Improving Molecular Design by Stochastic Iterative Target Augmentation

Kevin Yang [1] **Wengong Jin** [2] **Kyle Swanson** [3] **Regina Barzilay** [2] **Tommi Jaakkola** [2]

Abstract

Generative models in molecular design tend to be richly parameterized, data-hungry neural models, as they must create complex structured objects as outputs. Estimating such models from data may be challenging due to the lack of sufficient training data. In this paper, we propose a surprisingly effective self-training approach for iteratively creating additional molecular targets. We first pre-train the generative model together with a simple property predictor. The property predictor is then used as a likelihood model for filtering candidate structures from the generative model. Additional targets are iteratively produced and used in the course of stochastic EM iterations to maximize the log-likelihood that the candidate structures are accepted. A simple rejection (re-weighting) sampler suffices to draw posterior samples since the generative model is already reasonable after pre-training. We demonstrate significant gains over strong baselines for both unconditional and conditional molecular design. In particular, our approach outperforms the previous state-of-the-art in conditional molecular design by over 10% in absolute gain. Finally, we show that our approach is useful in other domains as well, such as program synthesis.

1. Introduction

The goal of molecular generation is to create molecules with the desired property profile. This task is a key component of pharmaceutical drug discovery, and has received intense attention in recent years, yielding a wide range of proposed architectures (You et al., 2018; Olivecrona et al., 2017; Popova et al., 2018; Jin et al., 2019a). A common feature of these architectures is reliance on a large number of parameters to generate molecules, which are represented as complex graph-structured objects. As a result, these models require copious amounts of training data, consisting of molecules with their target properties. Collecting such property data is often slow and expensive due to the required empirical measurements.

Our challenge is to achieve high-quality molecular generation in data-sparse regimes. While semi-supervised methods for representation learning have demonstrated significant benefits in natural language processing and computer vision (Edunov et al., 2018; Lee, 2013), they are relatively under-explored in chemistry. In this paper, we propose a simple and surprisingly effective self-training approach for iteratively creating additional molecular targets. This approach can be broadly applied to any generative architecture, without any modifications.

Our stochastic iterative target augmentation approach, shown in Figure 2, builds on the idea that it is easier to evaluate the properties of candidate molecules than to generate those molecules. Thus a learned property predictor can be used to effectively guide the generation process. To realize this idea, our method starts by pre-training the generative model on a small supervised dataset along with the property predictor. The property predictor then serves as a likelihood model for filtering candidate molecules from the generative model. Candidate generations that pass this filtering become part of the training data for the next training epoch. Theoretically, this procedure can be viewed as one iteration of stochastic EM, maximizing the log-likelihood that the candidate structures are accepted. As the generative model already produces reasonable samples after pre-training, a simple rejection (re-weighting) sampler suffices to draw posterior samples. For this reason, it is helpful to apply the filter at test time as well, or to use the approach transductively[1] to further adapt the generation process to novel test cases. The approach is reminiscent of self-training or reranking approaches employed with some success for parsing (McClosky et al., 2006; Charniak et al., 2016). However, in our case, it is the candidate generator that is complex while the filter is relatively simple and remains fixed during the iterative process.

We demonstrate that our target augmentation algorithm is

[1]UC Berkeley [2]MIT [3]University of Cambridge. Correspondence to: Kevin Yang <yangk@berkeley.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]Allowing the model to access test set inputs (but not targets) during training.

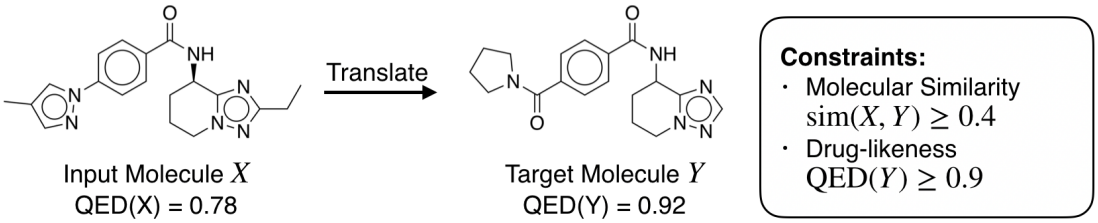

Figure 1. Illustration of conditional molecular design. Molecules can be modeled as graphs, with atoms as nodes and bonds as edges. Here, the task is to train a translation model to modify a given input molecule into a target molecule with higher drug-likeness (QED) score. The constraint has two components: the output Y must be highly drug-like, and must be sufficiently similar to the input X.

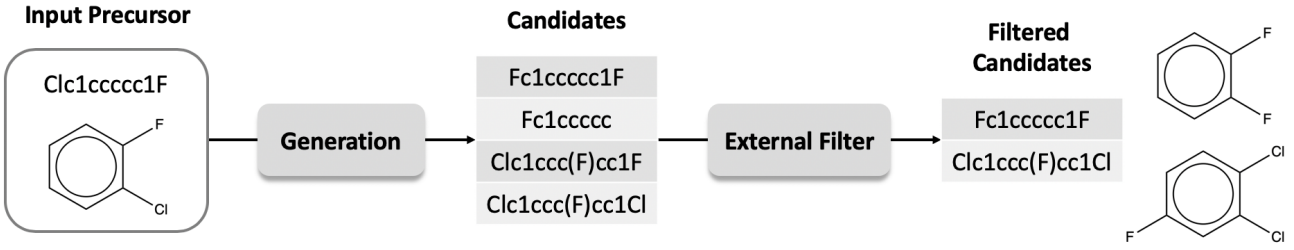

Figure 2. Illustration of data generation process for conditional molecular design. Given an input molecule, we first use our generative model to generate candidate modifications, and then select sufficiently similar molecules with high property score using our external filter. In the unconditional setting where the model takes no input, we simply sample outputs from the model and filter by property score.

effective and consistent across different generation tasks in its ability to improve molecular design performance. Our method is tested in two scenarios: molecular generative modeling (i.e., unconditional molecular design) and graph-to-graph translation, the corresponding conditional design problem of modifying an existing molecule to improve its properties. The latter is illustrated in Figure 1. We demonstrate significant gains over strong baselines for both settings. For instance, our approach outperforms the previous state-of-the-art (Jin et al., 2019a) in conditional molecular design by over 10% in absolute gain on two tasks.

