A. Code Availability

All code is available at https://github.com/yangkevin2/icml2020-stochastic-iterative-target-augmentation.

B. Toy Model

We investigate the performance of our model in a toy setting, as follows. The discrete output space Ω is the set of points (x_1, x_2) such that $-1 \leq x_i \leq 3$ and x_i is a multiple of 0.05 for each $i = 1, 2$. We assume that there is no input, that is, we are operating in the unconditional setting. We define our constraint set to be the unit ball, so a new sample will pass our filter if and only if is in the unit ball. Our (nonparametric) prior is estimated using k-nearest neighbors density estimation on the dataset $\mathcal{D}$ with $k = 50$, where $\mathcal{D}$ is initialized to be the set of points in Ω where both coordinates are multiples of 0.5 (in order to achieve a more even prior distribution over Ω, even though some of these initial points are outside the constraint set).

We draw samples using Metropolis-Hastings (Hastings, 1970) with interval of 50 steps between samples, using a uniform distribution over Ω as the proposal distribution. Upon drawing a sample, we add it to $\mathcal{D}$ if it lies in the unit ball. We repeat for a total of 20000 samples; adding the correct samples to $\mathcal{D}$ corresponds to our iterative augmentation and training procedure. Finally, for evaluation, we sample an additional 2000 samples (without filtering) and plot them in Figure 7. Nearly all of the samples lie in the desired constraint set.

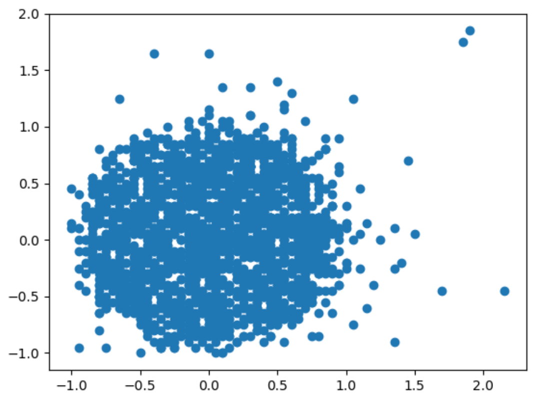


Figure 7. Distribution of samples at the end of toy model training. The unit ball is the desired constraint set, while the discrete space of possible samples is the points in $[-1, 3] \times [-1, 3]$ where both coordinates are multiples of 0.05.

C. Further Theoretical Analysis in Simplified Setting

We analyze our method further in a simplified setting in order to understand our method's ability to produce diverse outputs. In particular, compared to Section 3 in the main text, we will now drop the input X, effectively switching to the unconditional setting. The constraint $\boldsymbol{c}$ then depends only on Y. While producing diverse outputs is most important in the unconditional generation setting, our analysis here applies to the conditional setting as well, as we can view the conditional setting as a separate unconditional generation problem for each individual input.

We will demonstrate that our method indeed yields a diverse distribution over correct outputs in a simplified nonparametric, non-stochastic setting. In this setting, our model P has unlimited capacity, simulating an arbitrarily complex neural model in practice. Let $\mathcal{A} = \{Y : \boldsymbol{c} = 1|Y\}, \mathcal{B} = \{Y : \boldsymbol{c} = 0|Y\}$. Starting with a base distribution $P^{(0)}$, our objective will be to iteratively maximize $\log P(\mathcal{A})$, the log-probability that a sample from P lies in $\mathcal{A}$. We also add a KL-divergence penalty to keep $P^{(t+1)}$ close to $P^{(t)}$ because in practice, we make only a limited update to our model distribution in each training iteration, dependent on learning rate. Thus, fixing some $\lambda > 0$, we update P according to:

$$P^{(t+1)} = \arg\max_{P} \log P(\mathcal{A}) - \lambda KL(P||P^{(t)}) \quad (5)$$

where the argmax is over all possible models (distributions) P. We characterize $P^{(t+1)}$ by the following proposition, whose proof we defer to Appendix D:

Proposition 1 *Assume $P^{(0)}$ has nonzero support on $\mathcal{A}$ and $\mathcal{B}$. Let $P^{(t)}(Y)$ be the probability of sampling molecule Y from $P^{(t)}$, and $P^{(t)}(\mathcal{A})$ the probability that a given sample lies in $\mathcal{A}$. For any $\lambda > 0$, when updating P according to Equation 5, we have for all timesteps t and molecules Y:*

$$P^{(t)}(Y) = \alpha^{(t)} \frac{P^{(0)}(Y)}{P^{(0)}(\mathcal{A})} \mathbf{1}_{Y \in \mathcal{A}} + (1 - \alpha^{(t)}) \frac{P^{(0)}(Y)}{P^{(0)}(\mathcal{B})} \mathbf{1}_{Y \in \mathcal{B}} \quad (6)$$

for some sequence $\{\alpha^{(t)}\} \in [0, 1]$. Moreover, the sequence $\{\alpha^{(t)}\}$ converges to 1, with $\alpha^{(t)} \geq 1 - \epsilon$ for $\epsilon > 0$ whenever $t \geq -\lambda \log(\epsilon \alpha^{(0)})$.

