JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING
Article
Cite This: *J. Chem. Inf. Model.* 2019, 59, 43−52
pubs.acs.org/jcim

Conditional Molecular Design with Deep Generative Models

Seokho Kang*,[†] and Kyunghyun Cho[‡,¶,§]

[†]Department of Systems Management Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon 16419, Republic of Korea

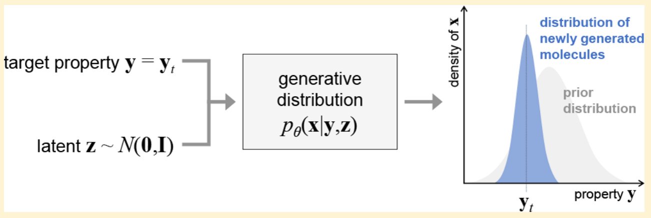
[‡]Department of Computer Science & Center for Data Science, New York University, 60 5th Avenue, New York, New York 10011, United States

[¶]Facebook AI Research, 770 Broadway, New York, New York 10003, United States

[§]CIFAR Azrieli Global Scholar, Canadian Institute for Advanced Research, 661 University Avenue, Toronto, ON M5G 1M1, Canada

Supporting Information

ABSTRACT: Although machine learning has been successfully used to propose novel molecules that satisfy desired properties, it is still challenging to explore a large chemical space efficiently. In this paper, we present a conditional molecular design method that facilitates generating new molecules with desired properties. The proposed model, which simultaneously performs both property prediction and molecule generation, is built as a semisupervised variational autoencoder trained on a set of existing molecules with only a partial annotation. We generate new molecules with desired properties by sampling from the generative distribution estimated by the model. We demonstrate the effectiveness of the proposed model by evaluating it on drug-like molecules. The model improves the performance of property prediction by exploiting unlabeled molecules and efficiently generates novel molecules fulfilling various target conditions.


target property $\mathbf{y} = \mathbf{y}_t$
latent $\mathbf{z} \sim N(\mathbf{0},\mathbf{I})$
generative distribution $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$
density of $\mathbf{x}$
distribution of newly generated molecules
prior distribution
$\mathbf{y}_t$
property $\mathbf{y}$

■ INTRODUCTION

The primary goal of molecular design is to propose novel molecules that satisfy desired properties, which has been challenging due to the difficulty in efficiently exploring a large chemical space. In the past, molecular design has been largely driven by human experts. They would suggest candidate molecules that are then evaluated through computer simulations and subsequent experimental syntheses.[1] This is time-consuming and costly and is inadequate when many candidate molecules must be considered. In recent decades, machine learning based approaches have been actively studied as efficient alternatives to expedite the molecular design processes.[2−4]

A conventional approach is to build a prediction model that estimates the properties of a given molecule, as shown in Figure 1a. Molecules with desired properties are then chosen after screening a possible set of candidate molecules by this prediction model.[5] The candidate molecules for screening need to be manually obtained from such sources as combinatorial enumerations of possible fragments[6−8] and public databases.[9,10] It is necessary to secure a sufficient amount of molecules that are properly labeled with properties, as prediction accuracy typically depends on the number of labeled molecules and the quality of the labels. Early work has attempted to transform a molecule into a hand-engineered feature representation, so-called molecular fingerprint, and to use it as an input to predict the properties.[11−13] With recent advances of deep learning,[14] prediction quality has improved by employing deep neural networks.[15,16] Moreover, various recent studies have extracted features directly from graph representations of molecules to better predict the properties.[17−22]

Another approach aims to automatically generate new molecules by building a molecule generation model, as in Figure 1b. This approach learns to map a molecule on a latent space. From this space, it randomly generates new molecules that are analogous to those in the original set. Molecules randomly generated by this model can be used as candidates for screening with a separate property prediction model. Most existing studies on this approach have represented a molecule as a simplified molecular-input line-entry system (SMILES)[23] with a recurrent neural network (RNN). They include RNN language models,[24−26] variational autoencoders (VAEs),[27−30] and generative adversarial networks (GANs)[31,32] with RNN decoders. More recently, the models directly generating the graph structures of molecules have also been proposed.[33−36]

The molecule generation approach has been extended to conditional molecular design, generating a new molecule whose properties are close to a predetermined target condition. This is often done by finding a latent representation that closely reflects the target condition using a property prediction model, as in Figure 1c. Previous work has proposed to use recursive fine-tuning,[24] Bayesian optimization,[28] and reinforce-

Received: April 30, 2018
Published: July 17, 2018

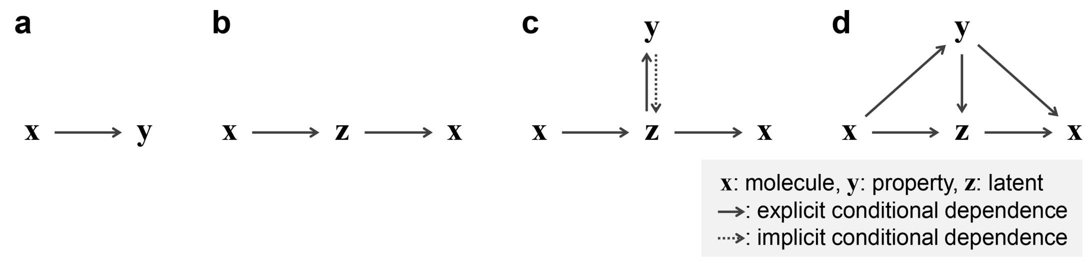

a
b
c
d
x
y
z
x: molecule, y: property, z: latent
→: explicit conditional dependence
⇢: implicit conditional dependence

Figure 1. Schematic diagram of machine learning applications to molecular design: (a) property prediction, (b) molecule generation, (c) conditional molecular design (previous work), and (d) conditional molecular design (present work).

ment learning.[25,26,32] These methods generate molecules with intended properties not directly but by an additional optimization procedure often in the latent space. This is inefficient especially when multiple target conditions are considered.

Here we present a novel approach to efficiently and accurately generating new molecules satisfying designated properties. We build a conditional molecular design model that simultaneously performs both property prediction and molecule generation, as illustrated in Figure 1d, using a semisupervised variational autoencoder (SSVAE).[37] Given a set of specific properties, conditional molecular design is done by directly sampling new molecules from a conditional generative distribution without any extra optimization procedure. The semisupervised model can effectively exploit unlabeled molecules. This is advantageous particularly when only a small portion of molecules in the data are labeled with their properties, which is usual due to the expensive cost of labeling molecules.

METHODS

Model Architecture. We adapt the original SSVAE[37] to incorporate continuous output variables. SSVAE is a directed probabilistic graphical model that captures the data distribution in a semisupervised manner. In the generative process of the SSVAE model, the input variable $\mathbf{x}$ is generated from a generative distribution $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$, which is parametrized by θ conditioned on the output variable $\mathbf{y}$ and latent variable $\mathbf{z}$. $\mathbf{y}$ is treated as an additional latent variable when $\mathbf{x}$ is not labeled, which necessitates introducing the distribution over $\mathbf{y}$. The prior distributions over $\mathbf{y}$ and $\mathbf{z}$ are assumed to be $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_\mathbf{y}, \boldsymbol{\Sigma}_\mathbf{y})$ and $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$. We use variational inference to address the intractability of the exact posterior inference of the model. We approximate the posterior distributions over $\mathbf{y}$ and $\mathbf{z}$ by

$$q_\phi(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}_\phi^2(\mathbf{x})))$$

and

$$q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\phi(\mathbf{x}, \mathbf{y}), \mathrm{diag}(\boldsymbol{\sigma}_\phi^2(\mathbf{x}, \mathbf{y})))$$

both of which are parametrized with ϕ. For the semisupervised learning scenario where some values of $\mathbf{y}$ are missing, the missing values are predicted by $q_\phi(\mathbf{y}|\mathbf{x})$.

