation on molecules would involve holding the physical structure of a molecule (its “style”) relatively fixed while modifying a salient property. Unlike (say) the style transfer example for the MNIST digits, the conditional distribution of molecules with a particular value of properties might be very diverse; for example, the QED score attempts to measure the drug-likeness of a molecule, and the set of molecules generated at high values of this score would hopefully have high probability on a large, varied set of molecules. An essential challenge here is that the property only provides a very weak signal as to the overall structure of the molecule. To account for this diversity, we model the conditional distribution with a latent variable $z$, such that $p_θ(x|y) = \int p_θ(x|z, y)p(z)dz$.

Disentangling the latent code $z$ from the property $y$ enables style transfer. This is done by taking an initial $x$, computing the posterior over the latent variable $z$, and then generating a new $x'$ with the property modified to have a target value $y'$, with $p_θ(x'|y', x) = \int p_θ(x'|y', z)p_θ(z|x)dz$.

Concretely, this involves fitting a generative model of the form $p_θ(x, y, z) = p_θ(x|y, z)p(y)p(z)$, in which $y$ and $z$ are independent under the prior, and we assume a unit multivariate normal prior $p(z)$. To infer the latent variable $z$ we will use a variational distribution $q_φ(z|x)$, which takes the form of a multivariate normal distribution with parameters a nonlinear function of $x$, to approximate the true posterior $p_θ(z|x, y)$. This objective function

$$L_{ELBO}(θ, φ) = \sum_{i=1}^{N} \left\{ E_{q_φ(z_i|x_i)}[\log p_θ(x_i|y_i, z_i)] - D_{KL}(q_φ(z_i|x_i)||p(z_i)) \right\}$$

(2)

corresponds to learning a supervised VAE [Kingma et al., 2014], and represents a fairly naïve approach to modeling a conditional distribution.

3 CONDITIONAL GENERATION BY DISENTANGLING

Maximizing this conditional ELBO in Eq (2) will likely yield good reconstructions of molecules from an embedding $z$ (alongside the true property $y$), but for properties which only weakly inform the generative model there is nothing to enforce that the variable $y$ actually directly has an effect on the generative process. Since the value $y$ is something we know is a derived property of the molecule $x$, it is completely possible for all information about $y$ to also be encoded in the representation $z$, in which case there is no guarantee that the learnt likelihood $p_θ(x|y, z)$ actually takes into account the value of $y$ — in fact, we know it is possible to fit variational autoencoders where the decoder simply has the form $p_θ(x|z)$ — and we are relying on the utility of $y$ in reconstructions to see any sort of disentangling effect.

3.1 CONSTRAINED ELBO

In the case of conditional generation of molecules, we often have access to some oracle function $f$ which for any given $x$ outputs a property estimate $y$, for instance, the chemoinformatics software RDKit [Landrum]. Since for conditional generation our ultimate goal is to generate a molecule $x$ for any given target property $y_0$, which then actually has $f(x) = y_0$, we can reframe the problem by introducing hard constraints on the generated values, i.e. if restricting to values of $y$ in the training set,

$$\max_{θ, φ} L_{ELBO}(θ, φ)$$

subject to $E_{x\sim p(x|y_i)}[|f(x) - y_i|] = 1$
for all \(i = 1, \ldots, N\). This is an unreasonably hard constraint, unlikely to be satisfied by any distribution other than one which simply places a point mass on the single training \(x_i\) associated with \(y_i\), but we can relax it by considering that (unlike the molecular space \(x\)) the property space \(y\) is typically smooth, as many properties are continuous-valued and correspond to a human-interpretable scale. Following [Ma et al., 2018] and [Hu et al., 2017], we reframe the constraint as a soft penalty on the ELBO,

\[
\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{ELBO}}(\theta, \phi) - \frac{\lambda_1}{2} \sum_{i=1}^{N} \mathbb{E}_{x \sim p_{\theta}(x|y_i)} \| f(\hat{x}) - y_i \|^2
\]

so that they are consistent with the property prediction, i.e., as we have an oracle function \(f\) which enable us to access the property of any generated data, we can explicitly add a soft constraint to our loss function to provide explicit guidance for the generative model such that \(f(\hat{x}) = y\). This constraint is expected to hold for any pair \((x, y)\) we may happen to come across, not just those in the training data. We also show optimizing the relaxed constraint is equivalent to maximizing mutual information with the target \(y_i\) and generated molecule \(x\); for details see appendix Section 6.1.

