SPEAR: Self-Supervised Post-Training Enhancer for Molecule Optimization

Tianfan Fu
tfu42@gatech.edu
Georgia Institute of Technology
Atlanta, USA

Cao Xiao
danicaxiao@gmail.com
Amplitude
San Francisco, USA

Kexin Huang
kexinhuang@hsph.harvard.edu
Harvard University
Cambridge, USA

Lucas M. Glass
Lucas.Glass@iqvia.com
Analytics Center of Excellence, IQVIA
Cambridge, USA

Jimeng Sun
jimeng.sun@gmail.com
University of Illinois,
Urbana-Champaign
Champaign, USA

ABSTRACT

The molecular optimization task is to generate molecules that are similar to a target molecule but with better chemical properties. Deep Generative Models (DGMs) have shown initial success in automatic molecule optimization. However, the training of DGMs often suffers from limited labeled molecule pairs due to the ad-hoc and restricted molecule pair construction. To solve this challenge and leverage the entire unpaired molecule database, we propose Self-Supervised Post-training EnhAncer method (SPEAR) to enhance any graph-based DGMs for molecule optimization. SPEAR mines molecular structure knowledge and learns the molecule generation procedure in a purely self-supervised fashion. Unlike most self-supervised deep learning models that rely on pre-training for better molecule representation, the SPEAR method is applied as post-processing step to enhance molecule optimization during inference time for DGMs without additional training. Our SPEAR model can be efficiently incorporated into any DGM model as part of the inference procedure. We evaluated SPEAR against several state-of-the-art DGMs, SPEAR successfully improved the performance of all DGMs and obtained 5–21% relative improvement over its corresponding DGM models in terms of success rate.

CCS CONCEPTS

• Computing methodologies Neural networks.

KEYWORDS

Molecule Optimization; Drug Discovery; Healthcare

1 INTRODUCTION

Designing molecules with desirable chemical properties is a fundamental task in drug discovery. Among others, the *molecule optimization* task takes a more targeted approach: given an input

molecule X , the goal is to find molecule Y with improved drug properties such as drug likeness, solubility, biological activity and synthetic accessibility while maintaining their similarity to X . The task is also named as “matched molecular pair analysis” (MMPA) in cheminformatics community [9, 15].

Over the years, deep generative models (DGM) have been developed and demonstrated initial success on molecule optimization [6, 14, 20, 21, 23, 25, 27]. However, to train those DGMs, we need to have a large number of paired molecules (X, Y), where X is treated as the original or input molecule and Y is the optimized molecule that is similar to the X but with better properties. However, it is costly and time-consuming to find sufficient molecule pairs as training data. The construction process often leads to a limited number of paired molecules (X, Y) due to various constraints. For example, for some realistic oracle such as docking [19], the number of molecule pairs are limited while most of the unpaired molecules can not be used during learning. The under-utilization of unpaired molecules also creates an obstacle as it affects the generalizability of DGMs on molecule optimization.

In this paper we study the following question: *how can we leverage the vast number of unlabeled (i.e., unpaired) molecule data to improve molecule optimization task?*

One strategy for leveraging unlabeled data is self-supervised model pre-training. It is widely adopted in various application areas, including computer vision [16, 24], natural language processing [8, 28]. The general idea of self-supervised model pretraining is to learn good representations from unlabeled structured data, hopefully it can help reduce or even eliminate the requirement for costly manual labelling labors. In molecule application, self-supervised model pre-training has also been used to learn molecular structured knowledge from massive unlabeled molecules. Specifically, via predicting masked atoms, existing works such as [17, 18, 33] proposed to pretrain molecule encoder to get better molecule representation and then finetune the model on downstream tasks, e.g., molecular property prediction. These methods are restricted to improve representation learning and hard to apply to generative models.

In contrast, we propose Self-Supervised Post-training EnhAncer (SPEAR) which is a molecule generator that mimics the molecule

This work is licensed under a Creative Commons Attribution International 4.0 License.
BCB '21, August 1–4, 2021, Gainesville, FL, USA
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8450-6/21/08.
<https://doi.org/10.1145/3459930.3469530>

generation procedure. SPEAR can be combined with any graph-based deep generative model in the inference procedure as a **post-hoc procedure**. We summarize the main technical contributions of the paper as follows.

- SPEAR is a self-supervised approach that can learn from all molecules, both paired and unpaired. Specifically, it imitates the scaffolding tree based molecule generation procedure to predict the expansion and substructure¹ component of the molecule itself. With such a self-supervised approach, SPEAR can extract useful knowledge about molecule generation from massive unlabelled molecules, and transfer this knowledge to a task-specific model through a simple and rapid interpolation step to elevate performance on any DGMs.

- The training of SPEAR is separate from training of DGMs and we only need to train SPEAR once for it to be used in any other task-specific model. The integration happens only in the inference level, which makes SPEAR a model-agnostic and ready-for-use enhancer for any of the molecule optimization DGMs (e.g., JTVAE, VJTNN, CORE) in various tasks (optimizing DRD2, QED and LogP).

- We experimentally evaluated SPEAR by incorporating it into three state-of-the-art models: JTVAE [20], VJTNN [23] and CORE [13]. We compare the original models with the SPEAR-augmented ones on three real-world molecule optimization tasks. Results show that SPEAR-enhanced models can achieve from 14% to 21% relative improvement over their base DGMs on similarity score and success rate, respectively. And we find that SPEAR works especially well for low-resource settings where only a small number of molecule pairs is available for model training.

2 RELATED WORKS ON MOLECULE OPTIMIZATION

Existing molecule optimization methods usually encode the raw molecule data using either its sequence representations or graph representations. The main sequence representation approaches include the simplified molecular-input line-entry system (SMILES)². While for the molecular graph representations, we represent molecules as an undirected graph by encoding each atom as nodes and bonds as edges that connect atoms. Accordingly, both sequence based and graph based neural architecture are developed to represent or generate the molecules based on these two representations, respectively. For sequence based approaches, many initial works formulate molecule generation or optimization as a sequence generation or translation problem [6, 7, 14, 25, 29]. However, they often have complex syntactic constraints. Small changes to the sequence may completely violate chemical validity [6, 14, 25].

To bypass the challenges of generating valid sequences, graph based approaches [23, 34] become more preferred strategies. Among others, reinforcement Learning (RL) has been adopted to generate molecular graphs with desirable properties [34, 35]. Their learning objective is based on molecule level reward, e.g., directly introducing chemical property as reward. However, RL based methods suffer from poor exploration efficiency [3] and are usually hard to train [23]. On the other hand, Deep Generative Models (DGMs)

exhibits state-of-the-art performance. Existing DGM works include [11, 13, 20, 23] that eliminates cycles in a molecular graph by representing the graph as a scaffolding tree where nodes are substructures such as rings and atoms. [22] is a recent enhancement that introduces hierarchical structure for the generation of relatively large chemical compounds. One potential problem of these methods is that they require a large amount of molecule pairs to train, and the molecule pairs is limited in number and expensive to obtain. In this paper, we propose SPEAR that train on massive individual molecules (in addition to the paired ones) to bypass this problem. It learns molecular structured knowledge from massive data to provide more guidance in molecule generation process and enhance the performance of molecule optimization.

