Rethinking Molecular Design: Integrating Latent Variable and Auto-Regressive Models for Enhanced Goal Directed Generation

Heath Arthur-Louis [1] **Amina Mollaysa** [2 3] **Michael Krauthammer** [2 3]

Abstract

De novo molecule design has become a highly active research area, advanced significantly through the use of state-of-the-art generative models. Despite these advances, several fundamental questions remain unanswered as the field increasingly focuses on more complex generative models and sophisticated molecular representations as an answer to the challenges of drug design. In this paper, we return to the simplest representation of molecules, and investigate overlooked limitations of classical generative approaches, particularly Variational Autoencoders (VAEs) and auto-regressive models. We propose a hybrid model in the form of a novel regularizer that leverages the strengths of both to improve validity, conditional generation, and style transfer of molecular sequences. Additionally, we provide an in depth discussion of overlooked assumptions of these models' behaviour.

1. Introduction

Molecule discovery tasks can be divided into two types. Global optimization aims to find molecules with a specific target property. Local optimization starts with an initial molecule and searches for similar molecules that have the desired property without deviating too much from the original. Past advances in molecular design have primarily leveraged either latent variable models or auto-regressive models, each with distinct advantages and shortcomings. While auto-regressive models excel in capturing conditional distributions and generating valid molecules with desired properties, they tend to overfit the training data, leading to a limited number of novel molecules when trained on small datasets (Mollaysa et al., 2020). Additionally, they lack the flexibility of latent variable models, which facilitate local optimization and complex transformations such as style transfer. On the other hand, latent variable models often struggle to generate valid molecules unless more restrictive and complex representations are applied, due to the challenges of generating valid molecules from a compressed latent space (Gómez-Bombarelli et al., 2018; Kusner et al., 2017; Dai et al., 2018; Jin et al., 2018).

In this work, instead of relying on increasingly complex generative models and sophisticated representations as solutions for every challenge, we return to the simple string based representation; SMILES. By combining the strength of auto-regressive models and latent variable models, we aim to resolve the performance gap in conditional generation and enable effective style transfer without sacrificing the generative model's validity. We introduce a theoretical framework that combines the robust conditional distribution modeling capability of auto-regressive models with the flexible representational power of latent variable models. Our approach involves a dual-training mechanism where the auto-regressive model informs the training of the latent variable model, ensuring that the generative process respects both the desired property of the generated samples as captured by the auto-regressive model and the local structural coherence enforced by the latent variable.

The core of our methodology is a hybrid generative model that employs a conditional VAE architecture with an embedded auto-regressive model to guide the decoder (generative distribution). The model is trained using a novel objective function that incorporates a regularizer to ensure the generation of valid molecules with desired properties. The regularizer is implemented in two forms: a calibration regularizer, which is a Kullback-Leibler divergence term between the marginalized generative distribution and a target distribution defined by the auto-regressive model; and a reward-based regularizer, which rewards the generative model for producing molecules with high probability under the target distribution. Each regularizer guides the generative model, enhancing its ability to produce valid and conditionally appropriate molecules. Our contribution includes:

[1]ETH, Zurich, Switzerland [2]Department of Quantitative Biomedicine, University of Zurich, Switzerland [3]ETH AI Center. Correspondence to: Heath Arthur-Louis <contact@louis-heath.me>, Amina Mollaysa <maolaaisha.aminanmu@uzh.ch>.

Proceedings of the ICML 2024 Workshop on Accessible and Efficient Foundation Models for Biological Discovery, Vienna, Austria. 2024. Copyright 2024 by the author(s).

- We present a comprehensive analysis of the limitations associated with latent generative models in molecular design, particularly focusing on their ability to perform conditional generation and style transfer.
- We propose a hybrid generative framework that effectively combines the strengths of auto-regressive and latent variable models, offering a balanced approach to molecular design.