Finally, our proposed method is not tied specifically to the molecular domain, and can generalize to any conditional or unconditional generation task with task-specific constraints. For example, in program synthesis, we show that our method outperforms a strong reinforcement learning baseline (Bunel et al., 2018).

2. Stochastic Iterative Target Augmentation

We present our method in the context of conditional molecular design (Jin et al., 2019a;b), the task of transforming a given molecule X into another compound Y with improved chemical properties, while constraining Y to remain similar to X (Figure 1). The corresponding unconditional task takes no input, seeking only to generate molecules with desired properties.

As our method can be adapted to the unconditional setting by just dropping the input conditioning, we present our method in the conditional context. For a given input X, the model learns to generate an output Y satisfying $\boldsymbol{c} = 1|X, Y$ for some constraint $\boldsymbol{c}$, represented as a binary random variable whose value is a function of X and Y. (That is, $\boldsymbol{c}$ corresponds to our filter.) For example, in conditional molecular generation, $\boldsymbol{c} = 1$ if Y exceeds a specified property score threshold while being sufficiently similar to X. The proposed augmentation framework can be applied to any translation model P trained on an existing dataset $\mathcal{D} = \{(X_i, Y_i)\}$, independent of the specific model architecture. As illustrated in Figure 2, our method is an iterative procedure in which each iteration consists of the following two steps:

- **Augmentation Step**: Let $\mathcal{D}$ be the original dataset and $\mathcal{D}_t$ the training set at iteration t. To construct the next epoch's augmented training set $\mathcal{D}_{t_1}$, we first initialize $\mathcal{D}_{t+1} = \mathcal{D}$. We then feed each input $X_i \in \mathcal{D}$ into the translation model up to C times to sample candidate translations $Y_i^1 \ldots Y_i^C$.[2] We take the first K distinct translations for each X_i satisfying the constraint $\boldsymbol{c}$ and add them to $\mathcal{D}_{t+1}$. When we do not find K distinct valid translations, we simply add copies of the original translation Y_i to $\mathcal{D}_{t+1}$ to preserve balance. In the unconditional setting,

[2]One could initialize $\mathcal{D}_{t+1} = \mathcal{D}_t$ instead of $\mathcal{D}_{t+1} = \mathcal{D}$ and continuously expand the dataset, but the empirical effect is small (see Appendix E.6). Note our augmentation step can be trivially parallelized for speed.

Algorithm 1 Stochastic iterative target augmentation

Input: Data $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, model $P^{(0)}$

procedure AUGMENTDATASET($\mathcal{D}$, $P^{(t)}$)
  $\mathcal{D}_{t+1} = \mathcal{D}$ ▷ Initialize augmented dataset
  for (X_i, Y_i) in $\mathcal{D}$ **do**
    for attempt in $1, \ldots, C$ **do**
      Apply $P^{(t)}$ to X_i to sample candidate Y'
      if $\boldsymbol{c} = 1|X_i, Y'$ and $(X_i, Y') \notin \mathcal{D}_{t+1}$ **then**
        Add (X_i, Y') to $\mathcal{D}_{t+1}$
      if K successful translations added **then**
        break from loop
  return augmented dataset $\mathcal{D}_{t+1}$

procedure TRAIN($\mathcal{D}$)
  for epoch in $1, \ldots, n_1$ **do** ▷ Regular training
    Train model on $\mathcal{D}$.
  for epoch in $1, \ldots, n_2$ **do** ▷ Augmentation
    $\mathcal{D}_{t+1}$ = AUGMENTDATASET($\mathcal{D}$, $P^{(t)}$)
    $P^{(t+1)} \leftarrow$ Train model $P^{(t)}$ on $\mathcal{D}_{t+1}$.

we instead just sample up to $C|\mathcal{D}|$ outputs and accept up to $K|\mathcal{D}|$ distinct new targets.

- **Training Step**: We continue to train the model $P^{(t)}$ over the new training set $\mathcal{D}_{t+1}$ for one epoch.

The above training procedure is summarized in Algorithm 1. As the constraint $\boldsymbol{c}$ is known a priori, we can construct an external property filter to remove generated outputs that violate $\boldsymbol{c}$ during the augmentation step. At test time, we also use this filter to screen predicted outputs. To propose the final translation of a given input X, we sample up to L outputs from the model until we find one satisfying the constraint $\boldsymbol{c}$. If all L attempts fail for a particular input, we output the first of the failed attempts.

Finally, as an additional improvement specific to the conditional setting, we observe that the augmentation step can be carried out for unlabeled inputs X that have no corresponding Y. Thus we can further augment our training dataset in the transductive setting by including test set inputs during the augmentation step, or in the semi-supervised setting by simply including unlabeled inputs.