### 3.2 Approximating the Property Predictor

Introducing the regularizer as in Eq. (3) implicitly guides the reconstruction to take into account the property information, such that the reconstructed data should exhibit properties which match the input properties it is conditioned on. However, existing implementations of \(f\) are often non-differentiable or CPU-bound, and \(\hat{x}\) are discrete samples from a categorical distribution, all of which means the gradient of the regularizer can’t flow back to the generator. This is outlined in Figure 2. To enable the gradient based methods on GPUs during training and avoid discrete sampling, one approach would be to first fit a differentiable approximation to \(f\), and then use either a Gumbel-softmax relaxation [Jang et al., 2016] or tricks like a “straight-through” estimator [Bengio et al., 2013b] as a continuous approximation for the discrete samples. Instead, we propose bypassing the discrete sampling step entirely and learning a function \(f_\omega\) that can map from a learned representation of the molecules directly to molecules property [Hu et al., 2017].

To do this, we take as input the last hidden layer of the decoder network which parameterizes \(p_\theta(x|z, y)\), denoting this deterministic transformation as \(g_\theta(z, y)\). For the grammar VAE and the syntax-directed VAE, this last layer \(h = g_\theta(z, y)\) is the output of a recurrent layer that generates logits corresponding to unmasked and unnormalized log probabilities for each character at each position in the string; see [Kusner et al., 2017] and [Dai et al., 2018] for details on the implementation of the somewhat complex sampling process in the decoder. Ideally, \(f_\omega\) would estimate the property distribution obtained by marginalizing out the discrete sampling step, with

\[
f_\omega(h \equiv g_\theta(z, y_0)) \approx \mathbb{E}_{p_{\theta}(x|z, y_0)}[f(x)],
\]

where we condition on \(z\), and \(y_0\) refers to an arbitrary input target property.
Assuming the approximation in Eq. 4, we have
\[ E_{p_	heta(z|x)}[f(\hat{x}) - y_i]^2 = E_{p(z)}[E_{p_	heta(z|y_i,x)}[f(\hat{x}) - y_i]^2] \approx E_{p(z)}[f_\omega(g_\theta(z, y_i)) - y_i]^2, \]
an expectation over a real-valued variable which does not depend on any of the parameters we are estimating, meaning we can use a simple path estimate of the gradient with respect to \( \theta, \omega \) by exchanging the gradient with the expectation. We thus define a regularization term
\[ L_{\text{disent}}(\theta, \omega) = \frac{\lambda_1}{2} \sum_{i=1}^N E_{p(z)}[f_\omega(g_\theta(z, y_i)) - y_i]^2 \]  
which can be used as a drop-in replacement for the non-differentiable penalty term in Eq. 3, yielding a candidate objective function
\[ L(\theta, \phi) \approx \hat{L}_\omega(\theta, \phi) = L_{\text{ELBO}}(\theta, \phi) - L_{\text{disent}}(\theta, \omega) \]

3.3 Learning the Property Estimator Jointly with Generative Model

While one could imagine attempting to learn \( f_\omega \) jointly with \( \phi, \theta \) by direct optimization of Eq. 6, in practice this is very unstable, as values of \( g_\theta(z, y_i) \) early in training may correspond to very poor generated molecules \( x_i \) which may not have properties at all similar to \( y_i \). This can be sidestepped by training the property estimator jointly as part of an extended generative model on \([x, y]\).

We note that the property estimator \( f_\omega \) parameterizes a probability distribution \( p_\omega(f(x)|z, y_0) \), where \( x \sim p_\theta(x|z, y_0) \) and \( f \) is the oracle function that \( f(x) = y \). With a Gaussian distribution over the error, we can consider
\[ p_\omega(f(x)|z, y_0) = \mathcal{N}(f(x)|f_\omega(g_\theta(z, y_0)), \lambda_2^{-1}I) \]  
for small, fixed \( \lambda_2 \). Therefore, we propose defining a new ELBO based on a joint autoencoder for \( \{f(x_i), y_i\} \), albeit with a factorization such that the input \( y_i \) bypasses the encoder and is passed directly into the decoder, with a joint likelihood
\[ p_{(\theta, \omega)}(x_i, f(x_i)|z_i, y_i) = p_\omega(f(x_i)|z_i, y_i)p_\theta(x_i|z_i, y_i). \]  
This yields a joint ELBO for the training set of
\[ L_{\text{ELBO}}(\omega, \theta, \phi) = \sum_{i=1}^N \mathbb{E}_{q_\phi(z|x)} \left[ \log \frac{p_\omega(f(x_i)|z_i, y_i)p_\theta(x_i|z_i, y_i)p(z)}{q_\phi(z|x_i)} \right]. \]