2. Methods

Suppose we are given a training set of pairs $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}; i = 1, \ldots, N$, where $\mathbf{x}_i$ corresponds to a molecule and $\mathbf{y}_i$ represents its corresponding molecular properties. Goal directed molecule design can be formulated as learning the conditional distribution $p(\mathbf{x}|\mathbf{y})$ which allow us to set a value for $\mathbf{y}$ and generate a diverse set of molecules exhibiting the specified properties. Additionally, this task can be framed as local optimization, where we start with a prototype molecule and modify it to achieve the intended property without deviating (structurally) too far from the original molecule. We refer to this as style transfer over molecules, where we aim to modify $\mathbf{x}$ to $\mathbf{x}'$ such that $\mathbf{x}'$ exhibits property $\mathbf{y}'$. This can be achieved with a latent variable model $p(\mathbf{x}|\mathbf{y}, \mathbf{z})$ that allows us to control latent factors corresponding to certain molecular structures. The style transfer task can then be defined as follows:

$$p_\theta(\mathbf{x}'|\mathbf{y}', \mathbf{x}) = \int p_\theta(\mathbf{x}'|\mathbf{y}', \mathbf{z}) q_\phi(\mathbf{z}|\mathbf{x}) \mathrm{d}\mathbf{z}. \tag{1}$$

Concretely, this involves fitting a joint generative model of the form $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) p(\mathbf{y}) p(\mathbf{z})$ and a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to infer the latent variable $\mathbf{z}$. This model can be trained with the standard ELBO objective function:

$$\mathcal{L}_{ELBO}(\theta, \phi) = \sum_{i=1}^{N} \Big\{ \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log p_\theta(\mathbf{x}_i|\mathbf{y}_i, \mathbf{z}_i)] - D_{KL}(q_\phi(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) \Big\} \tag{2}$$

which corresponds to learning a supervised (conditional) VAE (Kingma et al., 2014).

Such models are often trained to generate molecules in one shot (Kusner et al., 2017; Dai et al., 2018), meaning the decoder assumes $p(\mathbf{x}|\mathbf{y}, \mathbf{z}) = \prod_{t=0}^{T} p(x_t|\mathbf{y}, \mathbf{z})$. This was set to avoid using a strong decoder (auto-regressive network trained with teacher forcing) that assumes $p(\mathbf{x}|\mathbf{y}, \mathbf{z}) = \prod_{t=0}^{T} p(x_t|\mathbf{y}, \mathbf{z}, \mathbf{x}_{<\mathbf{t}})$ as it leads to the generative model $p(\mathbf{x}|\mathbf{z}, \mathbf{y})$ ignoring the latent variable $\mathbf{z}$ and defaulting to $p(\mathbf{x}|\mathbf{y})$ (Gómez-Bombarelli et al., 2018). As a result, often the generative model $p(\mathbf{x}|\mathbf{z}, \mathbf{y})$ is trained in a non-auto regressive fashion, which makes the task very challenging and often under performs compared to auto-regressive models that are designed to model $p(\mathbf{x_t}|\mathbf{y}, \mathbf{x}_{<\mathbf{t}})]$ which is trained to predict the next token.

Additionally, the task of conditional generation becomes challenging when using standard conditional VAEs to model the generative distribution $p(\mathbf{x}|\mathbf{z}, \mathbf{y})$. The standard objective (Eq.2) focuses only on reconstruction loss and a KL term, and has the potential to lead to scenarios where $p(\mathbf{x}|\mathbf{z}, \mathbf{y})$ may disregard $\mathbf{y}$, relying solely on $\mathbf{z}$ for molecule reconstruction. This often results in poor conditional generation performance, as the model effectively converges to $p(\mathbf{x}|\mathbf{z})$ and loses control over the properties of the generated molecules. Furthermore, the style transfer task also tends to perform poorly due to the lack of supervision during the training. The models are not exposed to examples where a latent $\mathbf{z}$ from a prototype molecule combines with a new property $\mathbf{y}$ to predict what a molecule with structure $\mathbf{z}$ and property $\mathbf{y}$ would look like. This gap in training results in the model's inability to effectively handle style transfer.