From Proposition 1, we observe that the converged model $P^{(\infty)}$ assigns probability to each output proportional to $P^{(0)} p(\boldsymbol{c} = 1|Y)$. We conclude that in this simplified setting, if our starting distribution $P^{(0)}$ is reasonably diverse (for example, a randomly initialized neural generator), the resulting converged $P^{(\infty)}$ will be a diverse distribution over $\mathcal{A}$.

Remark. In practice, since molecular structures are discrete and the distribution may be peaked, it is important to properly deal with repeated samples during our augmentation step. Thus we sample targets proportional to $P^{(t)}p(\mathbf{c}=1|Y)$ *without replacement*. This diverges from our theory, which corresponds to sampling *with replacement*: the KL-divergence penalty encourages $P^{(t+1)}$ to assign probability proportional to $P^{(t)}$, rather than uniform probability across $\mathcal{A}$. In the limit as the number of samples goes to infinity, sampling targets without replacement is preferred: this encourages $P^{(\infty)}$ to be uniform over the set $\mathcal{A}$.

Lastly, we note that our analysis here applies in principle to the conditional setting as well, viewing each input precursor as a separate unconditional design task.

D. Proof of Proposition 1

We now prove Proposition 1, reproduced below for convenience.

Proposition 1 *(a) Assume $P^{(0)}$ has nonzero support on $\mathcal{A}$ and $\mathcal{B}$. Let $P^{(t)}(Y)$ be the probability of sampling molecule Y from $P^{(t)}$, and $P^{(t)}(\mathcal{A})$ the probability that a given sample lies in $\mathcal{A}$. For any $\lambda > 0$, when updating P according to Equation 5, we have Equation 6 for all timesteps t and molecules Y:*

$$P^{(t)}(Y) = \alpha^{(t)}\frac{P^{(0)}(Y)}{P^{(0)}(\mathcal{A})}\mathbf{1}_{Y\in\mathcal{A}} + (1-\alpha^{(t)})\frac{P^{(0)}(Y)}{P^{(0)}(\mathcal{B})}\mathbf{1}_{Y\in\mathcal{B}}$$

for some sequence $\{\alpha^{(t)}\} \in [0,1]$

(b) Moreover, the sequence $\{\alpha^{(t)}\}$ converges to 1, with $\alpha^{(t)} \geq 1-\epsilon$ for $\epsilon > 0$ whenever $t \geq -\lambda \log(\epsilon\alpha^{(0)})$.

Proof (a) Recall Equation 5:

$$P^{(t+1)} = \arg\max_{P} \log P(\mathcal{A}) - \lambda KL(P||P^{(t)})$$

We first prove that the optimal P exists and takes the stated form. Note that it suffices to prove the statement with $P^{(0)}(Y)$ replaced by $P^{(t)}(Y)$, as we can use induction. Each timestep t simply results in a reweighting of the sets $\mathcal{A}$ and $\mathcal{B}$ by updating α.

Define $h(P) = \log P(\mathcal{A}) - \lambda KL(P||P^{(t)})$, and define a P of the form given in Equation 6 as *proportionality-preserving*, or prop-preserving. First, we use a smoothing argument to show that for any non-prop-preserving P_0, there exists a prop-preserving P^* such that $h(P_0) < h(P^*)$.

By definition,

$$D(P) \overset{def}{=} KL(P||P^{(t)}) \tag{7}$$
$$= \sum_Y P(Y)\log P(Y) - P(Y)\log P^{(t)}(Y) \tag{8}$$

Taking the derivative with respect to $P(Y_0)$ for fixed Y_0:

$$\frac{dD(P)}{dP(Y_0)} = 1 + \log P(Y_0) - \log P^{(t)}(Y_0) \tag{9}$$
$$= 1 + \log \frac{P(Y_0)}{P^{(t)}(Y_0)} \tag{10}$$

Now for any P_0, let α_0 be the weight it assigns to $\mathcal{A}$, and let $P^*_{\alpha_0}$ be the prop-preserving P^* with parameter α_0. For all $Y_0 \in \mathcal{A}$, because $P^*_{\alpha_0}$ is prop-preserving, $\frac{P^*(Y_0)}{P^{(t)}(Y_0)}$ is equal to some constant c. Hence, $\frac{dD(P^*_{\alpha_0})}{dP^*_{\alpha_0}(Y_0)}$ is a constant k for all $Y_0 \in \mathcal{A}$.