Note that we can rewrite this ELBO as a function of the previous one, with
\[ L_{\text{ELBO}}(\omega, \theta, \phi) = L_{\text{ELBO}}(\theta, \phi) - \frac{\lambda_2}{2} \sum_{i=1}^N \mathbb{E}_{q_\phi(z|x_i)}[f_\omega(g_\theta(z, y_i)) - y_i]^2, \]
where we also see that \( L_{\text{ELBO}}(\omega, \theta, \phi) \leq L_{\text{ELBO}}(\theta, \phi) \), allowing us to define an objective
\[ \hat{L}(\omega, \theta, \phi) = L_{\text{ELBO}}(\omega, \theta, \phi) - L_{\text{disent}}(\theta, \omega), \]
which is a lower bound on Eq. 6. Notice the two terms we have added to the original ELBO are quite similar, differing only in choice of distribution: for learning \( f_\omega \), we wish to use values of \( z \) simulated form the approximate posterior \( q_\phi(z|x) \), whereas for enforcing a constraint across all possible generations we simulate \( z \) from the prior \( p(z) \).

3.4 Gradient Estimation

As the regularizer \( L_{\text{disent}}(\theta, \omega) \) encourages disentangling by constraining the molecules generated from \( y_i \) to have property \( y_i \); no matter what value \( z \) takes, we found that it does not necessarily evaluate at meaningful values of \( z \) when sampled randomly from \( p(z) \). This roughly corresponds to the notion that not all combinations of "style" and property are physically attainable; ideally for style transfer we would like the generated molecule to stay "close" in structure to the original molecule that we intended to modify. When estimating (gradients of) the soft constraint term \( L_{\text{disent}}(\theta, \omega) \), we found it advantageous to use samples of \( z \) which correspond to encodings of actual data points, as
We also implemented supervised VAEs versions of SD-VAE which we denote Sup-VAE-X-GRU.

We explore its conditional generation performance in two settings: controlling only the property value

We evaluate the reconstruction accuracy and the quality of the molecules generated by our method,

We use the same encoder and decoder network structure as Dai et al. (2018) with the only difference

We represent molecules using the one-hot encoding of their SMILES production rules (Kusner et al.,

We experiment with the QM9 dataset (Ramakrishnan et al., 2014), that contains 134k molecules

When we use this in estimating gradients of the soft constraint, we can use samples from the same

opposed to random samples from the prior. We approximate expectations with respect to $p(x)$ by

$$p(z) = \int p_\theta(z|x)p_\theta(x)dx \approx \frac{1}{N} \sum_j p_\theta(z|x_j) \approx \frac{1}{N} \sum_j q_\phi(z|x_j),$$

where the first approximation uses the empirical data distribution as an approximation to the model

marginal $p_\theta(x)$, and the second uses our variational posterior approximation $q_\phi(z|x)$. We define this

quantity as $q(z) = \frac{1}{N} \sum_j q_\phi(z|x_j)$, a mixture of Gaussians, which we can sample from by drawing

random values from our dataset and then drawing from their encoding distributions.

When we use this in estimating gradients of the soft constraint, we can use samples from the same

minibatch, exactly corresponding to a property transfer task. That is, for any particular $y_i$ in the

dataset, we can estimate

$$\mathbb{E}_{p(x)} \nabla_{\theta,\omega} \| f_\omega(g_\theta(z, y_i)) - y_i \|^2 \approx \mathbb{E}_{q(z|x_i)} \nabla_{\theta,\omega} \| f_\omega(g_\theta(z, y_i)) - y_i \|^2.$$

for any uniformly randomly sampled $j \neq i$. This can be evaluated easily by simply evaluating

the penalty term of Eq. (10) twice per minibatch; once as in Eq. (10), and once to approximate

$L_{\text{div}}(\theta, \omega)$ by permuting the properties in the minibatch to be assigned to incorrect molecules. We

detail the training algorithm in Section 6.2 of the appendix.

4 EXPERIMENTS

We experiment with the QM9 dataset (Ramakrishnan et al., 2014), that contains 134k molecules with up to 9 heavy atoms, and the ZINC dataset (Sterling & Irwin, 2015) containing 250k drug-like molecules. Our goal here is two-fold: we would like to understand (1) whether a supervised variational autoencoder is capable of learning suitable conditional distributions over molecules, and (2) to what extent this task is assisted by the additional regularization term corresponding to the soft constraint.

We represent molecules using the one-hot encoding of their SMILES production rules (Kusner et al., 2017) and add a semantic constraint (Dai et al., 2018) on the decoder network to avoid generating syntactically correct but semantically invalid molecules. We use 80 production rules to describe

like molecules. Our goal here is two-fold: we would like to understand (1) whether a supervised

conditional generation from $p_\theta(x|y_0)$ given the target property $y_0$. Instead of marginalizing over the prior

Table 1: Reconstruction performance and generation quality (Valid, Unique, Novel).