In our framework for conditional molecular design, $\mathbf{x}$ and $\mathbf{y}$ denote a molecule and its continuous-valued properties, respectively. In this study, we consider molecules that can be represented by SMILES, which has been commonly used in the recent related work.[24−26,28,29,32] SMILES encodes the graph structure of a molecule in a compact line notation by depth-first traversal into a sequence with a simple vocabulary and grammar rules.[23] For example, a benzene is described in the form of SMILES as c1ccccc1. A molecule representation $\mathbf{x}$ is then formed as a sequence of one-hot vectors describing a SMILES string $(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(j)}, ..., \mathbf{x}^{(l)})$, where each one-hot vector $\mathbf{x}^{(j)}$ corresponds to the index of a symbol in a predefined vocabulary and l is the length of the sequence. The vocabulary consists of all unique characters in the data, except for atoms represented by two characters (e.g., Si, Cl, Br, and Sn) which are considered single symbols. A vector $\mathbf{y}$ consists of m scalar values $(y^1, ..., y^m)$, where m is the number of properties of a molecule.

We use an RNN to model p_θ and q_ϕ. The SSVAE model is composed of three RNNs, which are the predictor network $q_\phi(\mathbf{y}|\mathbf{x})$, the encoder network $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$, and the decoder network $p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$. We use bidirectional RNNs[38] for the predictor and encoder networks, while the decoder network is a unidirectional RNN. The input to the encoder network at each time step j contains $\mathbf{x}^{(j)}$ and $\mathbf{y}$. The decoder network, which generates sequences, takes the output of the current time step j, $\mathbf{y}$, and $\mathbf{z}$ as the input at time $j + 1$.

Objective Functions. We define two loss functions $\mathcal{L}(\mathbf{x}, \mathbf{y})$ and $\mathcal{U}(\mathbf{x})$ corresponding to labeled and unlabeled instances, respectively. The variational lower bound $-\mathcal{L}(\mathbf{x}, \mathbf{y})$ of the log-probability of a labeled instance $(\mathbf{x}, \mathbf{y})$ is

$$\begin{aligned}\log p(\mathbf{x}, \mathbf{y}) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) + \log p(\mathbf{y}) \\ &\quad + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})] + \log p(\mathbf{y}) \\ &\quad - \mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})\|p(\mathbf{z})) \\ &= -\mathcal{L}(\mathbf{x}, \mathbf{y})\end{aligned} \tag{1}$$

For an unlabeled instance $\mathbf{x}$, $\mathbf{y}$ is considered as a latent variable. The variational lower bound $-\mathcal{U}(\mathbf{x})$ is then:

$$\begin{aligned}\log p(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) + \log p(\mathbf{y}) + \log p(\mathbf{z}) \\ &\quad - \log q_\phi(\mathbf{y}, \mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})] - \mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{y}|\mathbf{x})\|p(\mathbf{y})) \\ &\quad - \mathbb{E}_{q_\phi(\mathbf{y}|\mathbf{x})}[\mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})\|p(\mathbf{z}))] \\ &= -\mathcal{U}(\mathbf{x})\end{aligned} \tag{2}$$

where $\log q_\phi(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \log q_\phi(\mathbf{y}|\mathbf{x}) + \log q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$.

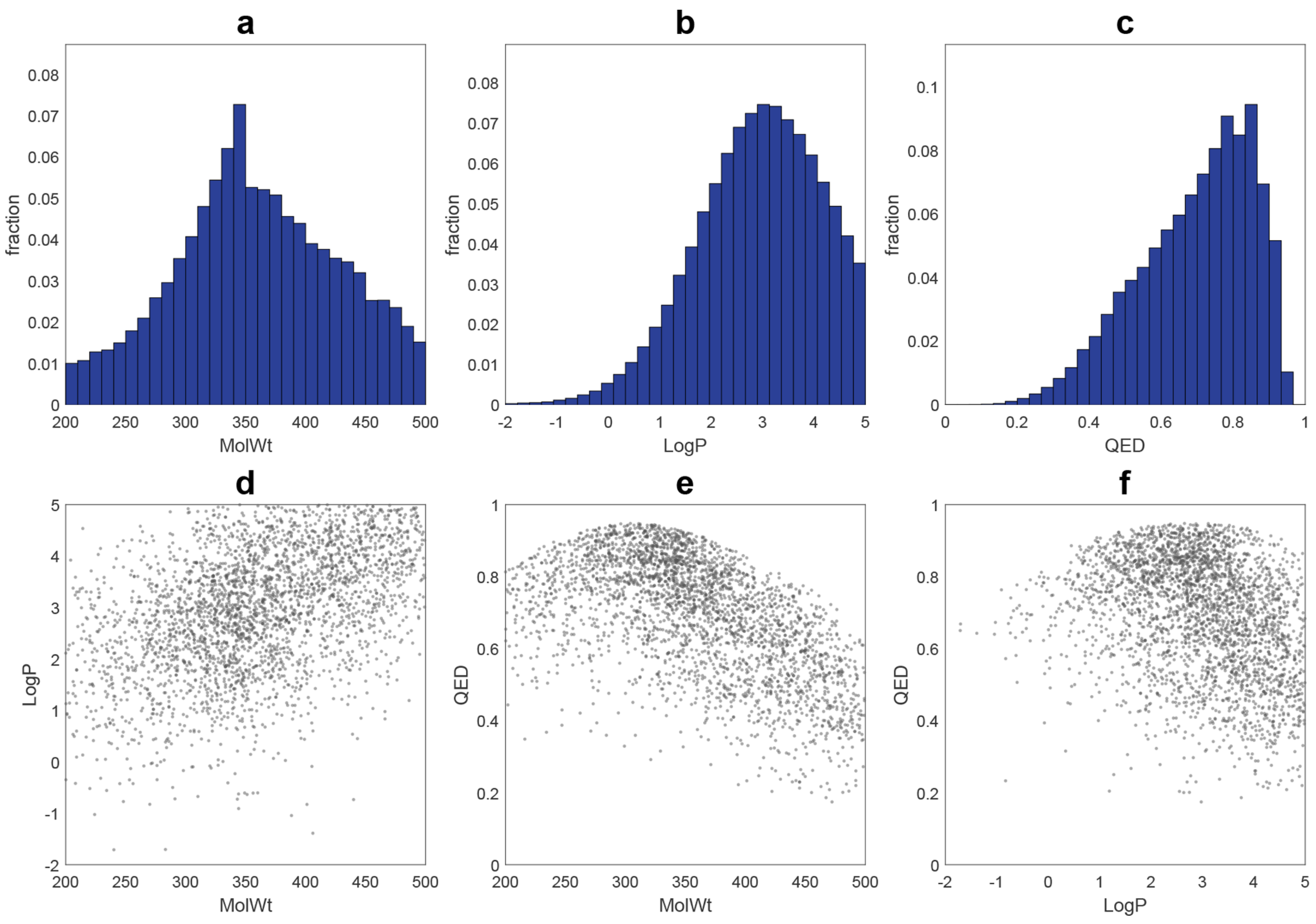


Figure 2. Distribution of properties in training set: histogram of (a) MolWt, (b) LogP, and (c) QED; scatterplot between (d) MolWt and LogP, (e) MolWt and QED, and (f) LogP and QED.