3. Algorithm Motivation

We provide here some theoretical motivation for our method in the conditional setting. Since molecules are discrete objects, we assume a discrete output space.

In the conditional context, the primary difficulty lies in generalizing to unseen inputs (precursors) at test time. Generating even a single successful Y for a given X is nontrivial. Therefore, we focus on maximizing the model's probability of generating successful translations.

We can characterize our method as a stochastic expectation-maximization (EM) algorithm (Celeux et al., 1996). As before, our external filter $\boldsymbol{c}$ is a binary random variable whose value is a function of X and Y, representing whether output Y satisfies the desired constraint in relation to input X. We would like to generate Y such that $Y \in B(X) \stackrel{def}{=} \{Y' : \boldsymbol{c} = 1|X, Y'\}$. If the initial translation model $P^{(0)}(Y|X)$ (after bootstrapping on the gold data, but before our augmentation) serves as a reasonable prior distribution over outputs Y for any given input X, we could simply "invert" the filter and use

$$P^{(*)}(Y|X) \propto P^{(0)}(Y|X) \cdot p(\boldsymbol{c} = 1|X, Y) \tag{1}$$

as the ideal translation model, noting that the probability $p(\boldsymbol{c} = 1|X, Y)$ is either 0 or 1 since $\boldsymbol{c}$ is a function of X and Y. This posterior calculation is typically infeasible but can be approximated through sampling; even so, it relies heavily on the appropriateness of the prior $P^{(0)}(Y|X)$. Instead, we go a step further and iteratively optimize our parametrically defined prior translation model $P_\theta(Y|X)$. Note that the resulting prior can become much more concentrated around acceptable translations.

We maximize the log-likelihood that candidate translations satisfy the constraints implicitly encoded in the filter:

$$\mathbb{E}_X \left[\log P_\theta(\boldsymbol{c} = 1 \mid X)\right] \tag{2}$$

In many cases there are multiple viable outputs for any given input X. The training data may provide only one (or none) of them. Therefore, we treat the output structure Y as a latent variable, and expand the inner term of Eq.(2) as

$$\log \sum_Y P_\theta(Y|X) \cdot p(\boldsymbol{c} = 1|X, Y) \tag{3}$$

Since the above objective involves discrete latent variables Y, we propose to maximize Eq.(3) using the standard EM algorithm, especially its incremental, approximate variant. The target augmentation step in our approach is a sampled version of the E-step where the posterior samples are drawn with rejection sampling guided by the filter. The number of samples K controls the quality of approximation to the posterior.[3] The additional training step based on the augmented targets corresponds to a generalized M-step (though improvement is not guaranteed due to stochasticity). More precisely, let $P_\theta^{(t)}(Y|X)$ be the current translation model after t epochs of augmentation training. In epoch $t + 1$, the augmentation step first samples C different candidates for each input X using the old model $P^{(t)}$ parameterized by $\theta^{(t)}$, and then removes those which violate the constraint $\boldsymbol{c}$; the

[3] See Appendix E.6 for details on the effect of sample size K.

remaining candidates are interpretable as samples from the current posterior $Q^{(t)}(Y|X) \propto P_\theta^{(t)}(Y|X)p(\boldsymbol{c}=1|X,Y)$. As a result, the training step maximizes the EM auxiliary objective via stochastic gradient descent:

$$J(\theta \mid \theta^{(t)}) = \mathbb{E}_X \left[\sum_Y Q^{(t)}(Y|X) \log P_\theta(Y|X) \right] \quad (4)$$

We train the model with multiple iterations and show empirically that model performance indeed keeps improving as we add more iterations, both in our main experiments as well as on a toy model in Appendix B. The EM approach is likely to converge to a different and better-performing translation model than the initial posterior calculation discussed in Equation 1.

4. Experiments

We present experiments showcasing the effectiveness of our method, starting with conditional molecular design.

4.1. Conditional Molecular Design

The goal of conditional molecular design is to modify molecules to improve their chemical properties. As illustrated in Figure 1, conditional molecular design is formulated as a graph-to-graph translation problem. The training data is a set of molecular pairs $\mathcal{D} = \{(X_i, Y_i)\}$. X is the input precursor and Y is a similar molecule with improved properties. Each molecule is further labeled with its property score. Our method is well-suited to conditional molecular design because the target molecule is not unique: each precursor can be modified in many different ways to optimize its properties. Thus we can potentially discover several new targets per precursor during data augmentation.

External Filter The constraint contains two parts: 1) the chemical property of Y must exceed a certain threshold β, and 2) the molecular similarity between X and Y must exceed a certain threshold δ. The molecular similarity $\mathrm{sim}(X, Y)$ is defined as Tanimoto similarity on Morgan fingerprints (Rogers & Hahn, 2010), which measures structural overlap between two molecules.

In real-world settings, ground truth values of chemical properties are often evaluated through experimental assays, which are too expensive and time-consuming to run for stochastic iterative target augmentation. Therefore, we construct a proxy *in silico* property predictor F_1 to approximate the true property evaluator F_0. To train this proxy predictor, we use the molecules in the training set and their labeled property values. The proxy predictor F_1 is parameterized as a graph convolutional network and trained using the Chemprop package (Yang et al., 2019). During data augmentation, we use F_1 to filter out molecules whose predicted property score is under the threshold β.