Consider next the sets $\mathcal{A}_s$ and $\mathcal{A}_b$ which are the subsets of $\mathcal{A}$ where $\frac{P_0(Y_0)}{P^{(t)}(Y_0)} < c$ and $\frac{P_0(Y_0)}{P^{(t)}(Y_0)} > c$, respectively. Since P_0 and $P^*_{\alpha_0}$ assign the same probability to $\mathcal{A}$ as a whole, we have:

$$P_0(\mathcal{A}_s) + P_0(\mathcal{A}_b) = P^*_{\alpha_0}(\mathcal{A}_s) + P^*_{\alpha_0}(\mathcal{A}_b) \tag{11}$$

However, as the log function is strictly increasing, from 10 we see that $\frac{dD(P_0)}{dP_0(Y_0)} < k$ whenever $\frac{P_0(Y_0)}{P^{(t)}(Y_0)} < c$ (i.e. $Y_0 \in \mathcal{A}_s$) and vice versa when $\frac{dD(P_0)}{dP_0(Y_0)} > k$ (i.e. $Y_0 \in \mathcal{A}_b$). Hence for $Y_0 \in \mathcal{A}_s$, by the mean value theorem we have that replacing $P_0(Y_0)$ with $P^*_{\alpha_0}(Y_0)$ would increase $D(P_0)$ by less than $k(P^*_{\alpha_0}(Y_0) - P_0(Y_0))$. Doing this replacement for all $Y_0 \in \mathcal{A}_s$ thus increases $D(P_0)$ by less than $k(P^*_{\alpha_0}(\mathcal{A}_s) - P_0(\mathcal{A}_s))$. Similarly, replacing $P_0(Y_0)$ with $P^*_{\alpha_0}(Y_0)$ for all $Y_0 \in \mathcal{A}_b$ decreases $D(P_0)$ by more than $k(P_0(\mathcal{A}_b) - P^*_{\alpha_0}(\mathcal{A}_b))$.

However, from rearranging Equation 11 we have that $P_0(\mathcal{A}_s) - P^*_{\alpha_0}(\mathcal{A}_s) = -(P_0(\mathcal{A}_b) - P^*_{\alpha_0}(\mathcal{A}_b))$. Therefore, replacing all values of $P_0(Y_0)$ with $P^*_{\alpha_0}(Y_0)$ for $Y_0 \in \mathcal{A}$ cannot increase the value of $D(P_0)$. Moreover, if $\mathcal{A}_s$ and $\mathcal{A}_b$ were nonempty, then $D(P_0)$ strictly decreases.

We can repeat the same argument as above for the probability mass in $\mathcal{B}$. If P_0 is not prop-preserving, then either $\mathcal{A}_s$ and $\mathcal{A}_b$ are nonempty or the corresponding sets for $\mathcal{B}$ are nonempty. We conclude that for any non-prop-preserving P_0 there exists a prop-preserving $P^*_{\alpha_0}$ achieving a strictly lower value of $D(P) = KL(P||P^{(t)})$. Since h places negative weight on $D(P)$, and our value replacements did not affect the value of $\log P_0(\mathcal{A})$, we conclude that $P^*_{\alpha_0}$ achieves a strictly higher value of h than does P_0.

Next, observe that the space of possible prop-preserving P^* is one-dimensional, parameterized by $\alpha \in [0,1]$. Thus, we can define a function $h'(\alpha)$ as $h(P^*(\alpha))$. Both $\log P(\mathcal{A})$ and $-\lambda KL(P||P^{(t)}$ are upper-bounded by 0, so $h' \to -\infty$ as $\alpha \to 0$. If $P^{(t)}(\mathcal{B}) = 0$ then we have the maximum at

$\alpha = 1$, otherwise $h' \to -\infty$ as $\alpha \to 1$ as well. Since h' is continuous and in fact strictly concave in α (due to strict concavity of log and convexity of KL), we conclude that $h'(\alpha)$ attains its unique maximum for some $\alpha^* \in [0, 1]$. Then since we showed previously that every non-prop-presering P_0 achieves a value of $h(P_0)$ at strictly less than that of some prop-preserving P^*, we conclude that a unique P^* maximizing h indeed exists and is prop-preserving. □

Proof (b) We now show that the sequence $\{\alpha^{(t)}\}$ converges to 1. Since we assumed $P^{(0)}$ has nonzero support on $\mathcal{A}$, we know that $\alpha^{(0)} > 0$. If $P^{(0)}(\mathcal{B}) = 0$, then we are trivially done. So henceforth we can assume $\alpha^{(0)} \in (0, 1)$.