Given the data distributions of labeled $\tilde{p}_l(\mathbf{x}, \mathbf{y})$ and unlabeled cases $\tilde{p}_u(\mathbf{x})$, the full loss function $\mathcal{J}$ for the entire data set is defined as

$$\mathcal{J} = \sum_{(\mathbf{x},\mathbf{y})\sim\tilde{p}_l} \mathcal{L}(\mathbf{x}, \mathbf{y}) + \sum_{(\mathbf{x})\sim\tilde{p}_u} \mathcal{U}(\mathbf{x}) + \beta\cdot \sum_{(\mathbf{x},\mathbf{y})\sim\tilde{p}_l} \|\mathbf{y} - \mathbb{E}_{q_\phi(\mathbf{y}|\mathbf{x})}[\mathbf{y}]\|^2 \tag{3}$$

where the last term is mean squared error for supervised learning. The distribution $q_\phi(\mathbf{y}|\mathbf{x})$ is not estimated from the labeled cases without the last term, because $q_\phi(\mathbf{y}|\mathbf{x})$ does not contribute to $\mathcal{L}(\mathbf{x}, \mathbf{y})$.[37] The last term encourages $q_\phi(\mathbf{y}|\mathbf{x})$ to be predictive of the observed properties based on the labeled instances. As $\mathbf{y}$ is assumed to follow a normal distribution, $\mathbb{E}_{q_\phi(\mathbf{y}|\mathbf{x})}[\mathbf{y}]$ is equivalent to $\boldsymbol{\mu}_\phi(\mathbf{x})$. The hyper-parameter β controls the trade-off between generative and supervised learning. It becomes fully generative learning when $\beta = 0$, while it focuses more on supervised learning with a larger β.

Property Prediction. Once the SSVAE model is trained, property prediction is performed using the predictor network $q_\phi(\mathbf{y}|\mathbf{x})$. Given an unlabeled instance $\mathbf{x}$, the corresponding properties $\hat{\mathbf{y}}$ are predicted as below.

$$\hat{\mathbf{y}} \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}_\phi^2(\mathbf{x}))) \tag{4}$$

The point estimate of $\hat{\mathbf{y}}$ can be obtained by maximizing the probability, which is equivalent to $\boldsymbol{\mu}_\phi(\mathbf{x})$.

Molecule Generation. We use the decoder network $p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$ to generate a molecule. A molecule representation $\hat{\mathbf{x}}$ is obtained from $\mathbf{y}$ and $\mathbf{z}$ by

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \log p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) \tag{5}$$

At each time step j of the decoder, the output $\mathbf{x}^{(j)}$ is predicted by conditioning on all the previous outputs $(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(j-1)})$, $\mathbf{y}$, and $\mathbf{z}$, because we decompose $p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$ as

$$p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) = \prod_j p_\theta(\mathbf{x}^{(j)}|\mathbf{x}^{(1)}, ..., \mathbf{x}^{(j-1)}, \mathbf{y}, \mathbf{z}) \tag{6}$$

The optimal decoding solution $\hat{\mathbf{x}}$ can be obtained by maximizing the autoregressive distribution of $p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$. This is however computationally intractable, because the search space grows exponentially with respect to the length of sequences. Sampling from the autoregressive distribution is simple and fast, but is vulnerable to the noise in sequence generation. Therefore, we use beam search to find an approximate solution efficiently, which has been successfully used to generate sequences with RNN.[39−41] Beam search generates a sequence from left to right based on a breadth-first tree search mechanism. At each time step j, top-K candidates are maintained.

To generate an arbitrary molecule unconditionally, $\mathbf{y}$ and $\mathbf{z}$ are sampled from their prior distributions $p(\mathbf{y})$ and $p(\mathbf{z})$, respectively. For conditional molecular design given a target value for a property, $\mathbf{z}$ is sampled from $p(\mathbf{z})$, while the corresponding element of $\mathbf{y}$ is set to the target value and the other elements are sampled from the conditional prior distribution given the target value. For example, if we want to generate a new molecule whose first property is close to 0.5, the first element y_1 is set to 0.5 while the other elements are sampled from $p(y_2, ..., y_m|y_1 = 0.5)$.

RESULTS AND DISCUSSION

Data Set. We collect 310 000 SMILES strings of drug-like molecules randomly sampled from the ZINC database.[9] We use 300 000 molecules for training and the remaining 10 000 molecules for testing the property prediction performance. The SMILES strings of the molecules are canonicalized using the RDKit package[42] and, then, are transformed into sequences of symbols occurring in the training set. The vocabulary contains 35 different symbols including {1, 2, 3, 4, 5, 6, 7, 8, 9, + , −, =, #, (,), [,], H, B, C, N, O, F, Si, P, S, Cl, Br, Sn, I, c, n, o, p, s}. The minimum, median, and maximum lengths of a SMILES string are 8, 42, and 86, respectively. A special symbol indicating the end of a sequence is appended at the end of each sequence.

It is time-consuming and costly to directly obtain the chemical properties of numerous newly generated molecules by performing first-principles calculations or experimental syntheses. In order to efficiently evaluate the proposed approach, we use the three properties that can be readily calculated using the RDKit package:[42] molecular weight (MolWt), Wildman−Crippen partition coefficient (LogP),[43] and quantitative estimation of drug-likeness (QED).[44] Figure 2 shows the distributions of these properties in the training set. In this figure, the histograms plot the distributions of individual properties, and the scatterplots represent the pairwise distributions of the properties on 3000 randomly selected molecules. MolWt ranges from 200 to 500 g/mol, and LogP has the range of [−2, 5], according to the drug-like criteria of the ZINC database. QED is valued from 0 to 1 by definition. The averages of MolWt, LogP, and QED are 359.019, 2.911, and 0.696, and their standard deviations are 67.669, 1.179, and 0.158, respectively. There is a positive correlation between MolWt and LogP with the correlation coefficient of 0.434, whereas QED is negatively correlated with both MolWt and LogP with the correlation coefficients of −0.548 and −0.298, respectively.

Experiments. We evaluate the SSVAE model against baseline models in terms of the prediction performance. We vary the number of labeled molecules (5%, 10%, 20%, and 50% of the training set) to investigate its effect on property prediction. 95% of the training set is used for training, while the remaining 5% is used for early stopping. During training, we normalize each output variable to have a mean of 0 and standard deviation of 1. We use backpropagation with the Adam optimizer.[45] We set the default learning rate to 0.001 and use a batch size of 200. Training is terminated if the validation error failed to decrease by 1% over ten consecutive epochs or the number of epochs reached 300. The property prediction performance is evaluated by mean absolute error (MAE) on the test set.

For the SSVAE model, its predictor, encoder, and decoder networks consist of three hidden layers each having 250 gated recurrent units (GRUs).[46] The dimension of $\mathbf{z}$ is set as 100. For the prior distribution $p(\mathbf{y})$, we estimate the mean vector $\hat{\boldsymbol{\mu}}_{\mathbf{y}}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}_{\mathbf{y}}$ from the labeled molecules in the training set. Figure 3 shows the architectural detail of the SSVAE model. We train the model with both the labeled and

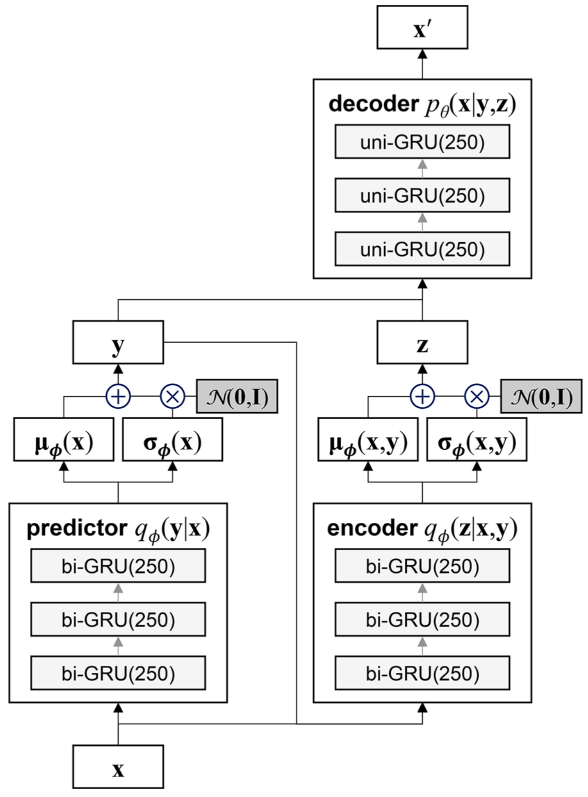

x′
decoder $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$
uni-GRU(250)
uni-GRU(250)
uni-GRU(250)
y
z
$\mathcal{N}(\mathbf{0},\mathbf{I})$
$\mathcal{N}(\mathbf{0},\mathbf{I})$
$\boldsymbol{\mu}_\phi(\mathbf{x})$
$\boldsymbol{\sigma}_\phi(\mathbf{x})$
$\boldsymbol{\mu}_\phi(\mathbf{x},\mathbf{y})$
$\boldsymbol{\sigma}_\phi(\mathbf{x},\mathbf{y})$
predictor $q_\phi(\mathbf{y}|\mathbf{x})$
bi-GRU(250)
bi-GRU(250)
bi-GRU(250)
encoder $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$
bi-GRU(250)
bi-GRU(250)
bi-GRU(250)
x

Figure 3. Illustration of SSVAE model architecture.

unlabeled molecules to minimize the objective function in eq 3. We set β to 10^4 by conducting a preliminary experiment of minimizing the average validation error of property prediction over the varying numbers of labeled molecules.