Our observations indicate that an auto-regressive model, which directly learns the conditional distribution $p(\mathbf{x}|\mathbf{y})$, demonstrates superior performance in conditional generation. This enhancement is attributed to its effective learning of the conditional distribution $\tilde{p}(\mathbf{x}|\mathbf{y}) \approx p(\mathbf{x}|\mathbf{y})$. By using this learned distribution as an approximation to the true conditional distribution, we can effectively guide our VAE model. This guidance enables the VAE to generate molecules that have a high probability under the learned distribution $\tilde{p}(\mathbf{x}|\mathbf{y})$.

2.1. Proposed Regularizer

We propose the use of a pre-trained auto-regressive model $\tilde{p}(\mathbf{x}|\mathbf{y})$ as a surrogate for the true conditional distribution. This surrogate distribution can then be employed to direct the generative network, encouraging the generative model to generate molecules that have a high probability under the surrogate model. To achieve this, we augment the supervised VAE objective with one of the regularizer terms outlined below.

Calibration Regularizer To achieve this constraint, we introduce a Kullback-Leibler (KL) divergence term that aligns the marginalized generative distribution $p_\theta(\mathbf{x}|\mathbf{y}_i)$ of a conditional VAE with the surrogate distribution $\tilde{p}(\mathbf{x}|\mathbf{y}_i)$.

$$\mathcal{R}_1(\theta) = \min_\theta D_{KL}(p_\theta(\mathbf{x}|\mathbf{y}_i)||\tilde{p}(\mathbf{x}|\mathbf{y}_i)) \tag{3}$$

where the marginalized conditional distribution is given by:

$$p_\theta(\mathbf{x}|\mathbf{y}_i) = \int p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}) p(\mathbf{z})\, d\mathbf{z} = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z})$$

Given we do not have direct access to the marginalized distribution $p_\theta(\mathbf{x}|\mathbf{y}_i)$, we estimate it using the Monte

Carlo integration approach: $\hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) \approx \hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) = \frac{1}{N}\sum_{n=1}^{N} p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}_n)$ and then apply the Kullback-Leibler (KL) divergence on the approximated distribution:

$$D_{KL}(\hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) \parallel \tilde{p}(\mathbf{x}|\mathbf{y}_i)) = \sum_{\mathbf{x}} \hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) \log \frac{\hat{p}_\theta(\mathbf{x}|\mathbf{y}_i)}{\tilde{p}(\mathbf{x}|\mathbf{y}_i)}$$

Reward-Based Regularizer As discussed in Section 3.1, the calibration regualrizer encountered issues when paired with the standard character VAE decoder structure. To attempt to address this we formulated an alternative regularizer based on the following reward function:

$$\begin{aligned}\mathcal{R}_2(\theta) &= \max_\theta \sum_{i=1}^{N} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z})} \log \tilde{p}(\mathbf{x}|\mathbf{y}_i) \\ &= \max_\theta \sum_{i=1}^{N}\sum_{j=1}^{M} \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}_j)} \log \tilde{p}(\mathbf{x}|\mathbf{y}_i) \quad (4)\end{aligned}$$

This regularizer can be seen as expected reward objective where $\log \tilde{p}(\mathbf{x}|\mathbf{y}_i)$ can be seen as reward function and $p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}_j)$ is the policy of the action, where we want to learn a policy model that maximizes our reward. The regularzier can be optimized using the score function gradient estimate, i.e., $\mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}_j)} [\log \tilde{p}(\mathbf{x}|\mathbf{y}_i) \nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}_j)]$ However, the score function gradient estimator is known to be noisy and unstable unless we use large samples and control variate techniques. To avoid excessive computational complexity, we propose an alternative approach to achieve the same regularization (Mollaysa et al., 2020). We reformulate our regularizer as $\max_\theta \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}_j)} \tilde{p}(\mathbf{x}|\mathbf{y}_i)$, using $\tilde{p}(\mathbf{x}|\mathbf{y}_i)$ as a positive reward function. This allows us to express the regularizer as:

$$\begin{aligned}\max_\theta \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}_j)} \tilde{p}(\mathbf{x}|\mathbf{y}_i) &= \max_\theta \int p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}_j) \tilde{p}(\mathbf{x}|\mathbf{y}_i) d\mathbf{x} \\ &= \max_\theta \mathbb{E}_{\mathbf{x}\sim \tilde{p}(\mathbf{x}|\mathbf{y}_i)} p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}_j) \quad (5)\end{aligned}$$