Noting that $\log P^*_{\alpha^{(t+1)}}(\mathcal{A}) = \alpha^{(t+1)}$ and $\log P^*_{\alpha^{(t+1)}}(\mathcal{B}) = 1 - \alpha^{(t+1)}$, we have:

$$\begin{aligned}
&KL(P^*_{\alpha^{(t+1)}}||P^{(t)}) \\
&= KL(P^*_{\alpha^{(t+1)}}||P^*_{\alpha^{(t)}}) \qquad (12)\\
&= \sum_{Y_0 \in \mathcal{A}} P^*_{\alpha^{(t+1)}}(Y_0) \log \frac{P^*_{\alpha^{(t+1)}}(Y_0)}{P^*_{\alpha^{(t)}}(Y_0)} \qquad (13)\\
&\quad + \sum_{Y_0 \in \mathcal{B}} P^*_{\alpha^{(t+1)}}(Y_0) \log \frac{P^*_{\alpha^{(t+1)}}(Y_0)}{P^*_{\alpha^{(t)}}(Y_0)} \\
&= P^*_{\alpha^{(t+1)}}(\mathcal{A}) \log \frac{\alpha^{(t+1)}}{\alpha^{(t)}} + P^*_{\alpha^{(t+1)}}(\mathcal{B}) \log \frac{1 - \alpha^{(t+1)}}{1 - \alpha^{(t)}} \qquad (14)\\
&= \alpha^{(t+1)} \log \frac{\alpha^{(t+1)}}{\alpha^{(t)}} + (1 - \alpha^{(t+1)}) \log \frac{1 - \alpha^{(t+1)}}{1 - \alpha^{(t)}} \qquad (15)
\end{aligned}$$

We are now ready to take the derivative of h with respect to $\alpha^{(t+1)}$:

$$\begin{aligned}
&\frac{dh(P^*_{\alpha^{(t+1)}})}{d\alpha^{(t+1)}} \qquad (16)\\
&= \frac{1}{\alpha^{(t+1)}} - \lambda(\log \alpha^{(t+1)} + 1 - \log \alpha^{(t)}) \qquad (17)\\
&\quad - \lambda(\log(1 - \alpha^{(t)}) - \log(1 - \alpha^{(t+1)}) - 1) \\
&= \frac{1}{\alpha^{(t+1)}} - \lambda \log \frac{\alpha^{(t+1)}}{1 - \alpha^{(t+1)}} + \lambda \log \frac{\alpha^{(t)}}{1 - \alpha^{(t)}} \qquad (18)
\end{aligned}$$

Observe that α is trivially nondecreasing: if $\alpha^{(t+1)} < \alpha^{(t)}$, then $\log P(\mathcal{A})$ decreases while $KL(P||P^{(t)})$ increases when comparing $P^*_{\alpha^{(t+1)}}$ and $P^*_{\alpha^{(t)}}$, as $P^{(t)} = P^*_{\alpha^{(t)}}$.

Moreover, the derivative $\frac{dh(P^*_{\alpha^{(t+1)}})}{d\alpha^{(t+1)}}$ is positive at $\alpha^{(t+1)} = \alpha^{(t)}$, so in fact α is strictly increasing. Since h is continuous, we have either $\alpha^{(t+1)} = 1$ or $\frac{dh(P^*_{\alpha^{(t+1)}})}{d\alpha^{(t+1)}} = 0$. Solving the latter equation gives us

$$\lambda \log \frac{\alpha^{(t)}}{1 - \alpha^{(t)}} + \frac{1}{\alpha^{(t+1)}} = \lambda \log \frac{\alpha^{(t+1)}}{1 - \alpha^{(t+1)}} \qquad (19)$$

$$\frac{\alpha^{(t)}}{1 - \alpha^{(t)}} e^{\frac{1}{\lambda \alpha^{(t+1)}}} = \frac{\alpha^{(t+1)}}{1 - \alpha^{(t+1)}} \qquad (20)$$

Now suppose for the sake of contradiction that $\{\alpha^{(t)}\}$ does not converge to 1, i.e. for some fixed $C < 1$, $\alpha^t < C$ for all t . Then from 20 we see that

$$\frac{\alpha^{(t)}}{1 - \alpha^{(t)}} e^{\frac{1}{\lambda C}} \leq \frac{\alpha^{(t+1)}}{1 - \alpha^{(t+1)}} \qquad (21)$$

Finally, since $\frac{1}{\lambda C} > 0$, we have

$$e^{\frac{1}{\lambda C}} > 1 \qquad (22)$$

We conclude from 21 and 22 that $\frac{\alpha^{(t)}}{1-\alpha^{(t)}}$ is exponentially increasing over time. This contradicts that $\alpha^t < C < 1$ for some fixed C for all t; therefore, the sequence $\{\alpha^{(t)}\}$ must converge to 1.