As baseline models, we use the extended-connectivity fingerprint (ECFP),[11] molecular graph convolutions (GraphConv),[19] independently trained predictor network $q_\phi(\mathbf{y}|\mathbf{x})$, and VAE model jointly trained with a property prediction model that predicts the properties from the latent representation ($\text{VAE}_\text{property}$).[28] In the cases of the ECFP and GraphConv models, a molecule is processed by three hidden layers, each of which consists of 2000, 500, and 500 sigmoid units with a dropout rate of 0.2.[47] It is then followed by a final output layer that predicts the three output variables. The baseline models except the $\text{VAE}_\text{property}$ model are trained only with the labeled molecules in the training set to minimize mean squared error between the actual and predicted properties. For the implementation of the $\text{VAE}_\text{property}$ model,[28] its VAE part is trained with the entire molecules without their labels and the joint prediction model is trained only with the labeled molecules using the same objective function as that of the SSVAE model.

To demonstrate conditional molecular design, we use the SSVAE model trained on the training set in which 50% of the molecules were labeled with their properties. New molecules are generated under various target conditions of properties, each of which sets one property with a specific target value and the others to be sampled from the corresponding conditional prior distribution.

Table 1. Property Prediction Performance with Varying Fractions of Labeled Molecules

frac. labeled	property	ECFP	GraphConv	predictor network	$VAE_{property}$	SSVAE
5%	MolWt	17.713 ± 0.396	6.723 ± 2.116	2.582 ± 0.288	3.463 ± 0.971	1.639 ± 0.577
	LogP	0.380 ± 0.009	0.187 ± 0.015	0.162 ± 0.006	0.125 ± 0.013	0.120 ± 0.006
	QED	0.053 ± 0.001	0.034 ± 0.004	0.037 ± 0.002	0.029 ± 0.002	0.028 ± 0.001
10%	MolWt	15.057 ± 0.358	5.255 ± 0.767	1.986 ± 0.470	2.464 ± 0.581	1.444 ± 0.618
	LogP	0.335 ± 0.005	0.148 ± 0.016	0.116 ± 0.006	0.097 ± 0.008	0.090 ± 0.004
	QED	0.045 ± 0.001	0.028 ± 0.003	0.027 ± 0.002	0.021 ± 0.002	0.021 ± 0.001
20%	MolWt	12.047 ± 0.168	4.597 ± 0.419	1.228 ± 0.229	1.748 ± 0.266	1.008 ± 0.370
	LogP	0.249 ± 0.004	0.112 ± 0.015	0.070 ± 0.007	0.074 ± 0.006	0.071 ± 0.007
	QED	0.033 ± 0.001	0.021 ± 0.002	0.017 ± 0.002	0.015 ± 0.001	0.016 ± 0.001
50%	MolWt	9.012 ± 0.184	4.506 ± 0.279	1.010 ± 0.250	1.350 ± 0.319	1.050 ± 0.164
	LogP	0.180 ± 0.003	0.086 ± 0.012	0.045 ± 0.005	0.049 ± 0.008	0.047 ± 0.003
	QED	0.023 ± 0.000	0.018 ± 0.001	0.011 ± 0.001	0.009 ± 0.002	0.010 ± 0.001

Table 2. Molecule Generation Efficacy of Conditional Molecular Design

model	target condition	no. generated	no. invalid	no. in training set	no. duplicated	no. new unique
$VAE_{unsupervised}$	uncond. gen. (sampling)	10000 (100%)	8771 (87.7%)	5 (0.1%)	65 (0.7%)	1159 (11.6%)
	uncond. gen. (beam search)	10000 (100%)	2 (0.0%)	1243 (12.4%)	6802 (68.0%)	1953 (19.5%)
$VAE_{property}$	unconditional generation	5940 (100%)	2 (0.0%)	486 (8.2%)	2452 (41.3%)	3000 (50.5%)
	MolWt = 250	10000 (100%)	7 (0.1%)	1136 (11.4%)	6400 (64.0%)	2457 (24.6%)
	MolWt = 350	8618 (100%)	1 (0.0%)	647 (7.5%)	4970 (57.7%)	3000 (34.8%)
	MolWt = 450	10000 (100%)	9 (0.1%)	1120 (11.2%)	6626 (66.3%)	2245 (22.5%)
	LogP = 1.5	9521 (100%)	10 (0.1%)	575 (6.0%)	5936 (62.3%)	3000 (31.5%)
	LogP = 3.0	7628 (100%)	4 (0.1%)	560 (7.3%)	4064 (53.3%)	3000 (39.3%)
	LogP = 4.5	10000 (100%)	13 (0.1%)	862 (8.6%)	6563 (65.6%)	2562 (25.6%)
	QED = 0.5	9643 (100%)	20 (0.2%)	764 (7.9%)	5859 (60.8%)	3000 (31.1%)
	QED = 0.7	6888 (100%)	3 (0.0%)	617 (9.0%)	3268 (47.4%)	3000 (43.6%)
	QED = 0.9	10000 (100%)	6 (0.1%)	851 (8.5%)	6476 (64.8%)	2667 (26.7%)
SSVAE	unconditional generation	3236 (100%)	23 (0.7%)	163 (5.0%)	50 (1.5%)	3000 (92.7%)
	MolWt = 250	4079 (100%)	16 (0.4%)	177 (4.3%)	886 (21.7%)	3000 (73.5%)
	MolWt = 350	3629 (100%)	17 (0.5%)	137 (3.8%)	475 (13.1%)	3000 (82.7%)
	MolWt = 450	4181 (100%)	31 (0.7%)	277 (6.6%)	873 (20.9%)	3000 (71.8%)
	LogP = 1.5	3457 (100%)	26 (0.8%)	127 (3.7%)	304 (8.8%)	3000 (86.8%)
	LogP = 3.0	3433 (100%)	21 (0.6%)	166 (4.8%)	246 (7.2%)	3000 (87.4%)
	LogP = 4.5	3507 (100%)	30 (0.9%)	186 (5.3%)	291 (8.3%)	3000 (85.5%)
	QED = 0.5	3456 (100%)	49 (1.4%)	171 (4.9%)	236 (6.8%)	3000 (86.8%)
	QED = 0.7	3308 (100%)	19 (0.6%)	168 (5.1%)	121 (3.7%)	3000 (90.7%)
	QED = 0.9	3233 (100%)	12 (0.4%)	125 (3.9%)	96 (3.0%)	3000 (92.8%)

We compare the SSVAE model with the unsupervised VAE model ($VAE_{unsupervised}$) and the $VAE_{property}$ model that are trained on the same training set. In the case of the $VAE_{property}$ model, we use Gaussian process to smoothly approximate the property prediction given a latent representation and perform Bayesian optimization.[28] The objective function for Bayesian optimization is set as the normalized absolute difference between the target value and the value predicted from the latent representation by the joint property prediction model. For generating a molecule, Bayesian optimization is terminated when the value of the objective function is below 0.01.