<table>
<thead>
<tr>
<th>Model</th>
<th>QM9 Reconstruction %</th>
<th>QM9 Valid %</th>
<th>QM9 Unique %</th>
<th>QM9 Novel %</th>
<th>ZINC Reconstruction %</th>
<th>ZINC Valid %</th>
<th>ZINC Unique %</th>
<th>ZINC Novel %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVAE-Gómez-Bombarelli et al. 2016</td>
<td>3.64</td>
<td>10.3</td>
<td>-</td>
<td>90.0</td>
<td>44.6</td>
<td>0.7</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>GVAE-Kusner et al. 2017</td>
<td>96.0</td>
<td>60.2</td>
<td>-</td>
<td>80.9</td>
<td>53.7</td>
<td>7.2</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>SD-VAE-Dai et al. 2018</td>
<td>97.84</td>
<td>98.40</td>
<td>99.28</td>
<td>91.97</td>
<td>76.2</td>
<td>43.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sup-VAE-1-GRU</td>
<td>97.53</td>
<td>95.61</td>
<td>93.65</td>
<td>87.87</td>
<td>88.64</td>
<td>29.0</td>
<td>99.24</td>
<td>100</td>
</tr>
<tr>
<td>CGD-VAE-1-GRU</td>
<td>99.27</td>
<td>96.61</td>
<td>93.65</td>
<td>87.87</td>
<td>88.64</td>
<td>29.0</td>
<td>99.24</td>
<td>100</td>
</tr>
<tr>
<td>Sup-VAE-3-GRU</td>
<td>97.81</td>
<td>97.9</td>
<td>95.09</td>
<td>89.47</td>
<td>82.4</td>
<td>36.16</td>
<td>86.26</td>
<td>100</td>
</tr>
<tr>
<td>CGD-VAE-3-GRU</td>
<td>99.31</td>
<td>97.8</td>
<td>98.77</td>
<td>96.21</td>
<td>81.8</td>
<td>37.78</td>
<td>98.75</td>
<td>100</td>
</tr>
</tbody>
</table>
We evaluate reconstruction performance in terms of the correctly reconstructed molecules on test sets of size 10k for QM9 and 5k for ZINC, for the latter we used the default test set. We evaluate the generated molecules’ quality by the percentage of valid, unique (i.e. percentage of unique molecules among the generated valid molecules) and novel (i.e. percentage of molecules never seen in the training set among the generated molecules) molecules. We estimate these quantities by sampling 10k (5k for ZINC) \( z \) from the \( \hat{q}_\sigma(z) \) and coupling each one of them with a logP value, \( y \), randomly selected from the test set, and we subsequently decode the \( z, y \) concatenation. We can see that our model has a better reconstruction performance compared to the baselines while in some cases generating slightly less valid molecules. As we can see our method has a considerably higher correlation score between the input and the obtained property than Sup-V AE. Conditional generation for our method and the Sup-V AE baselines. As we can see our method has a considerably higher correlation score between the input and the obtained property than Sup-V AE. Conditional generation for our method and the Sup-V AE baselines.

We randomly sample 1000 \( y \) values from the test set and 1000 \( z \) values from the approximate learned prior \( \hat{q}_\sigma(z) \). We decode each pair, obtain \( \hat{x} \sim p_\theta(x|y, z) \), and then measure the correlation of the original \( y \) with the \( \hat{y} \) of generated \( \hat{x} \). In Table 2 we give the correlation estimates for our method and the Sup-VAE baselines. As we can see our method has a considerably higher correlation score between the input and the obtained property than Sup-VAE. Conditional generation seems considerably harder for the ZINC dataset for all methods.

To visualise the style transfer behavior of our model we randomly sample two molecules \( x_A, x_B \) from the test set. We then sample \( z_A \) from the learned posterior \( q_\phi(z|x_A) \). We subsequently decode \( [z_A, y_B], y_B \) is the property of \( x_B \), and get a new molecule \( x_{AB} \). Ideally, the obtained molecule \( x_{AB} \) should have a property value (logP) close to the target \( y_B \) and be similar to \( x_A \). In Figure 4 we give one such example. To put the results into context in Figure 8 in appendix, we give the results of a virtual screening method, where we select from the full dataset five molecules which are structurally similar to \( x_A \) and have logP values close to \( y_B \). As we can see the molecule that our model generates is a new one.