3 PRELIMINARIES

In this section, we will first define some necessary concepts and formulate the molecule optimization problem. We list the important notations in Table 1.

Definition 1 (Molecular Graph). A molecule is represented by a 2D graph $\mathcal{G}$, where nodes are atoms and edges are bonds.

Definition 2 (Substructure). Substructure is the basic unit (i.e., a node) in the scaffolding tree. They can be any possible rings, bonds, and atoms. The pool of all the substructure (i.e., vocabulary) is denoted $\mathcal{S}$.

Definition 3 (Scaffolding Tree). Given any molecule graph $\mathcal{G}$, we can convert it to a scaffolding tree $\mathcal{T}_{\mathcal{G}}$, which is the skeleton of the molecular graph $\mathcal{G}$ by partitioning the original graph into substructures from the substructure pool $\mathcal{S}$ and connect the substructure based on their connections in molecular graph.

Molecular chemical property. A typical molecular property of a chemical compound includes better binding affinity to dopamine type 2 receptor target (measured by DRD2) [4], better drug-likeness (measured by QED) [2] and better lipophilicity, solubility and synthetic accessibility (measured by LogP score).

Molecular Similarity. Molecular similarity between two molecules X and Y is measured by Tanimoto similarity over Morgan fingerprints [31]. Morgan fingerprints is a binary vector where each bit records the presence (“1”) or absence (“0”) of a binary fragment descriptor in the molecule. For example, a fragment descriptor can be “if Benzene ring exist in the molecule”. It is assumed that two fingerprints with many bits in common represent similar molecules. Let $\mathcal{S}_X$ and $\mathcal{S}_Y$ denote the set of fragment descriptors that exist in molecule X and Y , respectively (i.e., presence of fragment descriptor). The definition of Tanimoto similarity is given in Equation 1,

$$\text{sim}(X, Y) = \frac{\sum_{j=1}^m x_j y_j}{\sum_{j=1}^m (x_j^2 + y_j^2 - x_j y_j)}, \quad (1)$$

where $x_j, y_j \in \{0, 1\}$ for $j = 1, 2, \dots, m$. $x_j = 1$ means the j -th feature occurs in molecule X while 0 means not. Similarly, $y_j = 1$ means the j -th feature occurs in molecule Y while 0 means not. The value ranges from 0 to 1. Higher means more similar.

Problem 1 (Molecule Optimization). Given an input molecule X , a model learns to generate new molecule Y that is similar to X

¹See Definition 2.

²SMILES is a line notation for encoding molecular structures using short ASCII strings [1]

with more desirable chemical properties. It is formulated in Equation 2

$$\arg \max_Y \text{Property}(Y), \quad \text{such that } \text{sim}(X, Y) \geq \eta_1. \quad (2)$$

where $\text{Property}(X)$ can be any chemical/biological property of the molecule X , $\eta_1 > 0$ is hyperparameter.

Table 1: Mathematical Notations used in this paper.

Notations	Description
$X, Y, \tilde{Y}$	input molecule, target molecule, generated molecule
$\mathcal{S}$	Substructure pool.
$\text{sim}(X, Y) \in [0, 1]$	Tanimoto similarity between X and Y
V/E	vertex(atom) set, edge(bond) set.
$\mathcal{G} = (V, E)$	molecular graph of input molecule X
$\mathcal{G}_0$	molecular graph of target molecule Y
$\mathcal{T}_{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$	scaffolding tree of graph $\mathcal{G}$.
$\mathbb{T}_{\mathcal{G}}$	scaffolding tree of $\mathcal{G}$ without substructure, output of expansion prediction
$\text{Neib}(v)$	set of neighbor nodes of vertex v
$\sigma/\text{ReLU}/\text{Softmax}$	sigmoid/ReLU/Softmax activation function
D	Depth of the message passing network
$\mathbf{w}_i^{\mathcal{G}}/\mathbf{w}_i^{\mathcal{T}}$	embedding of node i in $\mathcal{G}/\mathcal{T}_{\mathcal{G}}$
$\mathbf{U}_1, \dots, \mathbf{U}_5$	parameters in message passing network, 3, 4.
$\mathbf{a}_v/\mathbf{a}_{uv}$	feature vector for node v / edge (u, v)
$i_t \in \mathcal{S}$	substructure of t -th visit in DFS
$b_t \in \{0, 1\}$	expansion variable for t -th visit in DFS
$\mathbf{h}_{i_t, j_t}$	message vector for edge (i_t, j_t) , Eq. 6
p_t^{expand}	expansion prediction score at t -th step
$\mathbf{q}_t^{\text{sub}}$	substructure prediction distribution at t -th step
$\mathbf{z}_{\mathcal{G}}$	Embedding of Graph $\mathcal{G}$
$h_1(\cdot), h_2(\cdot)$	feedforward neural networks in DGM
$g_1(\cdot), g_2(\cdot)$	feedforward neural network in SPEAR
T	length of expansion/substructure sequence
$\{b_1, \dots, b_T\}$	Expansion sequence, Definition 4.
$\{i_1, \dots, i_T\}$	Substructure sequence, Definition 5.
λ	hyperparameter in learning SPEAR, Eqn 17.
γ_1, γ_2	hyperparameter in combining DGM and SPEAR, Eqn 18, 19.

4 METHOD

As shown in Fig. 1, task-dependent DGMs and task-independent SPEAR are trained separately and combined during inference. SPEAR has a carefully designed self-supervised task to extract general knowledge about molecule generation from massive unlabelled molecule data. Then for any scaffolding-tree based DGMs that is trained on paired molecules, SPEAR has a model-agnostic post-training step to rapidly integrate the task-independent knowledge to the task-dependent DGMs and elevate the performance.

4.1 Deep Generative Models for Molecule Optimization

To train a deep generative model (DGM) for molecule optimization, we follow the idea leveraged by a few recent works [13, 20, 23], and decompose the DGM learning into the following steps.

4.1.1 Representation Learning. To prepare input data for DGMs, we first learn embedding or representation for both scaffolding tree and molecular graph. In particular, we embed each node in both the input molecular graph $\mathcal{G}$ and its scaffolding tree $\mathcal{T}_{\mathcal{G}}$ using graph Message Passing Networks (MPN) [5, 20] as below.