Molecules are generated from the decoder network of the model using beam search, where the beam width K is set to 5. The target values for MolWt, LogP, and QED are set as {250, 350, 450}, {1.5, 3.0, 4.5}, and {0.5, 0.7, 0.9}, respectively. We also test generating new molecules unconditionally without specifying any target value. During the generation procedure given each target condition, we check the validity of each generated molecule using the SMILES grammar rules (e.g., the number of open/close parentheses and the existence of unclosed rings) and preconditions (e.g., kekulizability) using the RDKit package.[42] We discard those molecules that are identified as invalid, already existing in the training set, or duplicates. The generation procedure continues until 3000 novel unique molecules are obtained or the number of trials exceeds 10 000. Then, these molecules are labeled with MolWt, LogP, and QED to confirm whether their properties were distributed around their respective target values.

All the experiments are implemented based on GPU-accelerated TensorFlow in Python.[48] The source code of the SSVAE model used in the experiments is available at https://github.com/nyu-dl/conditional-molecular-design-ssvae.

Property Prediction. Table 1 shows the results in terms of MAE with the varying fractions of labeled molecules in the training set. We report the average and standard deviation over ten repetitions for each setting. Among the baseline models, the GraphConv model was superior to the ECFP model in every case. The GraphConv model yielded performance comparable to the predictor model with a fewer labeled molecules, while the predictor model was superior with more labeled molecules. The predictor model significantly outperformed the ECFP and GraphConv models on predicting MolWt, which is almost identical to the task of simply counting atoms in a SMILES string. The $VAE_{property}$ model performed

Table 3. Comparison between Original Training Set and Conditional Molecular Design Results

model	target condition	no. molecules	sequence length	MolWt	LogP	QED
training set	all molecules	300000	42.375 ± 9.316	359.019 ± 67.669	2.911 ± 1.179	0.696 ± 0.158
	labeled molecules	150000	42.402 ± 9.313	359.381 ± 67.666	2.912 ± 1.177	0.696 ± 0.158
	$240 \leq$ MolWt ≤ 260	4868	28.818 ± 3.707	250.237 ± 5.642	2.116 ± 1.074	0.762 ± 0.118
	$340 \leq$ MolWt ≤ 360	18799	41.094 ± 3.865	348.836 ± 5.751	2.793 ± 1.094	0.759 ± 0.123
	$440 \leq$ MolWt ≤ 460	8546	53.562 ± 4.586	448.959 ± 5.631	3.569 ± 0.989	0.540 ± 0.122
	$1.4 \leq$ LogP ≤ 1.6	4591	38.223 ± 8.679	320.299 ± 61.256	1.503 ± 0.057	0.757 ± 0.130
	$2.9 \leq$ LogP ≤ 3.1	9657	42.214 ± 9.044	357.686 ± 62.747	2.999 ± 0.058	0.722 ± 0.150
	$4.4 \leq$ LogP ≤ 4.6	6040	47.228 ± 8.404	404.425 ± 56.184	4.496 ± 0.059	0.589 ± 0.149
	$0.49 \leq$ QED ≤ 0.51	3336	49.028 ± 8.415	409.733 ± 63.362	3.467 ± 1.130	0.500 ± 0.006
	$0.69 \leq$ QED ≤ 0.71	5961	42.614 ± 8.571	362.448 ± 63.359	2.929 ± 1.255	0.700 ± 0.006
	$0.89 \leq$ QED ≤ 0.91	5466	36.910 ± 5.219	316.128 ± 32.789	2.529 ± 0.865	0.900 ± 0.006
$\text{VAE}_{\text{unsupervised}}$	uncond. gen. (sampling)	1159	34.965 ± 8.123	305.756 ± 65.817	2.900 ± 1.262	0.725 ± 0.143
	uncond. gen. (beam search)	1953	43.911 ± 8.384	366.502 ± 60.865	2.987 ± 0.995	0.707 ± 0.149
$\text{VAE}_{\text{property}}$	unconditional generation	3000	43.853 ± 7.477	362.037 ± 54.528	2.986 ± 0.994	0.716 ± 0.132
	MolWt = 250	2457	30.284 ± 4.261	255.676 ± 25.457	2.230 ± 0.965	0.789 ± 0.093
	MolWt = 350	3000	40.718 ± 4.534	338.858 ± 27.478	3.023 ± 0.982	0.766 ± 0.111
	MolWt = 450	2245	53.738 ± 4.636	443.253 ± 23.950	3.477 ± 0.946	0.573 ± 0.108
	LogP = 1.5	3000	40.296 ± 8.019	329.754 ± 58.057	1.478 ± 0.537	0.744 ± 0.117
	LogP = 3.0	3000	42.975 ± 7.352	353.535 ± 53.487	2.740 ± 0.643	0.728 ± 0.127
	LogP = 4.5	2562	46.198 ± 7.871	389.465 ± 53.514	4.290 ± 0.435	0.636 ± 0.136
	QED = 0.5	3000	49.955 ± 6.749	409.021 ± 47.190	3.386 ± 1.023	0.544 ± 0.096
	QED = 0.7	3000	45.331 ± 7.382	375.083 ± 53.888	3.079 ± 1.002	0.688 ± 0.111
	QED = 0.9	2667	37.441 ± 5.573	310.396 ± 38.871	2.515 ± 0.918	0.860 ± 0.062
SSVAE	unconditional generation	3000	42.093 ± 9.010	359.135 ± 65.534	2.873 ± 1.117	0.695 ± 0.148
	MolWt = 250	3000	28.513 ± 3.431	250.287 ± 6.742	2.077 ± 1.072	0.796 ± 0.094
	MolWt = 350	3000	41.401 ± 4.393	349.599 ± 7.345	2.782 ± 1.060	0.723 ± 0.129
	MolWt = 450	3000	53.179 ± 4.760	449.593 ± 8.901	3.544 ± 1.016	0.563 ± 0.122
	LogP = 1.5	3000	38.709 ± 8.669	323.336 ± 60.288	1.539 ± 0.301	0.750 ± 0.127
	LogP = 3.0	3000	42.523 ± 8.919	361.264 ± 61.524	2.984 ± 0.295	0.701 ± 0.149
	LogP = 4.5	3000	45.566 ± 8.698	397.609 ± 61.436	4.350 ± 0.309	0.624 ± 0.147
	QED = 0.5	3000	48.412 ± 7.904	404.159 ± 56.788	3.288 ± 1.069	0.527 ± 0.094
	QED = 0.7	3000	41.737 ± 7.659	356.672 ± 55.629	2.893 ± 1.093	0.719 ± 0.088
	QED = 0.9	3000	36.243 ± 7.689	312.985 ± 56.270	2.444 ± 1.079	0.840 ± 0.070

worse for MolWt but was superior in predicting LogP and QED with a fewer labeled molecules, when compared to the predictor model.

The SSVAE model outperformed the baseline models on most of the cases. The SSVAE model yielded better prediction performance than the predictor model did with a lower fraction of labeled molecules. On the other hand, the difference between the SSVAE model and the predictor model narrowed as the fraction of labeled molecules increased. The results successfully demonstrate the effectiveness of this semi-supervised learning scheme in improving property prediction.

Conditional Molecular Design. Table 2 shows the statistics of generated molecules given each target condition in order to investigate the efficacy of molecule generation. For the SSVAE model, the fraction of invalid molecules was generally less than 1% and was slightly higher when the target value had a lower density in the distribution of the training set. There were a few duplicated molecules from unconditional generation, and the fraction of new unique molecules was 92.7%. There were more duplicates when molecules were conditionally generated. In particular, the fraction of duplicated molecules for a target condition was higher when the prediction of the property for the condition was more accurate. As the normalized MAEs of MolWt, LogP, and QED by the SSVAE model were 0.016, 0.038, and 0.058, MolWt yielded the lowest fraction of new unique molecules and was followed by LogP and QED.