4.1.1. Experimental Setup

We follow the evaluation setup of Jin et al. (2019b) for two conditional molecular design tasks:

1. **QED Optimization**: The task is to improve the drug-likeness (QED) of a given compound X. The similarity constraint is $\mathrm{sim}(X, Y) \geq 0.4$ and the property constraint is $\mathrm{QED}(Y) \geq 0.9$, with $\mathrm{QED}(Y) \in [0, 1]$ defined by the system of Bickerton et al. (2012).
2. **DRD2 Optimization**: The task is to optimize biological activity against the dopamine type 2 receptor (DRD2). The similarity constraint is $\mathrm{sim}(X, Y) \geq 0.4$ and the property constraint is $\mathrm{DRD2}(Y) \geq 0.5$, where $\mathrm{DRD2}(Y) \in [0, 1]$ is the predicted probability of biological activity given by the model from Olivecrona et al. (2017).

We treat the output of the *in silico* evaluators from Bickerton et al. (2012) and Olivecrona et al. (2017) as ground truth, and we use them only during test-time evaluation to simulate a real-world scenario.[4]

Evaluation Metrics. During evaluation, we are interested both in the probability that the model finds a successful modification for a given molecule, as well as the diversity of the successful modifications when there are multiple. Thus we translate each molecule in the test set $Z = 20$ times,[5] yielding candidate modifications $Y_1 \dots Y_Z$ (not necessarily distinct), and use the following two evaluation metrics:

1. *Success*: The fraction of molecules X for which *any* of the outputs $Y_1 \dots Y_Z$ meet the required similarity and property constraints (specified previously for each task). This is our main metric.
2. *Diversity*: For each molecule X, we measure the average Tanimoto distance (defined as $1 - \mathrm{sim}(Y_i, Y_j)$) between pairs within the set of successfully translated compounds among $Y_1 \dots Y_Z$. If there are one or fewer successful translations then the diversity is 0. We average this quantity across all test precursors X.

Models and Baselines. We consider the following two model architectures from Jin et al. (2019a) to show that our algorithm is not tied to specific neural architectures.

1. VSeq2Seq, a sequence-to-sequence translation model generating molecules by their SMILES string (Weininger, 1988).

[4] Although the Chemprop model we use in our filter is quite powerful, it fails to perfectly approximate the ground truth models for both QED and DRD2. The test set RMSE between our Chemprop model and the ground truth is 0.015 on the QED task and 0.059 on DRD2, where both properties range from 0 to 1.

[5] Our budget constraint Z limits the number of accesses to the *ground truth* evaluator, not the proxy predictor. In practice the ground truth evaluator is expensive while the proxy is cheap.

Model	QED Succ.	QED Div.	DRD2 Succ.	DRD2 Div.
VSeq2Seq	58.5	0.331	75.9	0.176
VSeq2Seq+ (Ours)	**89.0**	**0.470**	**97.2**	**0.361**
VSeq2Seq+, semi-supervised (Ours)*	95.0	0.471	99.6	0.408
VSeq2Seq+, transductive (Ours)*	92.6	0.451	97.9	0.358
HierGNN	76.6	0.477	85.9	0.192
HierGNN+ (Ours)	**93.1**	**0.514**	**97.6**	**0.418**

Table 1. Performance of different models on QED and DRD2 conditional generation tasks. Italicized models with + are augmented by our algorithm. Best performance for each model architecture in bold, not including models that use additional unlabeled data. *Note that the semi-supervised and transductive settings for VSeq2Seq are not directly comparable to VSeq2Seq and VSeq2Seq+ due to using additional unlabeled data. However, they show that having access to such unlabeled inputs can substantially improve performance. But we emphasize that iterative target augmentation remains critical to performance in these settings: augmentation without an external filter instead decreases performance.

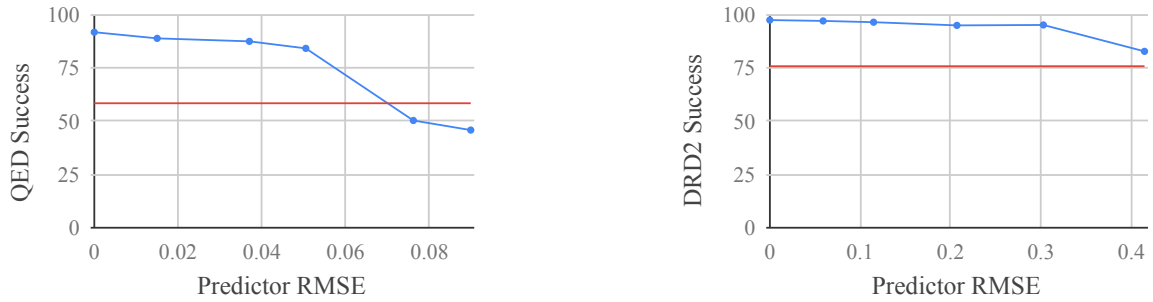

Figure 3. **Left**: QED test success rate vs. Chemprop predictor's RMSE with respect to ground truth. The red line shows the performance of the (unaugmented) VSeq2Seq baseline. **Right**: Same plot for DRD2. In each plot, the far left point with zero RMSE is obtained by reusing the ground truth predictor, while the second-from-left point is the Chemprop predictor we use to obtain our main results. Points further to the right are weaker predictors, simulating a scenario where the property is more difficult to model.

2. HierGNN, a hierarchical graph-to-graph architecture that achieves state-of-the-art performance on the QED and DRD2 tasks, outperforming VSeq2Seq by a wide margin.

We apply our iterative augmentation procedure to the above two models, generating up to $K = 4$ new targets per precursor in each augmentation epoch. Additionally, we evaluate our augmentation of VSeq2Seq in a transductive setting, as well as in a semi-supervised setting where we provide 100K additional source-side precursors from the ZINC database (Sterling & Irwin, 2015). Full hyperparameters are provided in Appendix E.1.