First, for each node v , we assume v has a feature vector $\mathbf{a}_v$. For example, in a molecular graph G , we can consider each atom as a node v , then the node feature $\mathbf{a}_v$ will include the atom type, valence, and other atomic properties. Another example is in the corresponding scaffolding tree $\mathcal{T}_{\mathcal{G}}$, node v often refers to a substructure, in which case $\mathbf{a}_v$ will be a one-hot vector indicating the category of the substructure.

Then for each edge $(u, v) \in E$ that connects node u and node v , there is an edge feature vector $\mathbf{a}_{uv}$, which can be iteratively updated as given by Eq. 3,

$$\mathbf{v}_{uv}^{(d)} = \text{ReLU}\left(\mathbf{U}_1 \mathbf{a}_u + \mathbf{U}_2 \mathbf{a}_{uv} + \mathbf{U}_3 \left(\sum_{w \in \text{Neib}(u) \setminus v} \mathbf{v}_{wu}^{(d-1)}\right)\right), \quad (3)$$

where $d = 1, \dots, D - 1$,

where we use ReLU as the activate function. D is the depth of the message passing network. $\text{Neib}(v)$ is the set of all the neighbor nodes of node v . $\mathbf{v}_{uv}$ and $\mathbf{v}_{vu}$ are the hidden variables that represent the message from node u to node v and vice versa. Also $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$ are learnable weight parameters, shared by different layers in message passing network, while $\mathbf{v}_{uv}^{(d)}$ is the message vector from node u to node v at the d -th iteration, whose initialization is all-0 vector, i.e., $\mathbf{v}_{uv}^{(0)} = \mathbf{0}$. After D steps of iteration, another fully-connected neural network with ReLU is used to aggregate these messages. Each vertex has a latent vector as in Eq. 4,

$$\mathbf{w}_u = \text{ReLU}\left(\mathbf{U}_4 \mathbf{a}_u + \mathbf{U}_5 \left(\sum_{v \in \text{Neib}(u)} \mathbf{v}_{vu}^{(D)}\right)\right), \quad (4)$$

where $\mathbf{U}_4$ and $\mathbf{U}_5$ are learnable weight parameters.

In summary, the encoder module yields embedding vectors for nodes in graph G and scaffolding tree $\mathcal{T}_{\mathcal{G}}$, denoted $\mathbf{W}^G = \{\mathbf{w}_1^G, \mathbf{w}_2^G, \dots\}$ and $\mathbf{W}^{\mathcal{T}_{\mathcal{G}}} = \{\mathbf{w}_1^{\mathcal{T}_{\mathcal{G}}}, \mathbf{w}_2^{\mathcal{T}_{\mathcal{G}}}, \dots\}$ as in Eq. 5,

$$\begin{aligned} \mathbf{W}^{\mathcal{G}} &= \text{MPN}(\mathcal{G}), \quad \text{where } \mathbf{W}^{\mathcal{G}} = \{\mathbf{w}_1^{\mathcal{G}}, \mathbf{w}_2^{\mathcal{G}}, \dots\}, \\ \mathbf{W}^{\mathcal{T}_{\mathcal{G}}} &= \text{MPN}(\mathcal{T}_{\mathcal{G}}), \quad \text{where } \mathbf{W}^{\mathcal{T}_{\mathcal{G}}} = \{\mathbf{w}_1^{\mathcal{T}_{\mathcal{G}}}, \mathbf{w}_2^{\mathcal{T}_{\mathcal{G}}}, \dots\}, \end{aligned} \quad (5)$$

where $\mathbf{w}_i^{\mathcal{G}}$ and $\mathbf{w}_j^{\mathcal{T}_{\mathcal{G}}}$ represent embedding of the i -th and j -th node in molecular graph $\mathcal{G}$ and scaffolding tree $\mathcal{T}_{\mathcal{G}}$, respectively. Average function is leveraged as readout function to produce graph-level representation.

4.1.2 Generator. The goal of generator is to generate new molecules that have optimized properties. Based on the representation of input molecule X in Eq. 5, the generator first construct a scaffolding tree and then recovers back to the molecular graph using the tree. The tree-to-graph transformation is straightforwardly enumerating all possible candidates and choosing the most likely one. The generated graph is thus decided largely by the generated scaffolding tree. SPEAR focuses on generating a better scaffolding tree as described below.

The generator constructs scaffolding tree from an empty root node in depth-first search (DFS) manner. In each single visit of DFS, the model performs two tasks. The first is a binary classification that predicts “expand” the current node or “backtrack” to its parent node (*expansion prediction*) and the second is to choose what to add from

Figure 1: SPEAR Framework. (I). **Learning** (Section 4.3). SPEAR leverages large number of molecules: transforms them into scaffolding tree, traverses the tree via Depth-First Search (DFS) and generates traverse sequences. Training of SPEAR is formulated as a sequence labelling task by predicting Y_t (Eq. 13) based on $\tilde{Y}_{1:t-1}$ (Eq. 12). (II). **Inference** (Section 4.4). SPEAR is combined with well-trained DGMs during inference. The key idea is to use SPEAR to modify all the prediction scores of a DGM model (in both expansion prediction and substructure prediction, as shown in Eq. 18 and 19).

substructure pool S (substructure prediction). When it backtracks to the root node and decided not to expand, the generation procedure is terminated automatically.

Expansion prediction. When the generator visits the node i_t , the model makes a binary prediction on either “expanding a new node” or “backtracking to the parent node of i_t ”. This prediction is called expansion prediction. To do that, scaffolding tree generator first generates the embedding for node i_t via a tree-based RNN [20], then use these embeddings of RNN to predict whether to expand or backtrack. Since RNN works on a sequence, the tree converts into a sequence of nodes and edges via depth-first search (DFS). Specifically, let $\tilde{\mathcal{E}} = \{(i_1, j_1), (i_2, j_2), \dots, (i_m, j_m)\}$ be the edges traversed in depth first search, each edge is visited twice in both directions, so we have $m = |\tilde{\mathcal{E}}| = 2|\mathcal{E}|$. Suppose during the first t steps, we have already generated the scaffolding tree $\tilde{Y}_{1:t-1}$, $\tilde{\mathcal{E}}_t$, the first t edges in $\tilde{\mathcal{E}}$, message vector $\mathbf{h}_{i_t, j_t}$ is updated as:

$$\mathbf{h}_{i_t, j_t} = \text{GRU}(\mathbf{a}_{i_t}, \{\mathbf{h}_{k, i_t}\}_{(k, i_t) \in \tilde{\mathcal{E}}_t, k \neq j_t}). \quad (6)$$

where $\mathbf{a}_{i_t}$ is the feature vector for node i_t while $\{\mathbf{h}_{k, i_t}\}_{(k, i_t) \in \tilde{\mathcal{E}}_t, k \neq j_t}$ is the hidden state.