Finally, we analyze the rate of convergence. Suppose we want $\alpha^{(t)} \geq 1 - \epsilon$ for some $\epsilon > 0$, i.e., $\frac{\alpha^{(t)}}{1-\alpha^{(t)}} \geq \frac{1-\epsilon}{\epsilon}$. From 21 we see when $\alpha < C$, $\frac{\alpha}{1-\alpha}$ is exponentially growing by a factor of at least $e^{\frac{1}{\lambda C}}$ with each timestep. Here, we have $C = 1 - \epsilon$. Therefore, we have:

$$\frac{\alpha^{(t)}}{1 - \alpha^{(t)}} \geq \left(\frac{\alpha^{(0)}}{1 - \alpha^{(0)}}\right) e^{\frac{t}{\lambda(1-\epsilon)}} \qquad (23)$$

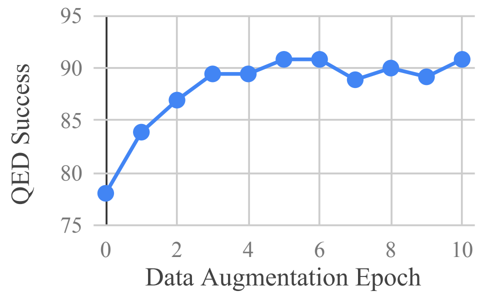
From this, we see that to achieve $\frac{\alpha^{(t)}}{1-\alpha^{(t)}} \geq \frac{1-\epsilon}{\epsilon}$, it suffices to have:

$$\left(\frac{\alpha^{(0)}}{1 - \alpha^{(0)}}\right) e^{\frac{t}{\lambda(1-\epsilon)}} \geq \frac{1 - \epsilon}{\epsilon} \qquad (24)$$

Rearranging gives us the following sufficient condition for $\alpha^{(t)} \geq 1 - \epsilon$:

$$t \geq \lambda(1 - \epsilon) \log \left(\frac{(1 - \epsilon)(1 - \alpha^{(0)})}{\epsilon \alpha^{(0)}}\right) \qquad (25)$$

Loosening the condition by observing $1 - \epsilon < 1$ and $1 - \alpha^{(0)} < 1$ gives us our desired $t \geq -\lambda \log(\epsilon \alpha^{(0)})$, although of course this bound is not tight. □



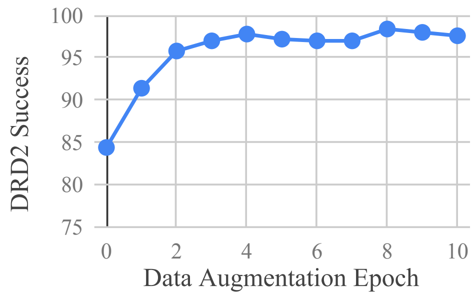


Figure 8. **Left**: Success rate for VSeq2Seq+ on validation set for each epoch of iterative target augmentation on conditional QED task. **Right**: Same plot for DRD2. For each plot, the far left point indicates the performance of the bootstrapped model.

E. Additional Experimental Details

E.1. Implementation and Hyperparameters

Our augmented models share the same hyperparameters as their baseline counterparts in all cases.

In the molecular design conditional case, for VSeq2Seq we use batch size 64, embedding and hidden dimension 300, VAE latent dimension 30, and an LSTM with depth 1 (bidirectional in the encoder, unidirectional in the decoder). For models using stochastic iterative target augmentation, n_1 is set to 5 and n_2 is set to 10, while for the baseline models we train for 20 epochs (corresponding to $n_1 = 20, n_2 = 0$). The HierGNN model shares the same hyperparameters as in Jin et al. (2019a).

In the unconditional setting, our VSeq model uses the same hyperparameters as the conditional-case VSeq2Seq model, while for the REINVENT baseline we use Olivecrona et al. (2017)'s default settings. Both models have approximately 4 million trainable parameters to facilitate fair comparison. We set n_1 to 1 and n_2 to 50, and train the VSeq baseline model for 50 epochs. We also discard the gold data altogether after the initial bootstrapping phase, as we find that this improves model performance. For the REINVENT baseline, we train their prior model for the recommended number of steps, and then finetune using their RL method until convergence. We additionally searched over their σ hyperparameter, although we found that this did not significantly affect performance on either the QED or DRD2 tasks, so our final runs use the default value of 20.