Both the $\text{VAE}_{\text{unsupervised}}$ and $\text{VAE}_{\text{property}}$ models were less efficient than the SSVAE model was, evident from the higher number of duplicated molecules generated. When we tried sampling from the $\text{VAE}_{\text{unsupervised}}$ model without beam search, the model rarely generated duplicated ones, while the majority of the generated ones were invalid.

Table 3 presents the summary statistics for newly generated molecules of each condition, and Figures 4 and 5 compare the histograms representing the distributions of MolWt, LogP, and QED between different target conditions by the SSVAE and $\text{VAE}_{\text{property}}$ models, respectively. For the SSVAE model, the unconditionally generated molecules without any target value followed the property distributions of the training set, as evident from contrasting Figures 4a−c and 2a−c. When a target condition is set, the SSVAE model successfully generated new molecules fulfilling the condition. In Figure 4d−f, we observe that the distributions of the conditionally generated molecules by the SSVAE model were centered around the corresponding target values with much smaller standard deviations. The conditionally generated molecules followed the property distributions of those molecules in the training set whose property was around the target value, as shown in Table 3. The accuracy of conditional molecular design for a target condition tended to be proportional to the

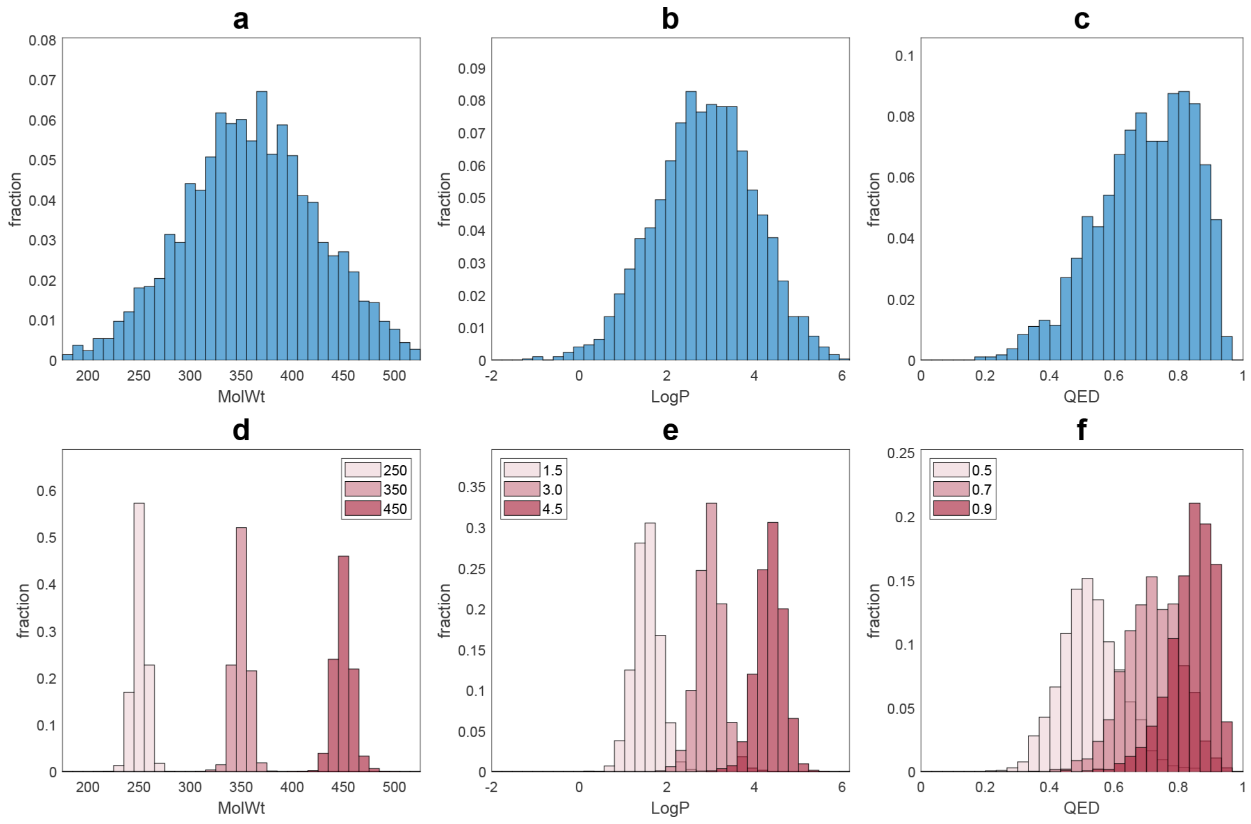


Figure 4. Distribution of properties by conditional molecular design using SSVAE model (proposed): histogram of unconditional generation results for (a) MolWt, (b) LogP, and (c) QED; histogram of conditional generation results for (d) MolWt, (e) LogP, and (f) QED.

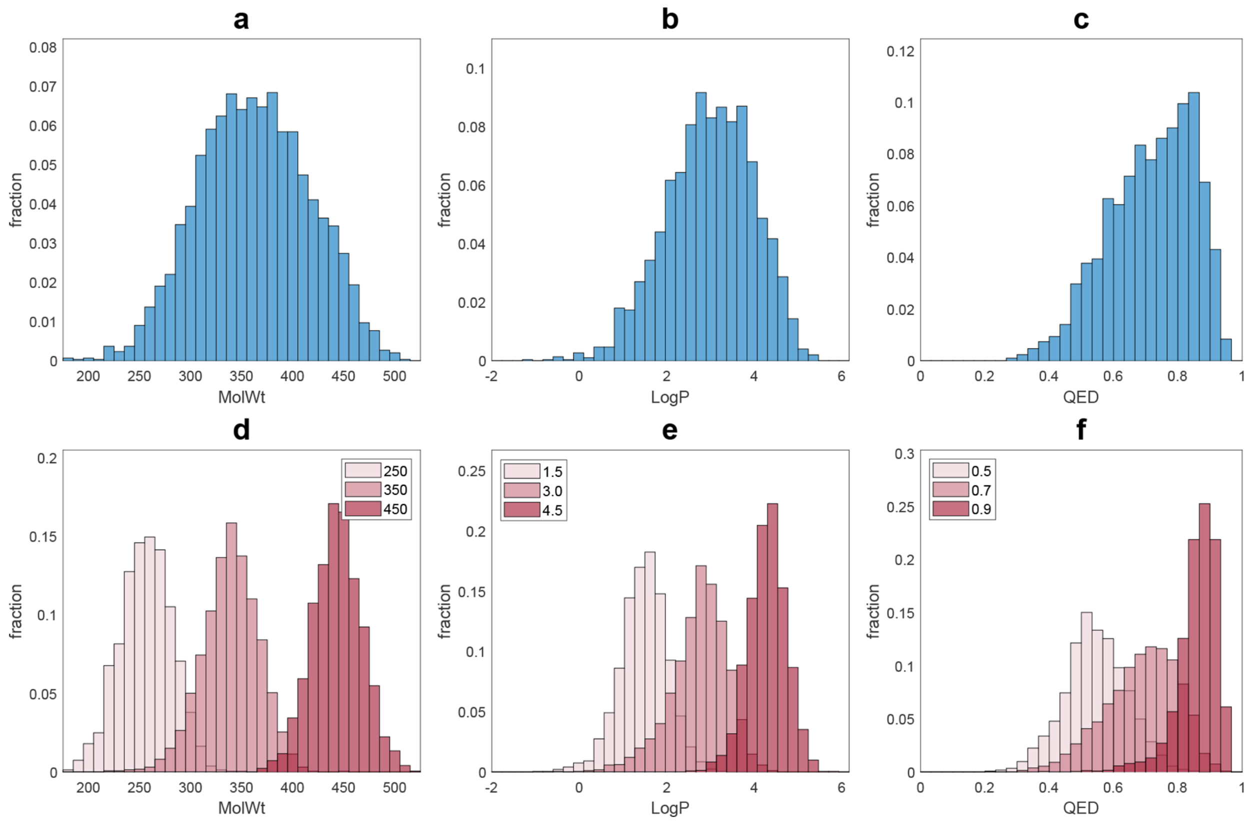


Figure 5. Distribution of properties by conditional molecular design using $VAE_{property}$ model (baseline): histogram of unconditional generation results for (a) MolWt, (b) LogP, and (c) QED; histogram of conditional generation results for (d) MolWt, (e) LogP, and (f) QED.