To quantify the style transfer performance we proceed in exactly the same manner as we did to quantify the conditional generation performance. However, now instead of sampling \( z \) from the approximate learned prior, \( \hat{q}_\sigma(z) \), we first sample some \( x \) from the test set and then we sample \( z \).
Figure 5: Style transfer. The $z$ of the nine real molecules placed in the x-axis is combined with 11 $y$ property values, sampled in $[-4.9, 4.9]$, the resulting pair is decoded to a molecule.

Table 2: Correlation between the desired input property and the obtained property. $z \sim \hat{q}(z)$ corresponding to conditional generation, and $x, z \sim q(z|x)$ to property transfer case.

| Model     | $z \sim \hat{q}(z)$ | $x \sim q(z|x)$ |
|-----------|----------------------|-----------------|
| QM9       | 0.5420               | 0.2526          |
| Sup-VAE-1-GRU | 0.7185               | 0.5005          |
| Sup-VAE-3-GRU | 0.6958               | 0.4204          |
| CGD-VAE-3-GRU | 0.7414               | 0.4715          |
| ZINC      | 0.2301               | 0.0481          |
| Sup-VAE-1-GRU | 0.3877               | 0.0880          |
| Sup-VAE-3-GRU | 0.3514               | 0.1808          |
| CGD-VAE-3-GRU | 0.3966               | 0.1559          |

Figure 6: Property optimization. Given a start molecule (red), we combine its $z$ with 1000 logP values and decode (blue).

from the learned posterior $q(z|x)$. The results are in the second column of Table 2. As we can see the correlation values are now lower than the ones we obtained in the simple conditional generation case. This can be explained by the fact that now we are forcing a specific combination of structure, $z$ comes from a real molecule, and property, which might simply be physically infeasible since the molecule space is discrete and not all combinations are possible. In addition, as it was the case for the conditional generation, style transfer is considerably more difficult for the ZINC dataset.

We further explore the style transfer and visualise how our model covers the combined space of molecule structure and properties. We sample nine molecules from the QM9 test set and get their $z$ encodings. For each such encoding we decode the vectors $[z, y], y \in [-4.9, 4.9]$, with the $y$ (logP) interval sampled at 11 points. We give in Figure 5 the resulting valid molecules, each column there corresponds to one of the nine original molecules and their decodings with different logP values. As we can see not all $(z, y)$ combinations produce a result. These holes can be explained either by the physical infeasibility of the combination and/or a limitation of the learned model.

We can use conditional generation to control in a fine manner the value of the desired property, to what can be seen as direct property optimization. We visualise the level of control we have on an experiment with a single molecule (with logP is -1.137), which we randomly sample from the test
Figure 7: A comparison of simulated logP values and Tanimoto similarity to a target on the ZINC dataset. While the stacked LSTM model has high accuracy in terms of matching the desired property, it would require drawing many samples before finding any close matches to any particular desired prototype. The CGD-VAE-3-GRU model represents a middle ground between a standard VAE model which does not condition on the property, and the stacked LSTM model which does not learn a reusable representation.

set. We obtain its \( z \) encoding and perform generations with increased logP taking values in 1000 point grid in \([-1.137, 4.9]\). We then decode \([z, y_i]\) and compute the logP value of the generated molecules. Among the 1000 generated molecules only 19 are unique. We get an increase of logP of a very discrete nature, Figure 6. As already discussed not all combinations of structure and properties are possible. The generated molecules themselves are shown in the supplemental material.

4.1 Conditional LSTM baseline

Finally, we consider a variant of the stacked LSTM model of Segler et al. (2017), with no latent space, where the model is modified to take a target logP value as an additional input at each generation step. This model forms a very strong baseline for many distribution matching tasks (Liu et al., 2018; Brown et al., 2019), though as best we are aware this has never been used directly for conditional generation given a target property. We use a modification of the implementation provided by (Brown et al., 2019) with three layers and default settings, and fit the model by maximum likelihood training on \( p_\theta(x|y) = \prod_{t=1}^{T} p_\theta(x_t|x_{1:t-1}, y) \).

Training on the ZINC dataset, we find the generated molecules from this model have a very high correlation 0.975 with the target logP value, greatly outperforming any of the latent variable models we consider. This suggests that such a model would be very useful for generating candidates globally, but as the model has no latent variable it is not amenable to style transfer. We observe this in Figure 7 which samples 100 candidate molecules from both the stacked LSTM model and for CGD-VAE-3-GRU, conditioning on the property of one randomly-chosen test set example, while computing the Tanimoto similarity (computed using Morgan fingerprints of radius 2) to a second randomly-chosen test set example, across 200 pairs. The VAE has higher Tanimoto similarities as it can condition on the latent variable of the target molecule, representing a trade-off against the better adherence to the target property value of the unconditioned LSTM.