For the training time and prediction time filtering parameters, we set $K = 4$, $C = 200$, and $L = 10$ for both the QED and DRD2 tasks, in both the conditional and unconditional cases. Although we ran experiments with different values of K, we found that the change did not significantly affect performance unless K was too small; see Appendix E.6.

For the Karel program synthesis task, we use the same hyperparameters as the MLE baseline model in Bunel et al. (2018). Our augmented model shares the same hyperparameters. We use a beam size of 64 at test time, the same as the MLE baseline, but simply sample programs from the decoder distribution when running iterative target augmentation during training. The baseline model is trained for 100 epochs, while for the model employing iterative target augmentation we train as normal for $n_1 = 15$ epochs followed by $n_2 = 50$ epochs of iterative target augmentation. Due to the large size of the full training dataset, in each epoch of iterative augmentation we use $\frac{1}{10}$ of the dataset, so in total we make 5 passes over the entire dataset.

For the training time and prediction time filtering parameters, we set $K = 4$, $C = 50$, and $L = 10$.

All code is in PyTorch (Paszke et al., 2017).

E.2. Dataset Sizes

In Table 5 we provide the training, validation, and test set sizes for all of our tasks. Note that the validation and test sizes are relevant only for the conditional case. For each task we use the same splits as our baselines.

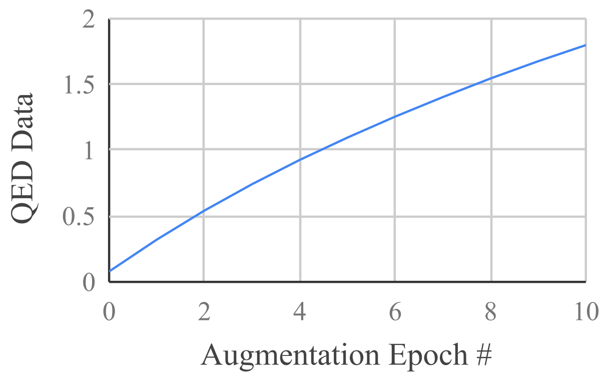
Task	Training Set	Validation Set	Test Set
QED	88306	360	800
DRD2	34404	500	1000
Karel	1116854	2500	2500

Table 5. Number of examples in training, validation, and test sets for each task.

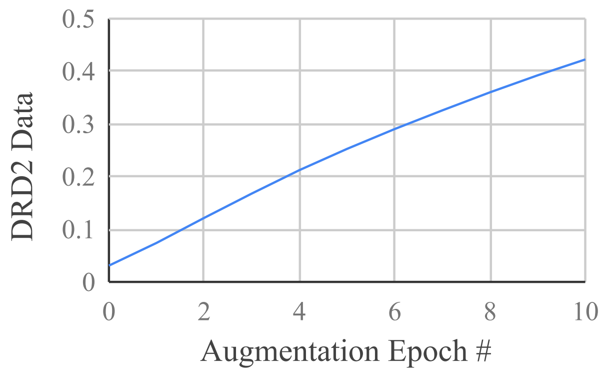
The QED data is obtained from filtering ZINC (Sterling & Irwin, 2015), while the DRD2 data is obtained from ZINC and Olivecrona et al. (2017). Our complete datasets are included together with our code submission.

E.3. Learning Curves

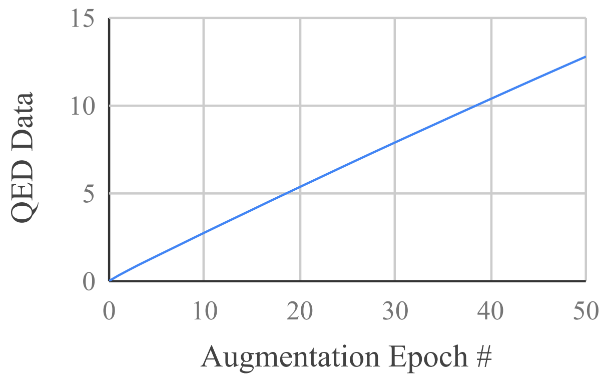
In Figure 8, we provide the validation set performance per augmentation epoch for our VSeq2Seq+ model on both the QED and DRD2 conditional tasks.