At the t -th step, the expansion probability p_t^{expand} , i.e., the probability whether to expand or backtrack at node i_t , expressed as a function of the embeddings $\mathbf{W}^{\mathcal{T}_{\mathcal{G}}}$, $\mathbf{W}^{\mathcal{G}}$ and the current state $\mathbf{a}_{i_t}$, $\sum_{(k, i_t) \in \tilde{\mathcal{E}}_t} \mathbf{h}_{k, i_t}$ using a two-layer feed-forward neural network $h_1(\cdot)$ with ReLU function in hidden layer:

$$p_t^{\text{expand}} = \sigma \left(h_1(\mathbf{a}_{i_t}, \sum_{(k, i_t) \in \tilde{\mathcal{E}}_t} \mathbf{h}_{k, i_t}, \mathbf{W}^{\mathcal{T}_{\mathcal{G}}}, \mathbf{W}^{\mathcal{G}}) \right), \quad (7)$$

for $t = 1, \dots, m$.

In sum, the expansion probability at t -th step is modelled as a function of input molecule X and the generated scaffolding tree $\tilde{Y}_{1:t-1}$ at first $(t-1)$ -th steps,

$$p_t^{\text{expand}} \triangleq p_{\text{DGM}}^{\text{expand}}(b_t | X, \tilde{Y}_{1:t-1}), \quad (8)$$

where the binary label $b_t \in \{0, 1\}$, representing either expanding ("1") or backtracking ("0") at the t -th visit in DFS. Binary cross entropy loss is used for model training.

Substructure prediction. If the expansion prediction decide to expand, we need to select a substructure from the substructure pool $\mathcal{S}$ to add. Similar to the expansion prediction above, it leverages tree-based RNN to generate the embedding for node i_t and add a two-layer feedforward neural network $h_2(\cdot)$ to predict the substructure’s category,

$$\mathbf{q}_t^{\text{sub}} = \text{Softmax}\left(h_2(\mathbf{a}_{i_t}, \sum_{(k, i_t) \in \tilde{\mathcal{E}}_t} \mathbf{h}_{k, i_t}, \mathbf{W}^{\mathcal{T}_{\mathcal{G}}}, \mathbf{W}^{\mathcal{G}})\right), \quad (9)$$

All in all, substructure probability at the t -th step is a function of X and $\tilde{Y}_{1:t-1}$,

$$\mathbf{q}_t^{\text{sub}} \triangleq \mathbf{q}_{\text{DGM}}^{\text{sub}}(i_t | X; \tilde{Y}_{1:t-1}), \quad (10)$$

where the substructure label $i_t \in \{1, \dots, |\mathcal{S}|\}$ represents the substructure at the t -th visit in DFS. Multi-class cross entropy loss is used for modeling training.

4.2 Overview of Self-Supervised Post-Training Enhancer (SPEAR)

To tackle the challenge where the number of molecule pairs is not sufficient to train a DGM, with a little abuse of mathematic notation, SPEAR trains $P(Y)$ on massive unlabeled molecule data and interpolates $P(Y)$ with DGM $P(Y|X)$ that is trained on paired molecules. Specifically, during learning procedure (Section 4.3), SPEAR imitates the sequential molecule generation procedure. Given an incomplete scaffolding tree that is already generated, the objective of SPEAR is to predict what to generate next, via (1) expansion prediction that decides whether to expand the tree, and (2) substructure prediction that determines which substructure to select. Then in inference (Section 4.4) we will combine SPEAR with DGM to generate new molecules.

4.3 The Learning Procedure of SPEAR

We reformulate the scaffolding tree based molecule generation as a sequence labelling task. As shown in Fig. 2, the learning comprises of two parts: (I) sequence generation, and (II) sequence labelling.

4.3.1 Sequence Generation. As discussed before, a scaffolding tree is traversed via DFS and the generation procedure can be divided into expansion and substructure prediction. First, based on the order of DFS, we transform tree structure into expansion sequence and substructure sequence as defined below.

Definition 4 (Expansion Sequence). An expansion sequence is denoted as $\{b_1, \dots, b_t, \dots, b_T\}$ where $b_t = 1$ means expanding a new node and $b_t = 0$ means backtracking to the previous node at the t -th visit in DFS. T is the number of search step required in the depth first search to traverse the molecule’s scaffolding tree, also the length of expansion sequence.

Definition 5 (Substructure Sequence). An substructure sequence is the sequence of nodes in the DFS traversal, denoted as

$$\{i_1, \dots, i_t, \dots, i_T\}$$

Figure 2: (I) Sequence generation (Section 4.3.1) is based on the order of Depth First Search (DFS), from 1 to 6. (II) Sequence labelling (Section 4.3.2). Then we leverage multi-layer LSTM to predict the operations for the next step. Particularly we predict whether to expand (1) or backtrack (0), as well as which substructure to add.

where $i_t \in \{1, 2, \dots, |\mathcal{S}|\}$ is the index of the substructure at the t -th node in DFS traversal, $\mathcal{S}$ is the pool of all the substructures (Definition 2).

Now we get the sequence $\{(i_t, b_t)\}_{t=1}^T$. Fig. 2 illustrates how to transform molecule (scaffolding tree) generation into a sequence prediction based on the order of DFS traversal.

4.3.2 Sequence Labelling. For every $t = 2, 3, \dots, T$, we want to predict both i_t and b_t based on all the previous sequence $\{(i_{t'}, b_{t'})\}_{t'=1}^{t-1}$, which can be formulated as below:

$$\mathbf{r}_t, \mathbf{h}_t = \text{LSTM}([\mathbf{e}_{i_t}, b_t]; \mathbf{h}_{t-1}), \quad (11)$$

where $t = 1, \dots, T - 1$.

where $\mathbf{r}_t$ and $\mathbf{h}_t$ represent the output and hidden state of current LSTM, respectively. In this paper, we use three-layer LSTM (Long short-term memory) model for SPEAR. $\mathbf{e}_{i_t}$ represents the embedding vector of the substructure i_t . $\mathbf{e}_{i_t}$ and b_t are concatenated as the feature at the t -th step and as the input of LSTM. All the previous information in the first $(t - 1)$ steps is

$$\tilde{Y}_{1:t-1} = \{(i_{t'}, b_{t'})\}_{t'=1}^{t-1}. \quad (12)$$

$\tilde{Y}_{1:t-1}$ is encoded in the hidden state $\mathbf{r}_{t-1}$.