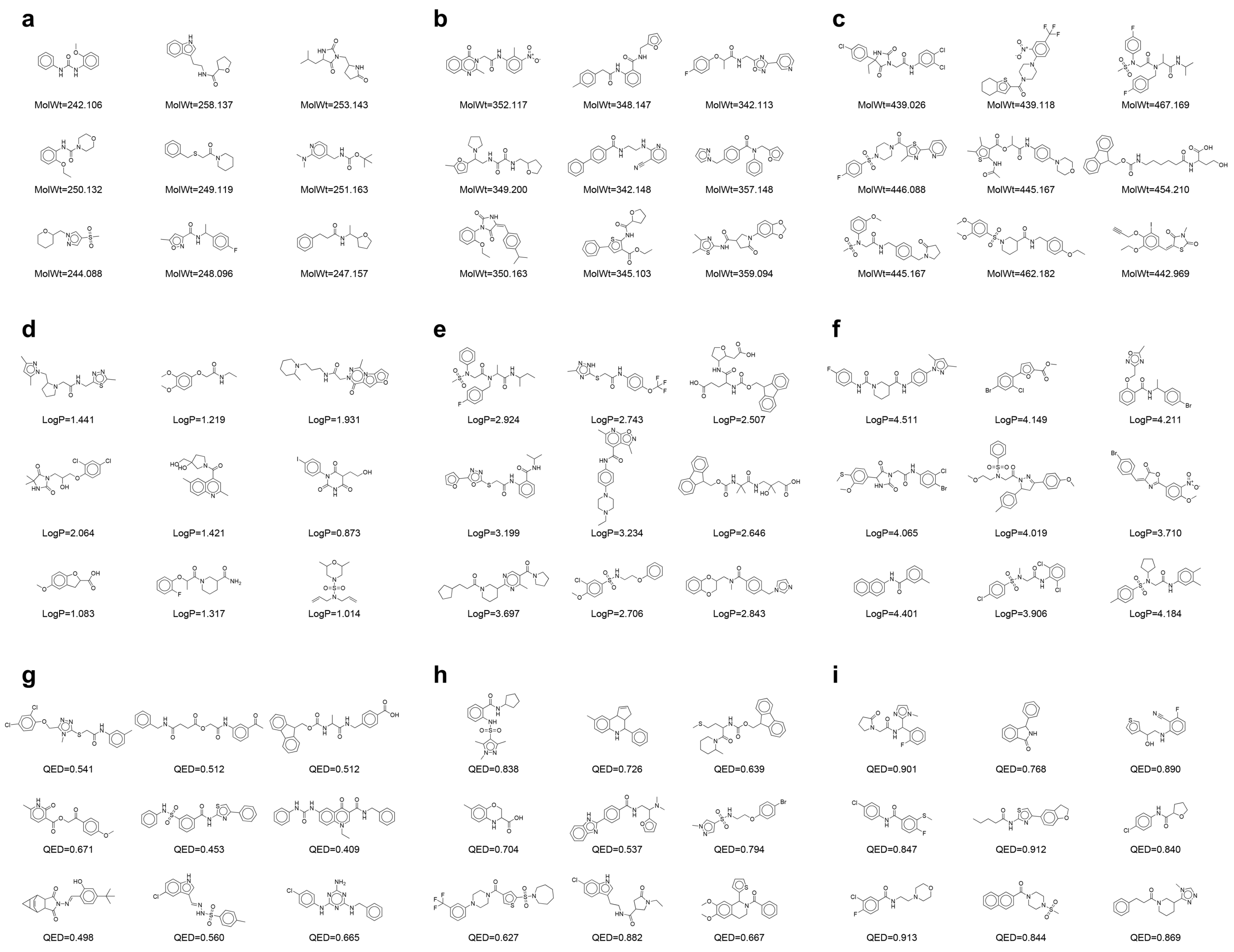


Figure 6. Examples of generated molecules using SSVAE model conditioning on (a) MolWt = 250, (b) MolWt = 350, (c) MolWt = 450, (d) LogP = 1.5, (e) LogP = 3.0, (f) LogP = 4.5, (g) QED = 0.5, (h) QED = 0.7, and (i) QED = 0.9.

prediction accuracy of the corresponding property. For MolWt which yielded the lowest normalized MAE, the distributions were relatively narrow and separated distinctly by its target values. On the other hand, LogP and QED exhibited larger overlap between target values. The VAE$_{\text{property}}$ model also generated new molecules satisfying the target conditions, but the distributions were relatively dispersed and far from the corresponding target values compared to those by the SSVAE model, as shown in Figure 5d−f.

We present some sample molecules generated from the SSVAE model under each target condition in Figure 6. From the glance at the sample molecules generated with three different target MolWt, we observe that the SSVAE had generated smaller molecules when the target condition of MolWT was set to 250. On the other hand, when MolWt was set to a higher value, relatively larger molecules were generated.

Table 4 compares training and inference time between the models. It took longer to train the SSVAE model than the other models, because it has one more RNN as the predictor network compared to the other models. For unconditional generation, there was a slight difference in the generation speed between the models. Conditional generation with the VAE$_{\text{property}}$ model, which involves Bayesian optimization, was time-consuming. On the other hand, conditional generation with the SSVAE model, which simply uses the decoder network without any extra optimization procedure, was as fast as unconditional generation.

Table 4. Training and Inference Time Comparison

		inference time (per generation)	
model	training time	unconditional gen.	conditional gen.
VAE$_{\text{unsupervised}}$	7.4 ± 1.9 h	4.9 ± 0.7 s	
VAE$_{\text{property}}$	10.2 ± 2.1 h	4.7 ± 0.7 s	46.6 ± 135.5 s
SSVAE	20.3 ± 5.3 h	4.6 ± 1.0 s	4.5 ± 1.1 s

CONCLUSION

We have presented a novel approach to conditionally generating molecules efficiently and accurately using the regression version of SSVAE. We designed and trained the SSVAE model on a partially labeled training set in which only a small portion of molecules were labeled with their properties. New molecules with desired properties were generated from the generative distribution of the SSVAE model given a target condition of properties. The experiments using drug-like molecules sampled from the ZINC database have successfully demonstrated the effectiveness in terms of both property prediction and conditional molecular design. The SSVAE model efficiently generates novel molecules satisfying the target conditions without any extra optimization procedure. Moreover, the conditional design procedure works by automatically learning implicit knowledge from data without necessitating any explicit knowledge.

The proposed approach can serve as an efficient tool for designing new chemical structures with a specified target condition. These structures generated as SMILES strings are to be examined further to obtain realistic molecules with desired properties. In this study, the application is limited to only a part of the chemical space that SMILES can represent. To broaden its applicability, we should investigate other alternatives to SMILES that provide higher coverage of the chemical space and are able to represent molecules more comprehensively.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jcim.8b00263.

ChemDraw and CSV formats of SMILES strings for the examples in Figure 6 (ZIP)

List of SMILES strings for the examples in Figure 6 (PDF)

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: s.kang@skku.edu. Phone: +82 31 290 7596. Fax: +82 31 290 7610.

ORCID

Seokho Kang: 0000-0002-0960-0294

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT; Ministry of Science and ICT) (No. NRF-2017R1C1B5075685). K.C. thanks AdeptMind, eBay, Ten-Cent, NVIDIA, and CIFAR for support. K.C. was partly supported for this work by Samsung Electronics (Improving Deep Learning using Latent Structure).