4.1.2. Results

As shown in Table 1, our iterative augmentation paradigm significantly improves the performance of VSeq2Seq and HierGNN. On both datasets, the translation success rate increases by over 10% in absolute terms for both models. In fact, VSeq2Seq+, our augmentation of the simple VSeq2Seq model, outperforms the non-augmented version of HierGNN. This result strongly confirms our hypothesis about the inherent challenge of learning translation models in data-sparse scenarios. Moreover, we find that adding more precursors during data augmentation further improves the VSeq2Seq model. On the QED dataset, the translation success rate improves from 89.0% to 92.6% by just adding test set molecules as precursors (VSeq2Seq+, transductive). When instead adding 100K precursors from the external ZINC database, the performance further increases to 95.0% (VSeq2Seq+, semi-supervised). We observe similar improvements for the DRD2 task as well. Beyond accuracy gain, our augmentation strategy also improves the diversity of generated molecules. For instance, on the DRD2 task, our approach yields a 100% relative gain in output diversity.

These improvements over the baselines are perhaps unsurprising when considering the much greater amount of augmented "data" pairs seen by our augmented model. For example, VSeq2Seq+ has seen over 20 times as much "data" as the base model by the end of training on the QED task (Figure 4).

Model	Train-Aug	Train+	Test+	QED Succ.	QED Div.	DRD2 Succ.	DRD2 Div.
VSeq2Seq	✗	✗	✗	58.5	0.331	75.9	0.176
VSeq2Seq(test)	✗	✗	✓	77.4	**0.471**	87.2	0.200
VSeq2Seq(train)	✓	✓	✗	81.8	0.430	92.2	0.321
VSeq2Seq+	✓	✓	✓	**89.0**	**0.470**	**97.2**	**0.361**
VSeq2Seq(no-filter)	✓	✗	✗	47.5	0.297	51.0	0.185

Table 2. Ablation analysis of filtering at training and test time. "Train-Aug" indicates a model whose training process uses self-generated candidates to augment the data, while "Train+" is a model that additionally filters these candidates using the proxy according to our framework. "Test+" indicates a model that filters outputs at prediction time using the learned proxy predictor. We emphasize that the ground truth predictor is used only for final evaluation. The evaluation for VSeq2Seq(no-filter) is conducted after 10 augmentation epochs, as the best validation set performance only decreases over the course of training.

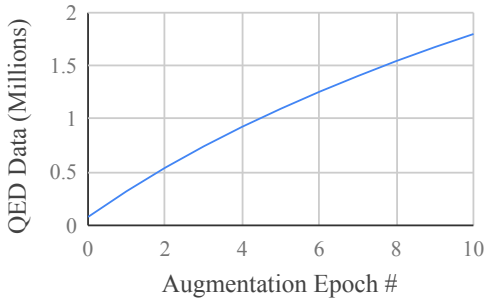

Figure 4. Cumulative number of unique training pairs seen by VSeq2Seq+ model after each augmentation epoch, on QED task.

Importance of Property Predictor Although the property predictor used in data augmentation differs from the ground truth property evaluator used at test time, the difference in evaluators does not derail the overall training process. Here we analyze the influence of the quality of the property predictor used in data augmentation. Specifically, we rerun our experiments using less accurate proxy predictors for our external filter. We obtain these weakened predictors by undertraining Chemprop and decreasing its hidden dimension. For comparison, we also report results with the oracle property predictor which is the ground truth evaluator.

As shown in Figure 3, on the DRD2 dataset we can maintain strong performance despite using predictors that deviate significantly from the ground truth. This implies that our framework can potentially be applied to other properties that are harder to predict. On the QED dataset, our method is less tolerant of inaccurate property prediction because the property constraint is much tighter — it requires the QED score of an output Y to be in the range [0.9, 1.0].

Importance of External Filtering Our full model VSeq2Seq+ uses the external filter during both training and testing. We further experiment with Vseq2seq(test), a version of our model trained without data augmentation but which uses the external filter to remove invalid outputs at test time. As shown in Table 2, VSeq2Seq(test) performs significantly worse than our full model trained under data augmentation. Similarly, a model VSeq2Seq(train) trained with data augmentation but without prediction time filtering also performs much worse than the full model.

We also run an augmentation-only version of the model without an external filter. This model (referred to as VSeq2Seq(no-filter) in Table 2) augments the data in each epoch by simply using the first K distinct candidate translations for each training precursor X, without using the external filter at all. We additionally provide this model with the 100K unlabeled precursors from the semi-supervised setting. Nevertheless, we find that during augmentation, this model's performance steadily declines from that of the bootstrapped prior. Thus the external filter is necessary to prevent poor targets from leading the model training astray.

4.2. Unconditional Molecular Design

In unconditional molecular design, we learn a distribution over molecules with desired properties. The setup is similar to the conditional case, and we reuse the same QED and DRD2 datasets. However, as there is no input in the unconditional case, we drop the precursors X and use only the set of targets Y as our training data. Additionally, we drop the similarity component from our external filter; we now require only that each generated molecule has sufficiently high property score. We use the same property thresholds for the QED and DRD2 tasks as in the conditional case.

Evaluation Metrics. We modify our metrics for the unconditional case:

1. *Success*: The fraction of sampled molecules Y above the property score threshold.
2. *Uniqueness*: The number of unique molecules generated in 20000 samples passing the property score threshold, as a fraction of 20000. This is our main metric.