5 Conclusion

We presented a single step approach for the conditional generation of molecules with desired properties. Our model allows also to condition generation on a prototype molecule with a desired high-level structure. This work thus directly inverts the traditional relationship between molecules and their properties. We found that training the deep generative models conditional on target properties, following a supervised VAE approach, does not appreciably harm the quality of the unconditional generative model as measured by validity, novelty, and uniqueness of samples. Furthermore, we see that the additional act of regularizing the output using an approximate property predictor helps improve both reconstruction accuracy and property correlations in most combinations of tasks and datasets, particularly for the smaller QM9 dataset and for smaller models with fewer RNN layers. We also note that although none of the deep latent variable models are competitive with an LSTM baseline when purely considering generation conditioned on a target property value, the low Tanimoto similarity between randomly sampled candidates and an arbitrary style transfer target makes clear
that such a model is not suitable for targeted generation of candidates which are close in structure to a particular prototype.

In future work, we want to explore how to further improve the correlation between the desired input properties to the decoder, and the properties of the generated molecules. Moreover, we want also to condition on multiple properties; while this is in principle possible in our framework, we do not explore it empirically here. Modifying a single property while constraining the remaining to be close to the original can further aggravate the infeasibility problem, as not all combinations of molecular properties may even be feasible, perhaps requiring learning a dependency structure between multiple properties.

REFERENCES


6 Appendix

6.1 The regularizer and its relation to the mutual information maximization

The soft constrain in the loss[3], in fact, is equivalent to a simple maximizing mutual information formulation between generated molecules $\hat{x}$ and the target property $y$ provided to the generator. Assume the true conditional distribution is $p(y|x)$:

$$I(y; \hat{x}) = H(y) - H(y|\hat{x})$$

$$= H(y) + E_{x \sim p(x|y)}E_{y' \sim \hat{p}(y|x)}[\log \hat{p}(y'|\hat{x})]$$

We do not know the true distribution $\hat{p}(y|x)$, however, the RDKit provides an estimation $p(y|x)$ of the distribution assuming a Gaussian distribution over error:

$$p(y|x) = \mathcal{N}(y|f(x), \lambda^{-1}I)$$

where $f$ is the molecule property estimator, i.e., RDKit. We have:

$$I(y; \hat{x}) = H(y) + E_{x \sim p(x|y)}E_{y' \sim \hat{p}(y|x)}[\log \hat{p}(y'|\hat{x})]$$

$$= H(y) + E_{x \sim p(x|y)}[D_{KL}(\hat{p}(y'|\hat{x})||p(y'|x))] + E_{x \sim \hat{p}(y|x)} \log p(y'|\hat{x})$$

$$\geq H(y) + E_{x \sim p(x|y)}[D_{KL}(\hat{p}(y'|\hat{x})||p(y'|x))]$$

By following the Lemma 5.1 given in [Chen et al., 2016], we have

$$I(y; \hat{x}) \geq H(y) + E_{y \sim p(y), x \sim p(x|y)}[\log p(y|x)]$$

As $H(y)$ is constant, minimizing $E_{x \sim p(x|y)}\|f(x) - y\|^2$ is equivalent to maximizing $I(y; \hat{x})$ under the assumption that $p(y|x)$ is close to $\hat{p}(y|x)$

6.2 Architecture and Training Procedure Description

We use the same encoder and decoder network structure as [Dai et al., 2018] with the only difference that our decoder takes as input the concatenation of $y$, $z$. As GRU layers become computationally expensive when the sequences length increase, we also examined the model using less layer GRU. To be precise, the decoder in [Dai et al., 2018] takes the from of a dense hidden layer with ReLU activation followed by three layers GRU [Chung et al., 2014]. We tried two different settings of decoder: in the first setting, we feed the concatenation of $y$, $z$ to dense layer then apply one layer GRU, in the second setting, to enhance the effect of $y$ in the decoder, we feed $y$ not only to the dense layer but also to each layer of GRU. Furthermore, we set the dimension of the latent representation to 56. For the oracle function estimator $f_w$, we use the same network architecture as the encoder (there is no parameter sharing) and we add one more fully connected layer followed by a Tanh transformation.