■ REFERENCES

(1) Hautier, G.; Jain, A.; Ong, S. P. From the Computer to the Laboratory: Materials Discovery and Design Using First-Principles Calculations. *J. Mater. Sci.* **2012**, *47*, 7317−7340.

(2) Katritzky, A. R.; Lobanov, V. S.; Karelson, M. QSPR: The Correlation and Quantitative Prediction of Chemical and Physical Properties from Structure. *Chem. Soc. Rev.* **1995**, *24*, 279−287.

(3) Varnek, A.; Baskin, I. Machine Learning Methods for Property Prediction in Chemoinformatics: Quo Vadis? *J. Chem. Inf. Model.* **2012**, *52*, 1413−1437.

(4) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. Big Data Meets Quantum Chemistry Approximations: The Δ-Machine Learning Approach. *J. Chem. Theory Comput.* **2015**, *11*, 2087−2096.

(5) Pyzer-Knapp, E. O.; Suh, C.; Gómez-Bombarelli, R.; Aguilera-Iparraguirre, J.; Aspuru-Guzik, A. What is High-Throughput Virtual Screening? A Perspective from Organic Materials Discovery. *Annu. Rev. Mater. Res.* **2015**, *45*, 195−216.

(6) Huc, I.; Lehn, J.-M. Virtual Combinatorial Libraries: Dynamic Generation of Molecular and Supramolecular Diversity by Self-Assembly. *Proc. Natl. Acad. Sci. U. S. A.* **1997**, *94*, 2106−2110.

(7) Lehn, J.-M. Dynamic Combinatorial Chemistry and Virtual Combinatorial Libraries. *Chem. - Eur. J.* **1999**, *5*, 2455−2463.

(8) Schneider, G. Trends in Virtual Combinatorial Library Design. *Curr. Med. Chem.* **2002**, *9*, 2095−2101.

(9) Sterling, T.; Irwin, J. J. ZINC 15−Ligand Discovery for Everyone. *J. Chem. Inf. Model.* **2015**, *55*, 2324−2337.

(10) Kim, S.; Thiessen, P. A.; Bolton, E. E.; Chen, J.; Fu, G.; Gindulyte, A.; Han, L.; He, J.; He, S.; Shoemaker, B. A.; Wang, J.; Yu, B.; Zhang, J.; Bryant, S. H. PubChem Substance and Compound Databases. *Nucleic Acids Res.* **2016**, *44*, D1202−D1213.

(11) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.

(12) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; Von Lilienfeld, O. A. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.* **2012**, *108* (5), 1−5.

(13) Hansen, K.; Biegler, F.; Ramakrishnan, R.; Pronobis, W.; Von Lilienfeld, O. A.; Müller, K.-R.; Tkatchenko, A. Machine Learning Predictions of Molecular Properties: Accurate Many-Body Potentials and Nonlocality in Chemical Space. *J. Phys. Chem. Lett.* **2015**, *6*, 2326−2331.

(14) LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436−444.

(15) Goh, G. B.; Hodas, N. O.; Vishnu, A. Deep Learning for Computational Chemistry. *J. Comput. Chem.* **2017**, *38*, 1291−1307.

(16) Ma, J.; Sheridan, R. P.; Liaw, A.; Dahl, G. E.; Svetnik, V. Deep Neural Nets as a Method for Quantitative Structure−Activity Relationships. *J. Chem. Inf. Model.* **2015**, *55*, 263−274.

(17) Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional Networks on Graphs for Learning Molecular Fingerprints. *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*; 2015; pp 2224−2232.

(18) Coley, C. W.; Barzilay, R.; Green, W. H.; Jaakkola, T. S.; Jensen, K. F. Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction. *J. Chem. Inf. Model.* **2017**, *57*, 1757−1772.

(19) Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; Riley, P. Molecular Graph Convolutions: Moving Beyond Fingerprints. *J. Comput.-Aided Mol. Des.* **2016**, *30*, 595−608.

(20) Lusci, A.; Pollastri, G.; Baldi, P. Deep Architectures and Deep Learning in Chemoinformatics: The Prediction of Aqueous Solubility for Drug-Like Molecules. *J. Chem. Inf. Model.* **2013**, *53*, 1563−1575.

(21) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural Message Passing for Quantum Chemistry. *Proceedings of the 34th International Conference on Machine Learning*; 2017; pp 1263−1272.

(22) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-Chemical Insights from Deep Tensor Neural Networks. *Nat. Commun.* **2017**, *8*, 13890.

(23) Weininger, D. SMILES, A Chemical Language and Information system. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Model.* **1988**, *28*, 31−36.

(24) Segler, M. H. S.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.* **2018**, *4*, 120−131.

(25) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular De-Novo Design through Deep Reinforcement Learning. *J. Cheminf.* **2017**, *9* (48), 1−14.

(26) Popova, M.; Isayev, O.; Tropsha, A. Deep Reinforcement Learning for De-Novo Drug Design. *arXiv.org* **2017**, No. arXiv:1711.10907.

(27) Kingma, D. P.; Welling, M. Auto-Encoding Variational Bayes. *Proceedings of the 2nd International Conference on Learning Representations*; 2014.

(28) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-

Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **2018**, *4*, 268−276.

(29) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. *Proceedings of the 34th International Conference on Machine Learning*; 2017; pp 1945−1954.

(30) Dai, H.; Tian, Y.; Dai, B.; Skiena, S.; Song, L. Syntax-Directed Variational Autoencoder for Structured Data. *Proceedings of the 6th International Conference on Learning Representations*; 2018.

(31) Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*; 2014; pp 2672−2680.

(32) Guimaraes, G. L.; Sanchez-Lengeling, B.; Farias, P. L. C.; Aspuru-Guzik, A. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv.org* **2017**, No. arXiv:1705.10843.

(33) Simonovsky, M.; Komodakis, N. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *arXiv.org* **2018**, No. arXiv:1802.03480.

(34) Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv.org* **2018**, No. arXiv:1802.04364.

(35) De Cao, N.; Kipf, T. MolGAN: An Implicit Generative Model for Small Molecular Graphs. *arXiv.org* **2018**, No. arXiv:1805.11973.

(36) Samanta, B.; De, A.; Ganguly, N.; Gomez-Rodriguez, M. Designing Random Graph Models Using Variational Autoencoders with Applications to Chemical Design. *arXiv.org* **2018**, No. arXiv:1802.05283.

(37) Kingma, D. P.; Mohamed, S.; Rezende, D. J.; Welling, M. Semi-Supervised Learning with Deep Generative Models. *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*; 2014; pp 3581−3589.

(38) Schuster, M.; Paliwal, K. K. Bidirectional Recurrent Neural Networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673−2681.

(39) Sutskever, I.; Vinyals, O.; Le, Q. V. Sequence to Sequence Learning with Neural Networks. *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*; 2014; pp 3104−3112.

(40) Graves, A. Sequence Transduction with Recurrent Neural Networks. *Proceedings of the 29th International Conference on Machine Learning*; 2012.

(41) Boulanger-Lewandowski, N.; Bengio, Y.; Vincent, P. Audio Chord Recognition with Recurrent Neural Networks. *Proceedings of the 14th International Society for Music Information Retrieval Conference*; 2013; pp 335−340.

(42) Landrum, G. RDKit: Open-Source Cheminformatics. http://www.rdkit.org (accessed June 13, 2018).

(43) Wildman, S. A.; Crippen, G. M. Prediction of Physicochemical Parameters by Atomic Contributions. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 868−873.

(44) Bickerton, G. R.; Paolini, G. V.; Besnard, J.; Muresan, S.; Hopkins, A. L. Quantifying the Chemical Beauty of Drugs. *Nat. Chem.* **2012**, *4*, 90−98.

(45) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations*; 2015.

(46) Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*; 2014; pp 1724−1734.

(47) Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929−1958.

(48) Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*; 2016; pp 265−283.