In the unconditional case, a model can achieve perfect success and high pairwise diversity simply by memorizing a small number of molecules with high property score. Therefore, uniqueness is our main metric in the unconditional

Model	QED Succ.	QED Uniq.	DRD2 Succ.	DRD2 Uniq.
VSeq	62.4	0.499	51.4	0.221
VSeq+ (Ours)	**95.8**	**0.957**	**92.8**	**0.927**
REINVENT	61.9	0.610	**92.2**	0.686

Table 3. Performance of different models on QED and DRD2 unconditional generation tasks. VSeq+ is our full augmented model.

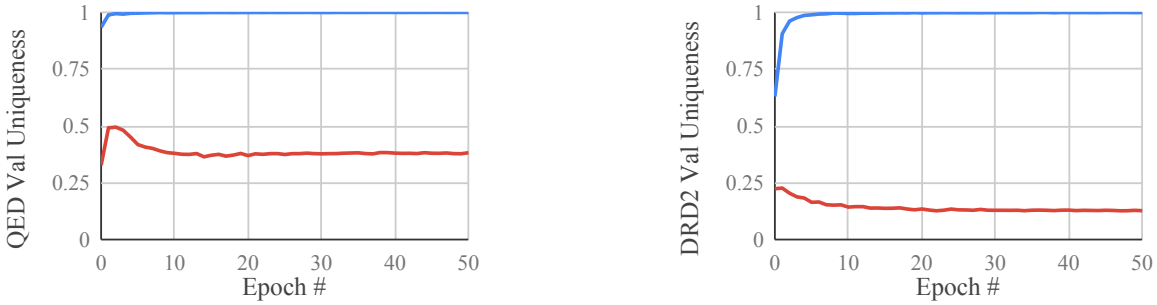

Figure 5. **Left**: Epoch number vs. uniqueness, evaluated with the Chemprop proxy predictor, for VSeq-based models on QED dataset. VSeq+ and VSeq in blue and red respectively. **Right**: Same plot for DRD2. VSeq+ is trained without iterative target augmentation for the initial epoch 0, and trained with augmentation thereafter.

setting, as a diverse distribution of molecules with high property scores is necessary to achieve high uniqueness.

Models and Baselines. We consider two baselines:

1. A modified version of VSeq2Seq which simply drops the input and corresponding attention layers; the resulting model is essentially a variational autoencoder (Kingma & Welling, 2013). We refer to this model as VSeq.
2. REINVENT, a sequence-based model from Olivecrona et al. (2017) which uses the external property scorer to fine-tune the model via reinforcement learning. This can be viewed as an alternate method of leveraging the external filter. We note that although Olivecrona et al. (2017) also originally evaluated on the DRD2 property, our setup is more challenging: we allow significantly less training data for bootstrapping, and prohibit the use of the ground truth predictor before test time.

REINVENT and our augmented model VSeq+ (obtained by augmenting VSeq) are trained to convergence. For VSeq, whose uniqueness score decreases with prolonged training, we choose the checkpoint maximizing uniqueness under the Chemprop proxy predictor. Although the VSeq and REINVENT architectures differ slightly, we match the number of trainable parameters. We provide full hyperparameters and ablations in Appendices E.1 and E.8 respectively.

4.2.1. Results

As shown in Table 3, our iterative augmentation scheme significantly improves the performance of VSeq, especially in uniqueness. In fact, uniqueness steadily decreases over time for the VSeq baseline as it overfits the training data (Figure 5). On the other hand, our augmented model VSeq+ sees a steady increase in uniqueness over time.

Moreover, our iterative augmentation scheme outperforms the REINVENT baseline on both tasks by over 0.2 in absolute terms. Especially on the QED task, the REINVENT algorithm struggles to generate high-property molecules consistently, performing comparably to the unaugmented VSeq baseline in success rate. Additionally, we observed that the REINVENT model is sometimes unstable on our DRD2 task, where the initial training dataset is smaller. Meanwhile, VSeq+ showed consistently strong performance on both tasks. Overall our experiments in this unconditional setting indicate that stochastic iterative target augmentation, at least in certain scenarios, is capable of leveraging the external property signal more effectively than an RL method.

4.3. Program Synthesis Experiments

Finally, we present additional experiments using the conditional version of our method in the program synthesis domain, demonstrating its generalizability across domains. Program synthesis is the task of generating a program (using domain-specific language) based on given input-output specifications (Bunel et al., 2018; Gulwani, 2011; Devlin et al., 2017). That is, the source is a set of input-output specifications for the program, and the target is a program that passes all test cases. Our method is suitable for this task because the target program is not unique. Multiple programs may be consistent with the given input-output specifications.

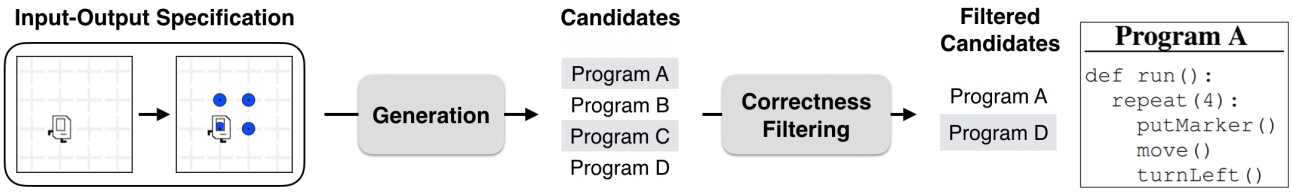

Figure 6. Illustration of our data generation process in the program synthesis setting. Given an input-output specification, we first use our generation model to generate candidate programs, and then select correct programs using our external filter. Images of input-output specification and the program A are from Bunel et al. (2018).