To speed up convergence, we initialize the $f_w$ from a pre-training, where we train $f_w$ on the well-trained (maximum 500 epochs with early stopping) supervised VAE’s decoder output to predict the molecules property value. We also initialize the parameters of the encoder/decoder networks with the partially trained supervised VAE model (after 40 epochs for QM9, 100 epochs for ZINC). We do not update $\omega$ and $\phi$, $\theta$ simultaneously, instead we do an alternate optimization. We update $\omega$ continuously for five epochs while holding $\phi$, $\theta$ and do the same for updating $\phi$, $\theta$. We set the hyper-parameter value $\lambda_1$ to 50 and $\lambda_2$ to 1. The mini-batch size is set to 300 for QM9 and 100 for ZINC. We use ADAM optimizer with learning rate 0.0001 and pytorch lr-schedular on the validation loss. The general training algorithm is described in below algorithm block[1]. In our experiment, to train $f_\omega$, we skipped the second term in step 7, which means we only train $f_\omega$ on the training data but not the newly generated molecules obtained by permuting the property. The reason for this is that, during the training, we found that it is easy for the model to learn to reconstruct but hard to conditionally generate the molecules with given properties while we have no guidance of what the molecules should look like. Further more, some combination of $z$ and $y$ are physically not feasible. In this case, when the conditional generation is not good enough yet during the training, we end up fitting $f_\omega$ on the miss represented molecules representations and it makes the optimization harder.
Algorithm 1 Training algorithm

1: Initialize $p_\theta(x|z,y)$, $q_\phi(z|x)$, $f_w$
2: for $i=1,2,\ldots,N$ (maximum epoch number) do
3:     for $j=1,2,\ldots,L$, sample a minibatch $D = (X, Y) = \{x_i, y_i\}_{i=1}^M$ of M samples do
4:         randomly permute the property set $Y$ to obtain $Y^*$ and define a label permuted mini
batch $D^* = (X, Y^*) = \{x_i, y_i^*\}_{i=1}^M$
5:         $\theta^j = \theta^{j-1} - \gamma (-\nabla_\theta \mathcal{ELBO}(\theta, \phi, \omega) + \frac{\lambda_1}{2} \sum_{(x_i, y_i) \in D} \mathbb{E}_{q(z|x_i)} \nabla_\theta \|f_\omega(g_\theta(z, y_i)) - y_i\|^2)$
6:         $\phi^j = \phi^{j-1} - \gamma (-\nabla_\phi \mathcal{ELBO}(\theta, \phi, \omega))$
7:         $w^j = w^{j-1} - \gamma (\frac{\lambda_2}{2} \sum_{(x_i, y_i) \in D} \mathbb{E}_{q(z|x_i)} \nabla_\omega \|f_\omega(g_\theta(z, y_i)) - y_i\|^2 + \frac{\lambda_1}{2} \sum_{(x_i, y_i) \in D^*} \mathbb{E}_{q(z|x_i)} \nabla_\omega \|f_\omega(g_\theta(z, y_i)) - y_i\|^2)$

6.3 USE A FIXED PRE-TRAINED PROPERTY PREDICTION FUNCTION

We also investigate the case where we train the property prediction function $f_\omega$ on the well trained
supervised VAE output and keep it fixed during the training of the main model. We give the
performance in tables 3, 4. Using a fixed $f_\omega$, in terms of reconstruction and generation performance,
delivers mixed results 3. However, in terms of conditional generation 4, it does perform better than
the baselines but worse compared to the case where we also update $f_\omega$ during the learning.

<table>
<thead>
<tr>
<th>Model</th>
<th>QM9</th>
<th>ZINC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reconstruction %</td>
<td>Valid %</td>
</tr>
<tr>
<td>CV AE</td>
<td>3.61</td>
<td>10.3</td>
</tr>
<tr>
<td>GVAE</td>
<td>96.0</td>
<td>60.2</td>
</tr>
<tr>
<td>SD-VAE</td>
<td>97.84</td>
<td>98.40</td>
</tr>
<tr>
<td>Sup-VAE-1-GRU</td>
<td>97.53</td>
<td>93.66</td>
</tr>
<tr>
<td>CGD-VAE-1-GRU</td>
<td>98.96</td>
<td>95.03</td>
</tr>
<tr>
<td>Sup-VAE-3-GRU</td>
<td>97.81</td>
<td>97.9</td>
</tr>
<tr>
<td>CGD-VAE-3-GRU</td>
<td>96.9</td>
<td>93.8</td>
</tr>
</tbody>
</table>

Table 3: Sup-VAE-1-GRU /Sup-VAE-3-GRU: supervised version of SD-VAE model where $y$ is been
feed to only the first layer/all layer decoder. CGD-VAE: our model, conditional generation with
disentangling. The result for CV AE and GVAE are taken from the literature, "-" refers that those
measures are not reported. The rest of baseline result is obtained by rerunning (SD-VAE) and editing
the original code (Sup-VAE).