To make the binary prediction whether the sequence would expand ("1" means expand, "0" means not), a two-layers feedforward network $g_1(\cdot)$ is stacked on top of LSTM’s output $\mathbf{r}_{t-1}$ to predict b_t (Definition 4)

$$p_{\text{SPEAR}}^{\text{expand}}(b_t | \tilde{Y}_{1:t-1}) = \sigma(g_1(\mathbf{r}_{t-1})), \quad (13)$$

where $\sigma(z) = \frac{1}{1 + \exp(-z)}$ is the sigmoid activation function. ReLU is leveraged as activation function in hidden layer of $g_1(\cdot)$. Binary cross entropy criterion is leveraged to evaluate the binary classification loss and guide the training. Specifically, for each molecule Y

whose expansion sequence has T binary labels, the loss becomes

$$\mathcal{L}_{\text{expand}} = \sum_{t=1}^T b_t \left(\log p_{\text{SPEAR}}^{\text{expand}}(b_t | \tilde{Y}_{1:t-1}) \right) + (1 - b_t) \log \left(1 - (p_{\text{SPEAR}}^{\text{expand}}(b_t | \tilde{Y}_{1:t-1})) \right). \quad (14)$$

We use the similar methods to predict the substructure sequence, which is a multi-class classification task. Concretely, the prediction is a probability over all the possible substructures, as

$$\mathbf{q}_{\text{SPEAR}}^{\text{sub}}(i_t | \tilde{Y}_{1:t-1}) = \text{Softmax}(g_2(\mathbf{r}_{t-1})). \quad (15)$$

where $g_2(\cdot)$ are two-layer fully-connected networks, softmax function is used as activation function in output layer. ReLU function is used as activation function in the hidden layer of $g_2(\cdot)$.

Categorical cross-entropy criterion is employed to evaluate the classification loss and guide the training of SPEAR. Specifically, the learning objective of SPEAR is

$$\mathcal{L}_{\text{sub}} = \sum_{t=1}^T b_t \left(\sum_{l=1}^{|S|} \mathbb{1}(i_t = l) \log \mathbf{q}_l^{(t)} \right), \quad (16)$$

where $\mathbb{1}(\cdot)$ is the indicator function, $\mathbf{q}^{(t)}$ is the abbreviation of $\mathbf{q}_{\text{SPEAR}}^{\text{sub}}(i_t | \tilde{Y}_{1:t-1})$ defined in Equation 15. $\mathbf{q}_l^{(t)}$ is its l -th element. $b_t \in \{0, 1\}$ is the expansion variable described in Definition 4. It is used as mask variable here, because only when the scaffolding tree decided to expand at the t -th step, i.e., $b_t = 1$, substructure prediction is needed. Otherwise, DFS backtracks and go back to the previous nodes, in this case, we do not need to predict the substructure. To summarize, the objective function in learning is a linear combination of $\mathcal{L}_{\text{expand}}$ and $\mathcal{L}_{\text{sub}}$, as in Eq. 17,

$$\mathcal{L} = \mathcal{L}_{\text{expand}} + \lambda \mathcal{L}_{\text{sub}}. \quad (17)$$

where $\lambda > 0$ is a hyperparameter that balances the weight of two prediction tasks. As illustrated by Figure 2, for the prediction of expansion sequence and substructure sequence, the input feature and LSTM architecture are shared. Two separate neural network $g_1(\cdot)$ and $g_2(\cdot)$ are stacked on the output of LSTM to predict the two different tasks. Worth to mention that all the valid molecules can be used as the training data for SPEAR since there are no molecule pairs required.

4.4 The Inference Procedure of SPEAR

During inference, we will integrate the trained SPEAR with a trained DGM model to enhance the performance of the DGM. Specifically, we have already generated the incomplete molecule in the first $t - 1$ steps, denoted $\tilde{Y}_{1:t-1}$ (Eq. 12), and need to compute the probability of whether to expand at the t -th step (denoted $p^{\text{expand}}(b_t)$). Then DGMs computes the probability directly based on both X and $\tilde{Y}_{1:t-1}$. However, owing to scarcity of molecule pairs, they often do not fit well to the general molecule distribution, which exhibits "low bias + high variance" behaviour.

On the other hand, the trained SPEAR model can also be used to compute the probability based on $\tilde{Y}_{1:t-1}$ alone (without X). But many more molecules have been used in training SPEAR model, which exhibit high bias and low variance behavior. Thus, we want to balance both probability estimates. Concretely, the predicted

probability is an interpolation between the predicted probability of the SPEAR model and the DGM, defined as given by Eq. 18,

$$p_t^{\text{expand}} = \gamma_1 \cdot p_{\text{SPEAR}}^{\text{expand}}(b_t | \tilde{Y}_{1:t-1}) + (1 - \gamma_1) \cdot p_{\text{DGM}}^{\text{expand}}(b_t | X, \tilde{Y}_{1:t-1}), \quad (18)$$

where γ_1 is a positive hyperparameter between 0 and 1, balancing the weight of the predictions from SPEAR and DGM. It is tuned on the validation set. p_t^{expand} ranges from 0 to 1 and is still a probability. The first term is the expansion prediction from SPEAR while the second is from DGM in Eq. 8. $\tilde{Y}_{1:t-1}$ represent the previously generated molecule in the first $(t - 1)$ steps, defined in Eq. 12.

Similarly, during substructure prediction, the prediction over the whole substructure space can be seen as an interpolation between the prediction of DGM and SPEAR as in Eq. 19,

$$\mathbf{q}_t^{\text{sub}} = \gamma_2 \cdot \mathbf{q}_{\text{SPEAR}}^{\text{sub}}(i_t | \tilde{Y}_{1:t-1}) + (1 - \gamma_2) \cdot \mathbf{q}_{\text{DGM}}^{\text{sub}}(i_t | X; \tilde{Y}_{1:t-1}), \quad (19)$$

where γ_2 is also a positive hyperparameter between 0 and 1, tuned on validation set. Since both $\mathbf{q}_{\text{SPEAR}}^{\text{sub}}(i_t | \tilde{Y}_{1:t-1})$ and $\mathbf{q}_{\text{DGM}}^{\text{sub}}(i_t | X; \tilde{Y}_{1:t-1})$ are probability distribution over the substructure pool $\mathcal{S}$, $\mathbf{q}_t^{\text{sub}}$ is also a distribution on $\mathcal{S}$.

Bias-Variance Tradeoff. Training DGMs require a great number of parallel molecules. However, parallel data are hard to collect and are limited in number. As shown in Table 2, for three common chemical properties in molecule optimization, thousands of molecule pairs are collected. Though it is "unbiased" for molecule optimization, the lack of data leads to a large variance in the learned DGMs. In contrast, SPEAR do not need paired molecules and any druglike (or bioactive) molecules can be used to learn SPEAR. In this paper, we use ZINC (<https://zinc.docking.org/>), which contains around 250K druglike molecules. Thus, although SPEAR is "biased" for molecule optimization, it is learned better and has smaller variance. We balance these two strategies via integrating DGMs and SPEAR.

Selection of Hyperparameter. Note that $\gamma_1, \gamma_2 \in (0, 1)$ are hyperparameters that balance the weight of SPEAR and DGMs and play a crucial role in inference. We tune them on validation set and select the hyperparameters' combination that get the best performance measured by success rate as described in Section 5.1.