External Filter The external filter is straightforward for this task: we simply check whether the generated output passes all test cases. Note that at evaluation time, each instance contains extra held-out input-output test cases; the program must pass these in addition to the given test cases to be considered correct. When we perform prediction time filtering, we do not use held-out test cases in our filter.

4.3.1. Experimental Setup

Our task is based on the educational Karel programming language (Pattis, 1981) used for evaluation in Bunel et al. (2018) and Chen et al. (2019). Commands in the Karel language guide a robot's actions in a 2D grid, and may include for loops, while loops, and conditionals. Figure 6 contains an example. We follow the experiment setup of Bunel et al. (2018).

Evaluation Metrics. The evaluation metric is top-1 generalization. This metric measures how often the model can generate a program that passes the input-output test cases on the test set. At test time, we use our model to generate up to L candidate programs and select the first one to pass the input-output specifications (not including held-out test cases).

Models and Baselines. Our main baseline is the MLE baseline from Bunel et al. (2018). This model consists of a CNN encoder for the input-output grids and an LSTM decoder along with a hand-coded syntax checker. It is trained to maximize the likelihood of the provided target program. Our model is the augmentation of this MLE baseline by our iterative target augmentation framework. As with molecular design, we generate up to $K = 4$ new targets per precursor during each augmentation step. Additionally, we compare against the best model from Bunel et al. (2018), which fine-tunes the same MLE architecture using an RL method with beam search to estimate gradients.[6] We use the same hyper-parameters as the original MLE baseline; see Appendix E.1 for details.

Model	Top-1
MLE	71.91
MLE + RL + Beam Search	77.12
MLE+ (Ours)	**85.02**

Table 4. Model performance measured by top-1 generalization accuracy on Karel program synthesis task. MLE+ is our augmented version of the MLE model (Bunel et al., 2018), while MLE + RL + Beam Search is their reinforcement learning method applied to the same architecture.

4.3.2. Results

Table 4 shows the performance of our model in comparison to previous work. Our model (MLE+) outperforms the base MLE model in Bunel et al. (2018) model by a wide margin. Moreover, our model outperforms the best reinforcement learning model (RL + Beam Search) in Bunel et al. (2018), which was trained to directly maximize the generalization metric. This demonstrates the efficacy of our approach in the program synthesis domain. Since our method is complementary to architectural improvements, we hypothesize that other techniques, such as execution based synthesis (Chen et al., 2019), can benefit from our approach as well.

5. Related Work

Molecular Design Several previous works explore molecular design using different architectures. Segler et al. (2017); Kusner et al. (2017); Gómez-Bombarelli et al. (2018); Kang & Cho (2018) adopt generative modeling approaches for molecular design. You et al. (2018); Popova et al. (2018); Olivecrona et al. (2017) use reinforcement learning methods for this task. Jin et al. (2019a;b) formulate this problem as graph-to-graph translation and significantly outperform previous methods in the conditional setting. However, their

[6] More recently, Chen et al. (2019) achieved state-of-the-art performance on the same Karel task, with top-1 generalization accuracy of 92%. They use a different architecture highly specialized for program synthesis as well as a specialized ensemble method. Thus their results are not directly comparable to our results in this paper for the MLE architecture.

performance remains imperfect due to the limited size of given training sets.

On the other hand, recent advances in graph convolutional networks (Duvenaud et al., 2015; Gilmer et al., 2017) have provided effective solutions for the related problem of property prediction. Our work leverages strong property prediction models to improve the performance of generative models for molecular design, by checking whether generated molecules have desired chemical properties and augmenting the training set with molecules passing the property filter.

Program Synthesis When correctness in program synthesis is defined by input-output test cases (Bunel et al., 2018; Gulwani, 2011; Devlin et al., 2017), one can check a generated program's correctness by simply executing it on each input and verifying its output. Indeed, Zhang et al. (2018); Chen et al. (2019) use this idea in their respective decoding procedures, while also using structural constraints on valid programs. We leverage this ability to check correctness during training time data augmentation as well.

Reward-guided Generation Recent work has proposed to incorporate rewards (e.g., properties) into generative models. In machine translation, Norouzi et al. (2016) propose reward augmented maximum likelihood, which samples new targets from a *stationary* exponentiated payoff distribution centered at a ground truth target based on edit distance. Their approach is only viable when ground truth targets are given. In the case of molecular design, the number of ground truth targets is very limited. Our approach, based on stochastic EM, samples new targets from a learned non-stationary distribution which is not tied to any ground truth.

Jaques et al. (2017) use reinforcement learning to impose task-specific rewards for sequence generation, while Brookes et al. (2019b) propose an adaptive sampling approach which generates additional targets based on parametric conditional density estimation. In contrast to these two approaches, our method is based on maximum likelihood and stochastic EM; Brookes et al. (2019a) explore additional theoretical connections.

Semi-supervised Learning Our method is related to various approaches to semi-supervised learning in different domains. In chemistry, Hu et al. (2019) and Sun et al. (2019) demonstrate pre-training approaches which use unlabeled molecules to learn initial representations for property prediction models. Our method instead tackles the problem of molecular generation, addressing the problem of limited data by generating additional data via a self-training technique. In machine translation, back-translation (Sennrich et al., 2015; Edunov et al., 2018) creates additional translation pairs by using a backward translation system to translate unlabeled sentences from a target language into a source language. In contrast, our method works in the forward direction because many translation tasks are not symmetric.