2.2. Incorporating the Regularizer

We use the standard conditional VAE (Eq.2) as our baseline and incorporated our regularizer to see if the regularizer would improve the generation, conditional generation as well as style transfer peformance of the basline model. We first train an LSTM-based model to learn $\hat{p}(\mathbf{x}|\mathbf{y})$. This pre-trained model is then served as the surrogate distribution used in our regularizer. The final training objective of our model consists of the standard ELBO (Eq. 2) and an additional regularizer, $\mathcal{L}(\theta, \phi) = \mathcal{L}_{ELBO}(\theta, \phi) + \lambda\mathcal{R}(\theta)$. where $\mathcal{R}(\theta)$ refers to either the calibration regularizer or the reward-based regularizer. Depending on which regularizer is used, it is either minimized together with the negative ELBO or its negative form is used when applying the reward-based regularizer. Note that during training of our model, the regularizer updates only the parameters of the generative model (decoder). The parameters of the surrogate distribution $\hat{p}(\mathbf{x}|\mathbf{y})$ remain fixed.

3. Experiments

Dataset We conducted a preliminary investigation of the regularizers' behaviour on the QM9 dataset (Blum & Reymond, 2009). We use the octanol-water partition coefficient (LogP) as the conditioning property. Preprocessing details and data analysis are available in Appendix D. Additionally, we tested our approach on the ZINC250k dataset, results are presented in Appendix B.

Models We implemented a baseline conditional VAE along with three versions of the regularizer: the Calibration regularizer, which uses the KL divergence and is referred to as cVAE-KLD, and the two implementations of the reward-based regularizer referred to as cVAE-Pol1 and cVAE-Pol2. For each of these, we trained three versions with different decoding strategies (details provided in Appendix E.1): a one-shot decoder, where the sampling of the current token does not depend on the previous token (denoted as cVAE); an auto-regressive decoder, which uses an auto-regressive structure and explicitly samples next tokens at each generation step trained without teacher forcing (EXP-cVAE); and an auto-regressive decoder that employs teacher forcing during training to provide input tokens for the sequence (EXP-cVAE-TF). We use a pre-trained LSTM model as the surrogate. Model architecture and training parameters are given in Appendix E.

Evaluation Generated sample quality was evaluated using validity, novelty and uniqueness, while conditional performance was assessed with the mean error (MAE) between target and generated properties. Style transfer performance was measured by the proportion of valid style transferred molecules (style transfer validity), the proportion of transfers where the source molecule was reproduced (PCT Fail), Tanimoto similarity of Morgan fingerprints of source and target molecule (Tan) compared with two baselines as lower bounds (Tan-B1 and Tan-B2), and the MAE between target and generated properties. Examples of successful style transfer are provided in Appendix C. Details on metric computation and sampling are provided in Appendix F. The code is available on GitHub.

3.1. Experimental Results

One-Shot Decoder Models The experimental results for each regularizer type (KLD, Pol1, Pol2) are presented in Tables 1-6. As shown in Tables 1 and 2, models trained with the one-shot version of the decoder perform poorly in generation and conditional generation compared to the baseline cVAE. Both the calibration regularizer (cVAE-KLD) and the reward-based regularizer (cVAE-Pol2) result in reduced va-

lidity. Additionally, cVAE-Pol1 suffers from mode collapse. We attribute the failure of the regularizers to the misalignment between the one-shot decoder's molecule generation and the auto-regressive surrogate distribution.