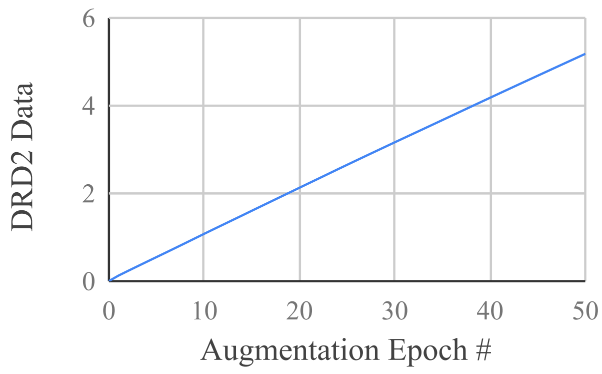
(a) QED Conditional



(b) DRD2 Conditional



(c) QED Unconditional



(d) DRD2 Unconditional

Figure 9. Graphs of the cumulative number of unique training pairs our augmented sequence-based model has seen by the time of each augmentation epoch, on both QED and DRD2 tasks in both conditional and unconditional settings. All vertical axis scales in millions.

E.4. Unique Data Seen Over Time

In Figure 9, we show the cumulative number of unique data points seen during augmentation epochs. The four subplots show the QED and DRD2 tasks for both the VSeq2Seq+ model in the conditional setting as well as the VSeq+ model in the unconditional setting. Even after several epochs, the number of unique data points is still increasing in all cases. Due to the large number of additional data points, we find that in both settings, empirical model performance at test time is limited more by the discrepancy between the proxy predictor and the ground truth evaluator than by the number of new data points seen. This is evidenced by the near-perfect performance we observe for both VSeq2Seq+ and VSeq+ when evaluated using the proxy predictor.

E.5. Frechet ChemNet Distance Analysis

As another evaluation of our model on a metric it was not optimized for, we further evaluate Frechet ChemNet Distance (FCD) (Preuer et al., 2018) between model outputs and a reference set of gold targets for both the QED and DRD2 tasks, in both the conditional and unconditional set-

Conditional		
Model	**QED**	**DRD2**
VSeq2Seq	1.34	7.74
VSeq2Seq+	1.28	7.10
Unconditional		
Model	**QED**	**DRD2**
VSeq	3.21	12.45
REINVENT	4.79	19.81
VSeq+	3.33	10.86

Table 6. FCD evaluation of baselines and augmentations on two datasets in both conditional and unconditional settings; in isolation, lower is better. Our augmentation method maintains similar FCD between outputs and gold targets compared to the baseline on QED, and decreases the distance on DRD2, while substantially improving the success rate and diversity of modifications. By contrast, the reinforcement-learning based REINVENT method greatly increases the FCD on both QED and DRD2 in the unconditional setting.

Model	QED Succ.	QED Div.	DRD2 Succ.	DRD2 Div.
VSeq2Seq+, K=2	85.1	0.453	95.9	0.327
VSeq2Seq+, K=4	89.0	0.470	97.2	0.361
VSeq2Seq+, K=8	88.4	0.480	97.6	0.373

Table 7. Performance of our model VSeq2Seq+ in the conditional setting with different values of K. All other experiments use $K = 4$.

Model	QED Succ.	QED Div.	DRD2 Succ.	DRD2 Div.
VSeq2Seq+	89.0	0.470	97.2	0.361
VSeq2Seq+, keep-targets	89.8	0.465	97.6	0.363

Table 8. Performance in conditional setting of our proposed augmentation scheme, VSeq2Seq+, compared to an alternative version (VSeq2Seq+, keep-targets) which keeps all generated targets and continually grows the training dataset.

tings. FCD is the molecular analogue of Frechet Inception Distance for images (Heusel et al., 2017), measuring distributional distance. Considering the FCD metric in isolation, we prefer models whose outputs have lower distributional distance with the reference set.

In Table 6, we observe that on the QED task our model and the baseline perform similarly. On DRD2, our augmentation method is quite successful at decreasing the distributional distance, where the distributional distances between the training targets and the reference gold targets leave more room for improvement compared to QED. Thus our method improves over the baseline in success and diversity (our main metrics in the paper) while also performing equal or better by FCD. By contrast, we observe that the REINVENT baseline heavily degrades performance on the FCD metric compared to the baseline on both tasks.

E.6. Further Molecular Design Experiments

In the conditional case, we experiment with the effect of modifying K, the number of new targets added per precursor during each training epoch. In all other experiments we have used $K = 4$. Since taking $K = 0$ corresponds to the base non-augmented model, it is unsurprising that performance may suffer when K is too small. However, as shown in Table 7, at least in the conditional case there is relatively little change in performance for K much larger than 4.

We also experiment with a version of our method which continually grows the training dataset by keeping all augmented targets, instead of discarding new targets at the end of each epoch. We chose the latter version for our main experiments due to its closer alignment to our EM motivation. However, we demonstrate in Table 8 that performance gains from continually growing the dataset are small to insignificant in our conditional molecular design tasks.