In image and text classification, data augmentation and label guessing (Lee, 2013; Berthelot et al., 2019; Xie et al., 2019) are commonly applied to obtain artificial labels for unlabeled data. Rather than generating new source-target pairs by augmenting the source side, we augment the target side. In syntactic parsing, our method is closely related to self-training (McClosky et al., 2006). They generate new parse trees from unlabeled sentences by applying an existing parser followed by a reranker, and then treat the resulting parse trees as new training targets. However, their method is not iterative, and their reranker is explicitly trained to operate over the top k outputs of the parser; in contrast, our filter is independent of the generative model. In addition we show that our approach, which can be viewed as iteratively combining reranking and self-training, is theoretically motivated and can improve the performance of highly complex neural models. Co-training (Blum & Mitchell, 1998) and tri-training (Zhou & Li, 2005; Charniak et al., 2016) also augment a parsing dataset by adding targets on which multiple baseline models agree. Instead of using multiple learners, our method uses task-specific constraints to select correct outputs.

6. Conclusion

In this work, we have presented a stochastic iterative target augmentation framework for molecular design. Our approach is theoretically motivated, and we demonstrate strong empirical results in both the conditional and unconditional molecular design settings, significantly outperforming baseline models in each case. Moreover, we find that stochastic iterative target augmentation is complementary to architectural improvements, and that its effect can be quite robust to the external filter's quality. Finally, in principle our approach is applicable to other domains as well.

Acknowledgements

We thank Guang-He Lee, Yujia Bao, Rachel Wu, Benson Chen, Mitchell Stern, Nikita Kitaev, Ruiqi Zhong, Rudy Corona, Eric Wallace, Kevin Lin, Daniel Fried, Dan Klein, and our anonymous reviewers for their helpful comments and feedback which helped us to greatly improve the paper.

References

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S.,

and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90, 2012.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100. Citeseer, 1998.

Brookes, D. H., Busia, A., Fannjiang, C., Murphy, K., and Listgarten, J. A view of estimation of distribution algorithms through the lens of expectation-maximization. *arXiv preprint arXiv:1905.10474*, 2019a.

Brookes, D. H., Park, H., and Listgarten, J. Conditioning by adaptive sampling for robust design. *arXiv preprint arXiv:1901.10060*, 2019b.

Bunel, R., Hausknecht, M., Devlin, J., Singh, R., and Kohli, P. Leveraging grammar and reinforcement learning for neural program synthesis. *arXiv preprint arXiv:1805.04276*, 2018.

Celeux, G., Chauveau, D., and Diebolt, J. Stochastic versions of the em algorithm: an experimental study in the mixture case. *Journal of statistical computation and simulation*, 55(4):287–314, 1996.

Charniak, E. et al. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2331–2336, 2016.

Chen, X., Liu, C., and Song, D. Execution-guided neural program synthesis. *International Conference on Learning Representations*, 2019.

Devlin, J., Bunel, R. R., Singh, R., Hausknecht, M., and Kohli, P. Neural program meta-induction. In *Advances in Neural Information Processing Systems*, pp. 2080–2088, 2017.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. *Advances in Neural Information Processing Systems*, pp. 2224–2232, 2015.

Edunov, S., Ott, M., Auli, M., and Grangier, D. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *Proceedings of the 34th International Conference on Machine Learning*, 2017.

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., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 2018. doi: 10.1021/acscentsci.7b00572.

Gulwani, S. Automating string processing in spreadsheets using input-output examples. In *ACM Sigplan Notices*, volume 46, pp. 317–330. ACM, 2011.

Hastings, W. K. Monte carlo sampling methods using markov chains and their applications. 1970.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pp. 6626–6637, 2017.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1645–1654. JMLR. org, 2017.

Jin, W., Barzilay, R., and Jaakkola, T. Multi-resolution autoregressive graph-to-graph translation for molecules. *arXiv preprint arXiv:1907.11223*, 2019a.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. Learning multimodal graph-to-graph translation for molecular optimization. *International Conference on Learning Representation*, 2019b.

Kang, S. and Cho, K. Conditional molecular design with deep generative models. *Journal of chemical information and modeling*, 59(1):43–52, 2018.

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

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

Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 2, 2013.

McClosky, D., Charniak, E., and Johnson, M. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of*

Computational Linguistics, pp. 152–159. Association for Computational Linguistics, 2006.

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., Schuurmans, D., et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pp. 1723–1731, 2016.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pattis, R. E. *Karel the Robot: A Gentle Introduction to the Art of Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1981. ISBN 0471089281.

Popova, M., Isayev, O., and Tropsha, A. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7): eaap7885, 2018.

Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S., and Klambauer, G. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

Rogers, D. and Hahn, M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.*, 50(5):742–754, 2010.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. Generating focussed molecule libraries for drug discovery with recurrent neural networks. *arXiv preprint arXiv:1701.01329*, 2017.

Sennrich, R., Haddow, B., and Birch, A. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

Sterling, T. and Irwin, J. J. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

Sun, F.-Y., Hoffmann, J., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Model.*, 28(1):31–36, 1988.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*, 2019.

Yang, K., Swanson, K., Jin, W., Coley, C. W., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 2019.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pp. 6410–6421, 2018.

Zhang, L., Rosenblatt, G., Fetaya, E., Liao, R., Byrd, W. E., Urtasun, R., and Zemel, R. Leveraging constraint logic programming for neural guided program synthesis. 2018.

Zhou, Z.-H. and Li, M. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, (11):1529–1541, 2005.