<table>
<thead>
<tr>
<th>Model</th>
<th>QM9</th>
<th>ZINC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>z ~ $\hat{q}_\phi(z)$</td>
<td>z ~ $q(z</td>
</tr>
<tr>
<td>QM9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sup-VAE-1-GRU</td>
<td>0.5420</td>
<td>0.2526</td>
</tr>
<tr>
<td>CGD-VAE-1-GRU</td>
<td>0.6331</td>
<td>0.3835</td>
</tr>
<tr>
<td>Sup-VAE-3-GRU</td>
<td>0.6958</td>
<td>0.4204</td>
</tr>
<tr>
<td>CGD-VAE-3-GRU</td>
<td>0.6665</td>
<td>0.4507</td>
</tr>
</tbody>
</table>

| ZINC          |       |       |
| Sup-VAE-1-GRU | 0.2301 | 0.0481 |
| CGD-VAE-1-GRU | 0.2981 | 0.0866 |
| Sup-VAE-3-GRU | 0.3638 | 0.1818 |
| CGD-VAE-3-GRU | 0.3765 | 0.1310 |

Table 4: Correlation between the desired input property and the obtained property when $z$ is sampled
from the approximate learned prior, $z \sim \hat{q}_\phi(z)$, (conditional generation), and when $z$ is sampled from
the learned posterior given some $x$, $z \sim q(z|x)$, (property transfer).
6.4 Virtual Screening

The figure 8 displays the results of a virtual screening method, where we select from the full dataset five molecules which are structurally similar to $x_A$ in figure 4 in section 6 and have logP values close to $y_B$. As we can see the molecule that our model generates is a new one.

![Figure 8: Molecules selected with virtual screening over the full dataset in a manner that they are structurally similar to the A molecule of figure 4 while they have a logP value which is close to the logP value of the B molecule also of figure 4.](image)

6.5 Metrics as function of $y - y'$

The validity and novelty were mainly used to assess purely the generative model performance. However, it is also interesting to see if the model capable of generating valid and novel molecules if we start from an existing molecules and drift away from the original label, i.e., if we get $z$ from $q(z|x)$, and then compare different values from $p(x|z, y')$ as $y'$ moves far from $y$. We randomly sample a molecule $x$ whose LogP is $y$ from the test set, then sample 10 $z$ from $q(z|x)$. For each such $z$ we couple it with a $y'$ that is different that $y$, and sample 10 molecules from $p(x|z, y')$. Eventually, for each such $y'$, starting from original molecule $x$, we generated 100 molecules, and we report the validity, uniqueness and novelty as a function of $y - y'$. The figure 9 displays result of repeating above process for 20 randomly sampled $(x, y)$ along 100 grid points for $y - y'$. The result confirms that on a big data set, conditional generative models uniqueness, validity and novelty performance is not affected by the size of the modification done on the property. However, on a small dataset, uniqueness is not affected. As expected, novelty increases as the properties modification size increases and validity drops slightly as the property modification size increase.

![Figure 9: CGD-VAE-3-GRU model validity, novelty, uniqueness performance on property transfer task as a function of $y - y'$ on QM9 dataset.](image)

![Figure 10: CGD-VAE-3-GRU model validity, novelty, uniqueness performance on property transfer task as a function of $y - y'$ on ZINC dataset.](image)

6.6 Sampling from approximated marginal posterior for generation

With our model, during the generation, we observe that sampling from the approximated marginal posterior improves generation performance when compared to sampling from the prior. Here we
investigate if this findings holds for other baseline models or not. We explored the behavior of baselines when the $z$ is sampled from the approximate marginal posterior, we observe that for CVAE and GVAE the validity did not change (as the $\hat{q}(z)$ and $p(z)$ are essentially identical); for the SD-VAE the validity increases (Table 5).

<table>
<thead>
<tr>
<th>SDVAE</th>
<th>Valid%</th>
<th>Unique %</th>
<th>Novel %</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z \sim \hat{q}_v(z)$</td>
<td>56.05</td>
<td>80.69</td>
<td>100</td>
</tr>
<tr>
<td>$z \sim p(z)$</td>
<td>43.50</td>
<td>82.26</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5: Baseline model performance on ZINC

6.7 Molecule Property Optimization

Figure 11 is the visualization of the generated molecules from property optimization task, given in figure 6 section 4. The molecules are generated by increasing the logP of a given molecule, i.e. we hold $z$ fixed and increase the $y$ value. Among the all generated molecules, 19 of them are unique.

Figure 11: Molecules generated with an increasing logP