5 EXPERIMENT

5.1 Experimental Setup

Molecular Property Datasets. Our experiments mainly concern improving three molecular properties, including QED [2], DRD2 [4] and LogP [10]. A compound's QED score is quantitative estimate of drug likeness, which an integrative score to evaluate compounds' favorability to become a hit. DRD2 score measures biological activity against dopamine type 2 receptor, the predicted probability that owns biological activity is generated by a well-trained support vector machine [19]. LogP score is Octanol-water Partition Coefficient, measures the lipophilicity, solubility and synthetic accessibility. QED-score and LogP-score can be evaluated by Rdkit package [26]. DRD2-score is evaluated by a well-trained support vector machine (SVM) model with a Gaussian kernel with ECFP6 fingerprint on ExCAPE-DB dataset [30], following [23, 30]. QED

and DRD2 score range from 0 to 1 while LogP score ranges from $-\infty$ to $+\infty$. In experiments, we used the following datasets.

- **DGM learning.** Molecule pairs are extracted from the ZINC dataset [32]³. In addition, we also use the following publicly available datasets including QED, DRD2, and LogP. The three datasets each label one target chemical property. Data statistics are provided in Table 2.

- **SPEAR learning.** we remove all the molecules that occurs in either validation set or test set in these three above tasks, the remaining 235K molecules are used for learning SPEAR. Among them, we hold out 25K molecules as validation set.

Paired Dataset Construction The paired dataset used to train DGM is also extracted from the ZINC dataset. A molecular pair (X, Y) is selected into the training set if their Tanimoto similarity is greater than a prespecified threshold (i.e., $\text{sim}(X, Y) \geq \eta_1$) and the property improvement is also greater than certain threshold, as defined in Equation 2. Here we follow [23] to set $\eta_1 = 0.4$ for all three datasets. The thresholds for property improvement are listed below: For QED, the QED score of X is in [0.7, 0.8] while the score of Y is in [0.9, 1]. For DRD2, X’s property is lower than 0.05 and Y’s property is greater than 0.5. For LogP, we adopt the property constraint according to [23, 32]. Also, we guaranteed that all molecules in validation and test set are not in training set.

Base Methods We select several state-of-the-art molecule optimization approaches as our base models.

- **JTVAE** (Junction Tree Variational Auto-Encoder) [20]. It generates molecules via a tree-structured scaffold over chemical substructures and assembles the scaffold into a molecule.

- **VJTNN** (Variational Junction Tree Encoder-Decoder, also called graph-to-graph translation) [23]. It includes a junction tree encoder-decoder for learning diverse graph translations along with adversarial training.

- **CORE** (COpy&REfine) [13]. To further enhance VJTNN and simplify substructure generation procedure, CORE selects (copy) some substructures from input molecule based on attention mechanism.

We also incorporate **GCPN** (Graph Convolutional Policy Network) [34] as baseline for sake of completeness. GCPN is state-of-the-art reinforcement learning based approach, which also use molecular graph representation, incorporates chemical property into reward function and then optimizes the reward function through policy gradient.

Implementation Details. We follow most of the hyperparameter setting of JTVAE [20], VJTNN [23] and CORE [13] in our experiments. For all base models, the max epoch number is set to 10, batch size is set to 32. During encoder module, we set embedding size as 300, the depth of graph message passing network as 6, and the depth for trees and graphs as 3. In SPEAR, the layer number of LSTM is 3, input size is 200, hidden size is 500. The initial learning rate is set to $1e^{-3}$ with the Adam optimizer. Every epoch the learning rate is annealed by 0.8. Convergence criteria is checked every epoch on validation set. When the success rate of the current epoch is lower than that of previous one, we stop the learning procedure.

For VJTNN, during adversarial training, discriminator $D(\cdot)$ is a three-layer feed-forward network with hidden layer dimension 300 and LeakyReLU activation function.

Evaluation Metrics We evaluated the performance using the following popular metrics.

- **Similarity.** Molecular similarity between the input molecule and the generated molecule is measured by Tanimoto similarity over Morgan fingerprints [31]. The molecular similarity between two molecules is defined in Equation 1. The value ranges from 0 to 1. Higher value means more similarity. Both generation of Morgan fingerprints and computation of Tanimoto similarity are standard tools in Rdkit package [26].

- **Molecular Property of Generated Molecule.** The second metric is the desired property of a generated molecule, i.e., $\text{Property}(Y)$. The molecular property of interest includes QED, DRD2, LogP, as mentioned.

- **Success Rate (SR).** Success Rate is a metrics that consider both similarity and property improvement. Following [21, 23], we design a criteria to judge whether it satisfied these two aspects: (a) Input and generated molecules are similar enough, $\text{sim}(X, Y) \geq \lambda_1$; (b) improvement are big enough, i.e., $\text{property}(Y) - \text{property}(X) \geq \lambda_2$ or $\text{property}(Y) \geq \lambda_3$. The selection of λ_1 and λ_2 (or λ_3) depend on datasets. Following [12, 13, 21, 23], for QED and DRD2 dataset, when the similarity between input and generated molecule is greater than 0.3 and chemical property (QED, DRD2) of generated molecule is greater than 0.6, we regard the target molecule as a success. For LogP dataset, when the similarity between input and generated molecule is greater than 0.4 and LogP improvement is greater than 0.8, we regard the target as a success.

- **Novelty.** A simple sanity check is to see if the generated molecule is new from the training data. A number of equally valid SMILES strings can be written for a molecule. So we need to convert the SMILES into canonical form using RDKit package [26]. If the generated molecule does not appear in training set and is different from input molecule in the test set, we regard it as a new molecule. Keep in mind that this simple metric is mainly as a sanity check to make sure we are generating new molecules. Due to the huge chemical space, almost all the methods we compared perform very well on this metric.

- **Run time & Model Size.** To measure the computational cost, training time and the model size for each model are also provided. The unit of training time is hour.

Convergence criteria for SPEAR Learning. Due to the huge number of training data, convergence criteria is checked every 0.1 epoch on validation set. When the loss on validation set would not decrease, we claims that the learning procedure converges and terminate the learning process.

Hardware and Software Configuration. All the experiments are run on an Intel Xeon E5-2690 machine with 256G RAM and 8 NVIDIA Pascal Titan X GPUs. Our method is implemented by Python 3.7 and Pytorch 1.7.1.

³<https://zinc.docking.org>

Table 2: Data statistics. All basic units are molecular substructures from the substructure vocabulary S . Here “Avg # Substructures” is the average number of substructures per molecule. For each test sample, we generate 20 molecules using different random seeds. η_1 is hyperparameter defined in Equation 2.