Despite this poor generation performance, the one-shot versions of the models performed better compared to models trained with auto-regressive decoders at the style transfer task (see Table 2), implying latent points and their neighbourhoods belonging to the posterior distribution learned by $q_\phi(\mathbf{z}|\mathbf{x})$ are meaningful under the decoder.

Table 1. Conditional generation performance of models trained with one-shot decoders.

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
LSTM	-	0.9700	1.0000	0.3024	0.3211
cVAE	0.8872	0.2210	1.0000	0.8959	0.7976
cVAE-KLD	0.8796	0.0110	1.0000	0.9090	0.7919
cVAE-Pol1	0.9436	1.0000	0.0010	0.0000	1.0121
cVAE-Pol2	0.8769	0.0050	1.0000	1.0000	1.4675
(Gómez-Bombarelli et al., 2018)	0.0361	0.1030	-	0.9000	-

Table 2. Style transfer performance of cVAE with one-shot decoders.

Model	PCT Valid ↑	PCT Fail ↓	MAE ↓	Tan ↑	Tan-B1	Tan-B2
cVAE	0.9260	0.7482	0.8636	0.4194	0.0622	0.0655
cVAE-KLD	0.9072	0.7579	0.8395	0.3894	0.0490	0.0677
cVAE-Pol1	0.9468	0.8906	0.9909	0.4451	0.0296	0.0652
cVAE-Pol2	0.9148	0.7420	0.8497	0.4111	0.0536	0.0659

Auto-Regressive Decoder Models To ensure alignment with the surrogate distribution used in the regularizer, which is trained auto-regressively, we also investigated the auto-regressive decoder models. However, to prevent latent representation degradation due to the strong decoder, we abstained from using teacher forcing. Instead, at each step, we sampled from the model's own prediction, forcing the decoder to rely on the latent representation during generation. The results are presented in Table 3 and 4. Tables demonstrates promising results for the calibration regularizer (EXP-cVAE-KLD) and reward-based regularzier (EXP-cVAE-Pol2). Compared to the baseline, both improve the validity of the generated molecules and have significant improvement in conditonal generation as well as the style transfer task. However, the EXP-cVAE-Pol1 model continues to fail.

Table 3. Conditional generation performance of models trained with auto-regressive decoders without teacher forcing.

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
EXP-cVAE	0.9509	0.2860	1.0000	0.8287	0.7058
EXP-cVAE-KLD	0.6783	0.7460	1.0000	0.5161	0.2831
EXP-cVAE-Pol1	0.9363	0.0010	1.0000	1.0000	0.1987
EXP-cVAE-Pol2	0.6711	0.8250	0.9976	0.4642	0.2300

Table 4. Style transfer performance of cVAE with auto-regressive decoder trained without teacher forcing.

Model	PCT Valid. ↑	PCT Fail ↓	MAE ↓	Tan ↑	Tan-B1	Tan-B2
EXP-cVAE	0.8968	0.7243	0.7514	0.3711	0.0653	0.0646
EXP-cVAE-KLD	0.8260	0.4459	0.5377	0.3507	0.0679	0.0667
EXP-cVAE-Pol1	0.8888	0.7115	1.5314	0.3206	0.0209	0.0666
EXP-cVAE-Pol2	0.8404	0.4131	0.5493	0.3481	0.0653	0.0662

In our final exploration, we extended the decoder to full auto-regressive setting trained with teacher forcing (Tables 5, 6). As expected, the introduction of teacher forcing significantly improves the decoder's ability to generate valid molecules. Consistent as previous results, the two regularizer (EXP-cVAE-TF-KLD and EXP-cVAE-TF-Pol2) greatly improves both the proportion of valid molecules and the conditional performance of the model. In terms of style transfer performance we see a similar trend to the non teacher forcing case, with the regularized version of the model having a much lower proportion of failed style transfer attempts. To illustrate this we have included a plot of the style transfer performance of both models (EXP-cVAE-TF-KLD and EXP-cVAE-TF) in Appendix C.