E.7. Model Stability and Number of Runs

We found that the reinforcement-learning based REINVENT model was sometimes unstable on our DRD2 dataset, resulting in wide variance in results between different runs. To confirm statistical significance, we ran VSeq+ and REINVENT 10 times each on this dataset, resulting in VSeq+ having higher uniqueness with p-value 0.003 in a t-test.

All other models were highly stable and performed consistently between runs, particularly in the conditional setting. For our final experiments we ran all models 3 times in the unconditional setting, reporting mean metrics, and once in the conditional setting.

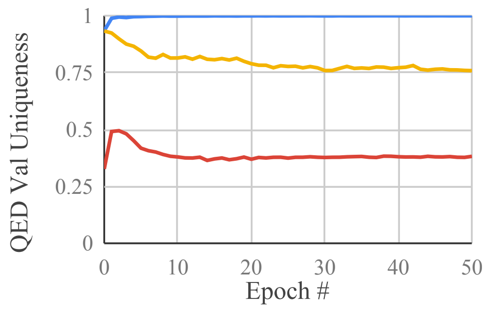
E.8. Unconditional Molecular Design Ablations

In Table 9 we present an ablation analysis for the unconditional setting, similar to that for the conditional setting in the main text. We also analyze an ablation VSeq(dupe), an ablation of our stochastic iterative target augmentation method applied to VSeq. It samples targets with replacement during augmentation, unlike our full method which deduplicates. As suggested by our theoretical remark on the difference between sampling with and without replacement in Appendix C, VSeq(dupe) underperforms VSeq+. As Figure 10 demonstrates, its diversity eventually decreases over time.

Interestingly, VSeq(train) achieves nearly the same uniqueness score as VSeq+, indicating that the additional training targets from our stochastic iterative augmentation method are responsible for most if not all the gains over the baseline. In particular, even our ablation model VSeq(train) significantly outperforms the REINVENT baseline, demonstrating that our model's advantage over RL is not limited to our prediction-time filtering procedure.

Model	Train+	Test+	QED Succ.	QED Uniq.	DRD2 Succ.	DRD2 Uniq.
VSeq	✗	✗	62.4	0.499	51.4	0.221
VSeq(test)	✗	✓	96.5	0.732	92.4	0.338
VSeq(train)	✓	✗	95.3	0.953	92.5	0.924
VSeq+	✓	✓	95.8	0.957	92.8	0.927
VSeq(dupe)	✓	✓	93.2	0.886	83.9	0.511

Table 9. Ablation analysis of filtering at training and test time for unconditional molecular generation. “Train+” indicates a model whose training process uses data augmentation according to our framework. “Test+” indicates a model that uses the external filter at prediction time to discard candidate outputs which fail to pass the filter.



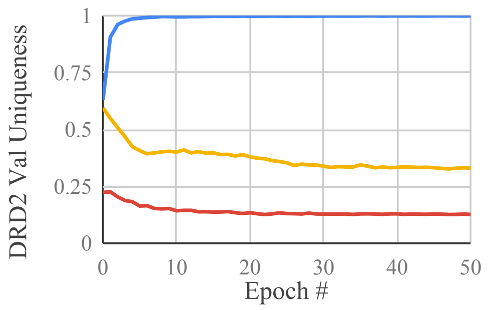


Figure 10. **Left**: Epoch number vs. uniqueness, evaluated with the Chemprop proxy predictor, for VSeq-based models on QED dataset. VSeq+, Vseq(dupe), and VSeq in blue, yellow, and red respectively. **Right**: Same plot for DRD2. Note that both VSeq(dupe) and VSeq+ are trained without iterative target augmentation for the initial epoch 0, and trained with augmentation thereafter.

Model	Train+	Test+	Top-1 Generalization
MLE*	✗	✗	70.91
*MLE(test)**	✗	✓	79.61
MLE(train)	✓	✗	77.92
MLE+	✓	✓	**85.02**

Table 10. Ablation analysis of filtering at training and test time for program synthesis. “Train+” indicates a model whose training process uses data augmentation according to our framework. “Test+” indicates a model that uses the external filter at prediction time. *Note that MLE and MLE(test) are based on an MLE checkpoint which underperforms the published result from Bunel et al. (2018) by 1 point, due to training for fewer epochs.

E.9. Program Synthesis Ablations

In Table 10 we provide the same ablation analysis that we provided in the main text for the conditional molecular design task, demonstrating that both training time iterative target augmentation as well as prediction time filtering are beneficial to model performance. We hypothesize that the effect of test-time filtering is relatively larger in program synthesis than in molecular design because checking correctness is easier in this domain. However, we note that even MLE(train), our model without prediction time filtering, outperforms the best RL method from Bunel et al. (2018).