Dataset	Range	# Training Pairs	# Valid Pairs	# Test	$ S $, vocab size	η_1
QED (Drug likeness)	[0,1]	84,306	4,000	800	780	0.4
DRD2 (dopamine type 2 receptor)	[0,1]	32,404	2,000	1,000	967	0.4
LogP (Solubility and synthetic accessibility)	$[-\infty, +\infty]$	94,909	5,000	800	785	0.4

Table 3: Comparison of novelty of the generated molecule.

Method	QED	DRD2	LogP
GCPN [34]	92.8%	97.4%	89.7%
JTVAE [20]	95.2%	96.2%	94.2%
JTVAE+SPEAR	96.3%	95.4%	94.3%
VJTNN [23]	97.3%	96.8%	94.8%
VJTNN+SPEAR	98.3%	96.7%	95.0%
CORE [13]	94.2%	93.4%	92.8%
CORE+SPEAR	94.2%	92.7%	92.9%

5.2 Results

5.2.1 Performance Comparison. We report model performance in Table 4. From the result, we find SPEAR brings consistent and significant performance improvement to all base DGM models across all tasks and metrics. Particularly, SPEAR brings up to 21.01%, 11.84% and 7.83% performance gain on success rate for JTVAE, VJTNN, and CORE, respectively. When measured by similarity, SPEAR achieves at most 14.90%, 9.91% and 7.79% relative improvement for JTVAE, VJTNN and CORE, respectively.

Compared with significant improvement on similarity and success rate, SPEAR slightly improves molecular properties. SPEAR achieves at most 1.81%, 3.04% and 2.81% relative improvement on chemical properties for JTVAE, VJTNN and CORE, respectively. The reason is that SPEAR is property agnostic and only able to contribute to enhance molecular structured knowledge.

Among all base models, SPEAR benefits JTVAE most and CORE least. The reason is JTVAE has a simpler neural architecture while VJTNN and CORE are more expressive. Most of the SPEAR enhanced methods outperform GCPN, validating the superiority of deep generative models in molecule optimization task.

5.2.2 Exp 2. Varying Number of Training Pairs. To evaluate the impact of number of training pairs, we vary the number of training pairs and compare base DGMs with their SPEAR enhancement. We report the results measured by success rate in Fig. 3.

We observe that SPEAR achieve significant and consistent improvement over their base DGM approaches, especially when the number of molecule pair is limited. In real-world molecule optimization task, it’s prohibitively time-consuming and expensive to obtain enough molecule pairs. Thus SPEAR is expected to benefit real world molecule optimization problem.

5.2.3 Results on Molecule Novelty, Model Efficiency, and Important Hyperparameter Tuning. In addition, we also compare the novelty of

generated molecules in Table 3. We find that almost all the methods maintain high novelty (more than 90%) in all the tasks and using SPEAR does not hurt the novelty of DGMs.

Efficiency wise, The run time to use SPEAR for a new DGM on a new task is negligible since the SPEAR integration is during inference. Note that we only need to train SPEAR once and it can be reused to improve performance for any scaffold based model in any molecular optimization task. Also, the model size for SPEAR is 1.4 million parameters whereas the size of DGMs is at least 15 millions. Thus SPEAR does not add much space complexity.

We also show how to set the model’s hyperparameters. When learning SPEAR, λ defined in Equation 17 balances the weight of expansion prediction and substructure prediction tasks in the whole learning objective. We select λ from $\{0.1, 0.3, 1.0, 3.0, 10.0\}$ and find $\lambda = 1.0$ achieves best performance on validation set. Then during inference, γ_1 in Eq. 18 and γ_2 in Eq. 19 play important role in final performance. We use validation set to tune it and select them from the set $\{0.5, 0.3, 0.2, 0.1, 0.05, 0.03, 0.01\}$. The best selections of γ_1 and γ_2 depend on dataset. When optimizing QED score, $\gamma_1 = 0.2$, $\gamma_2 = 0.2$ usually achieves best performance. When optimizing DRD2 and LogP score, $\gamma_1 = 0.1$, $\gamma_2 = 0.1$. Empirically, we observe increased model performance using large γ when DGM is trained with less paired molecules. This confirms with our assumption that leveraging large unlabeled data can improve performance in low resource setting.

5.2.4 Case Study. We also showcase some case study to demonstrate the effectiveness of the proposed methods in Figure 4 on the task of optimizing QED score. To showcase the impact of SPEAR, we compare the two molecules generated by VJTNN+SPEAR and VJTNN, respectively. Although both generated molecules has higher QED score than the original input molecule and exhibit high similarity with the input molecule, VJTNN+SPEAR brings much more improvement in terms of both similarity and property gain. Specifically, both generated molecules have a benzene ring with two chlorine atoms (although in different positions. Their main difference is VJTNN can not reproduce the long single chain in the left part of the input molecule and early terminate the generation process. The key reason behind it is the inaccurate expansion prediction. In contrast, VJTNN+SPEAR is able to correct the expansion prediction so that VJTNN+SPEAR can recover more complete molecule and achieves better performance.

6 CONCLUSION

In this paper, we proposed self-supervised post-training enhancer (SPEAR) for improving deep generative models for automatic molecule optimization during the inference time. Specifically, SPEAR

Table 4: SPEAR enhancement outperforms the base DGMs: JTVAE [20], VJTNN [23] and CORE [13]. We mainly focus on the setting that the number of molecule pairs is limited. Here, we use QED with 10K pairs, DRD2 with 4K pairs and LogP with 12K pairs. The reward function of GCPN includes the target chemical property we want to optimize.

Method	Similarity			Property			Success Rate (%)		
	QED	DRD2	LogP	QED	DRD2	LogP	QED	DRD2	LogP
GCPN [34]	0.2430	0.2520	0.2709	0.714	0.563	2.540	34.50	30.73	37.34
JTVAE [20]	0.2386	0.2013	0.2432	0.663	0.534	2.253	27.43	24.32	31.32
JTVAE+SPEAR improvement	0.2584	0.2313	0.2530	0.668	0.543	2.294	30.52	29.43	33.21
	8.29%	14.90%	4.02%	0.75%	1.68%	1.81%	11.26%	21.01%	6.03%
VJTNN [23]	0.2532	0.2452	0.3053	0.734	0.624	2.462	34.35	31.81	38.42
VJTNN+SPEAR improvement	0.2783	0.2543	0.3254	0.747	0.643	2.479	38.42	35.43	42.12
	9.91%	3.71%	6.58%	1.77%	3.04%	0.69%	11.84%	11.38%	9.63%
CORE [13]	0.2643	0.2645	0.3102	0.752	0.649	2.584	37.83	34.82	39.70
CORE+SPEAR improvement	0.2849	0.2798	0.3247	0.754	0.658	2.643	39.94	36.64	43.81
	7.79%	5.78%	4.67%	0.10%	1.38%	2.81%	5.57%	5.23%	7.83%

Figure 3: Performance of various DGMs (solid line) and their SPEAR enhancement (dashed line) on various molecule optimization tasks with varying number of training molecule pairs measured by Success Rate. SPEAR achieves significant and consistent improvement over their base DGM approaches, especially when the number of molecule pair is limited.