Note that the results of the auto-regressive decoder (trained with teacher forcing) contradict the common assumption that a "strong decoder" could degrade the latent space. Therefore, we conducted a more in-depth analysis, which we provide in Appendix Section A.4. We conclude when trained in an auto-regressive fashion with teacher forcing, the latent representation is utilized until a certain length of the sequence, beyond which the decoder does not rely on the latent representation to generate the rest of the sequence.

Table 5. Conditional generation performance of models trained with auto-regressive decoders with teacher forcing.

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
EXP-cVAE-TF	0.9481	0.8340	1.0000	0.7014	0.3636
EXP-cVAE-TF-KLD	0.6857	0.9580	0.9958	0.3862	0.1292
EXP-cVAE-TF-Pol1	0.9575	0.0010	1.0000	0.0000	0.0816
EXP-cVAE-TF-Pol2	0.6799	0.9710	0.9938	0.3326	0.1259
Lim et. all.-TF	0.4381	0.5700	0.9863	0.9520	0.4510

Table 6. Style transfer performance of cVAE with auto-regressive decoder trained with teacher forcing.

Model	PCT Valid ↑	PCT Fail ↓	MAE ↓	Tan ↑	Tan B1	Tan B2
EXP-cVAE-TF	0.9176	0.3391	0.3955	0.3071	0.0667	0.0668
EXP-cVAE-TF-KLD	0.9668	0.0852	0.1493	0.2328	0.0651	0.0658
EXP-cVAE-TF-Pol1	0.4312	0.6790	0.2935	0.3614	0.0532	0.0652
EXP-cVAE-TF-Pol2	0.9656	0.0849	0.1486	0.2280	0.0652	0.0651
Lim et al.-TF	-	0.0210	0.3439	0.1424	0.08160	0.11450

4. Conclusions

Our framework addresses key limitations in classic generative methods, particularly enhancing conditional generation and style transfer. Novel regularizers, such as the calibration and reward-based regularizers, guide the generative process, improving the fidelity and validity of generated molecules. This work is intended to return to fundamental models based on simple SMILES representations to investigate commonly accepted hypotheses and provide in-depth analysis. Experimental results demonstrated the potential of the proposed regularizers, however, further investigation and testing are still needed. Our work also provided insights about the models and assumption used in molecular design, paving the way for future research in more complex settings.

References

Aminanmu, M. *Structural and Functional Regularization of Deep Learning Models*. PhD thesis, Université de Genève, 2021.

Bagal, V., Aggarwal, R., Vinod, P., and Priyakumar, U. D. MolGPT: molecular generation using a transformer-decoder model. *Journal of Chemical Information and Modeling*, 62(9):2064–2076, 2021.

Bayer, J. and Osendorfer, C. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Bird, A. and Williams, C. K. Customizing sequence generation with multi-task dynamical systems. *arXiv preprint arXiv:1910.05026*, 2019.

Bjerrum, E. J. and Threlfall, R. Molecular generation with recurrent neural networks (RNNs). *ArXiv preprint ArXiv: 1705.04612*, 2017.

Blum, L. C. and Reymond, J.-L. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

Fabius, O. and van Amersfoort, J. R. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

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*, 4(2):268–276, 2018.

Grechishnikova, D. Transformer neural network for protein-specific de novo drug generation as a machine translation problem. *Scientific Reports*, 11, 2019.

Irwin, J. J. and Shoichet, B. K. ZINC - a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1): 177–182, 2005.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

Kotsias, P.-C., Arús-Pous, J., Chen, H., Engkvist, O., Tyrchan, C., and Bjerrum, E. J. Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks. *Nature Machine Intelligence*, 2:254 – 265, 2020.

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

Lim, J., Ryu, S., Kim, J. W., and Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of cheminformatics*, 10:1–9, 2018a.

Lim, J., Ryu, S., Kim, J. W., and Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of cheminformatics*, 10(1):1–9, 2018b.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems 31*, pp. 7806–7815. 2018.