Figure 4: Case Study of optimizing QED score.

reformulates molecule generation procedure as two sequence prediction tasks. Unlike deep generative models that requires a large amount of paired molecules, SPEAR is able to learn molecule structure using all the available molecules without pairing. Also, since SPEAR is applied during inference time, it only needs to be trained once and for all uses. Experiments are conducted on three state-of-the-art DGMs including JTVAE [20], VJTNN [23], CORE [13] and three popular molecule optimization tasks, including optimizing QED score, DRD2 score and LogP score. Thorough empirical studies have been conducted on several real-world molecule optimization tasks to validate the effectiveness of SPEAR, with significant performance improvement when the number of molecule pairs is limited.

ACM Reference Format:

Tianfan Fu, Cao Xiao, Kexin Huang, Lucas M. Glass, and Jimeng Sun. 2021. PEARL: Prototype Learning via Rule Learning. In *12th ACM International*

Conference on Bioinformatics, Computational Biology and Health Informatics (ACM-BCB’21), August 1–4, 2021. ACM, New York, NY, USA. 10 pages. <https://doi.org/10.1145/3307339.3343250>

ACKNOWLEDGEMENTS

This work was in part supported by the National Science Foundation award SCH-2014438, PPOSS 2028839, IIS-1838042, the National Institute of Health award NIH R01 1R01NS107291-01 and OSF Healthcare.

REFERENCES

- [1] Eric Anderson, Gilman D Veith, and David Weininger. 1987. *SMILES, a line notation and computerized interpreter for chemical structures*. US Environmental Protection Agency, Environmental Research Laboratory.
- [2] G Richard Bickerton, Gaia V Paolini, Jrmy Besnard, Sorel Muresan, and Andrew L Hopkins. 2012. Quantifying the chemical beauty of drugs. *Nature chemistry* 4, 2 (2012), 90.
- [3] Cdric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. 2018. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *International Conference on Machine Learning (ICML)* (2018), 1039–1048.
- [4] DE Comings, D Muhleman, and R Gysin. 1996. Dopamine D2 receptor (DRD2) gene and susceptibility to posttraumatic stress disorder: a study and replication. *Biological psychiatry* 40, 5 (1996), 368–372.
- [5] Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning (ICML)*. 2702–2711.
- [6] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. 2018. Syntax-directed variational autoencoder for structured data. *The International Conference*

- on *Learning Representations (ICLR)* (2018).
- [7] Payel Das, Kahini Wadhawan, Oscar Chang, Tom Sercu, Cicero Dos Santos, Matthew Riemer, Inkit Padhi, Vijil Chenthamarakshan, and Aleksandra Molsilovic. 2018. PepCVAE: Semi-Supervised Targeted Design of Antimicrobial Peptide Sequences. *arXiv preprint arXiv:1810.07743* (2018).
 - [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)* (2018).
 - [9] Alexander G Dosseter, Edward J Griffen, and Andrew G Leach. 2013. Matched molecular pair analysis in drug discovery. *Drug Discovery Today* 18, 15-16 (2013), 724–731.
 - [10] Peter Ertl and Ansgar Schuffenhauer. 2009. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics* 1, 1 (2009), 8.
 - [11] Tianfan Fu, Cao Xiao, Lucas Glass, and Jimeng Sun. 2021. MOLER: Incorporate Molecule-Level Reward to Enhance Deep Generative Model for Molecule Optimization. *IEEE Transactions on Knowledge and Data Engineering* (2021).
 - [12] Tianfan Fu, Cao Xiao, Xinhao Li, M. Lucas Glass, and Jimeng Sun. 2021. MIMOSA: Multi-constraint Molecule Sampling for Molecule Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence* (2021).
 - [13] Tianfan Fu, Cao Xiao, and Jimeng Sun. 2020. CORE: Automatic Molecule Optimization using Copy and Refine Strategy. *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, (2020)*, 638–645.
 - [14] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamin Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* 4, 2 (2018), 268–276.
 - [15] Ed Griffen, Andrew G Leach, Graeme R Robb, and Daniel J Warner. 2011. Matched molecular pairs as a medicinal chemistry tool: miniperspective. *Journal of medicinal chemistry* 54, 22 (2011), 7739–7750.
 - [16] Kaiming He, Ross Girshick, and Piotr Dollár. 2019. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 4918–4927.
 - [17] Shion Honda, Shoi Shi, and Hiroki R Ueda. 2019. SMILES transformer: pre-trained molecular fingerprint for low data drug discovery. *arXiv preprint arXiv:1911.04738* (2019).
 - [18] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
 - [19] Kexin Huang et al. 2021. Therapeutics data Commons: machine learning datasets and tasks for therapeutics. *arXiv preprint arXiv:2102.09548* (2021).
 - [20] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. *International Conference on Machine Learning (ICML)* (2018).
 - [21] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2019. Multi-resolution Autoregressive Graph-to-Graph Translation for Molecules. *arXiv preprint arXiv:1907.11223* (2019).
 - [22] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2020. Hierarchical Generation of Molecular Graphs using Structural Motifs. *International Conference on Machine Learning (ICML)* (2020).
 - [23] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. 2019. Learning Multimodal Graph-to-Graph Translation for Molecular Optimization. *The International Conference on Learning Representations (ICLR)* (2019).
 - [24] Simon Kornblith, Jonathon Shlens, and Quoc V Le. 2019. Do better imagenet models transfer better?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2661–2671.
 - [25] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning (ICML) -Volume 70*. JMLR. org, 1945–1954.
 - [26] Greg Landrum et al. 2006. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling.
 - [27] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. 2018. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 7795–7804.
 - [28] Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3728–3738.
 - [29] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. 2017. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*. 9, 1 (Sept. 2017), 48.
 - [30] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. 2017. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics* 9, 1 (2017), 48.
 - [31] David Rogers and Mathew Hahn. 2010. Extended-connectivity fingerprints. *Journal of chemical information and modeling* 50, 5 (2010), 742–754.
 - [32] Teague Sterling and John J Irwin. 2015. ZINC 15—ligand discovery for everyone. *Journal of chemical information and modeling* 55, 11 (2015), 2324–2337.
 - [33] Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. 2019. SMILES-BERT: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (ACM-BCB)*. 429–436.
 - [34] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. 2018. Graph Convolutional Policy Network for Goal-directed Molecular Graph Generation. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems (NIPS’18)*. Curran Associates Inc., USA, 6412–6422.
 - [35] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. 2019. Optimization of molecules via deep reinforcement learning. *Scientific reports* 9, 1 (2019), 10752.