Mokaya, M., Imrie, F., van Hoorn, W. P., Kalisz, A., Bradley, A. R., and Deane, C. M. Testing the limits of smiles-based de novo molecular generation with curriculum and deep reinforcement learning. *Nature Machine Intelligence*, 5: 386–394, 2022.

Mollaysa, A., Paige, B., and Kalousis, A. Conditional generation of molecules from disentangled representations.

Mollaysa, A., Paige, B., and Kalousis, A. Goal-directed generation of discrete structures with conditional generative models. *Advances in Neural Information Processing Systems*, 33, 2020.

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

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

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. Generating focused molecule libraries for drug discovery

with recurrent neural networks. *ACS central science*, 4 (1):120–131, 2017.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018.

Thiede, L. A., Krenn, M., Nigam, A., and Aspuru-Guzik, A. Curiosity in exploring chemical spaces: intrinsic rewards for molecular reinforcement learning. *Machine Learning: Science and Technology*, 3(3):035008, 2022.

Wang, J., Hsieh, C.-Y., Wang, M., Wang, X., Wu, Z., Jiang, D., Liao, B., Zhang, X., Yang, B., He, Q., Cao, D., Chen, X., and Hou, T. Multi-constraint molecular generation based on conditional transformer, knowledge distillation and reinforcement learning. *Nature Machine Intelligence*, 3:914 – 922, 2021.

Wang, M., Lu, Z., Zhou, J., and Liu, Q. Deep neural machine translation with linear associative unit. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pp. 136–145. Association for Computational Linguistics, 2017.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems 31*, pp. 6412–6422. 2018.

A. Literature Review

A.1. Auto Regressive Models Trained on SMILES

Early auto-regressive approaches to molecule generation (Segler et al., 2017) (Bjerrum & Threlfall, 2017) showed that auto-regressive model can successfully learn a distribution of SMILES strings and generate realistic candidates with high accuracy. Furthermore, Segler et al. (2017) proposed conditional generation be achieved by filtering the generated molecules for candidates with desirable properties or by fine tuning the model on a curated dataset of desirable molecules. Methods have also been developed to directly control the conditioning and avoid the need for further fine-tuning. For example, concatenating conditional information to inputs at the first or every generation step and encoding conditional information into the model's initial hidden state (Kotsias et al., 2020).

A parallel approach to guiding auto-regressive models for conditional generation uses policy optimization and a set of objective functions to fine-tune auto retrogressive models. The seminal examples in this direction are REINVENT (Olivecrona et al., 2017) and ReLeaSe (Popova et al., 2018). More recent work has incorporated various other innovations from the field of reinforcement learning such as (Thiede et al., 2022) and curriculum learning (Mokaya et al., 2022).

Following their success in the field of natural language processing, the majority of recent publications have adopted various transformer based architectures. For example, Wang et al. (2017) and Grechishnikova (2019) propose transformer based approaches to structure based drug design. Another notable recent example, Wang et al. (2021) propose a multi-step pipeline where knowledge is distilled from a transformer model to an RNN-based model which is then further fine-tuned using reinforcement learning.

Although these have been found to outperform older architectures such as RNN and LSTM, the difference is minimal in simpler settings (Bagal et al., 2021). For this reason and to maintain a similar architecture between the VAE decoder and auto-regressive model and prior VAE based models, we have opted to employ a conditional LSTM model in this study.

A.2. Latent Variable Models Trained on SMILES

The seminal implementation of a latent variable model for SMILES (Gómez-Bombarelli et al., 2018) is based on a conditional character VAE that encodes SMILES as continuous latent vectors. It also uses a network to predict molecular properties from latent vectors, allowing for efficient property optimization. Lim et al. (2018a) propose an approach based on a conditional VAE for simultaneous control of multiple properties. Although successful, the SMILES generated by the model are prone to syntactic and semantic errors. Following works aimed to address this by operating on grammar rules for generating SMILES (Kusner et al., 2017), and later on attribute grammars (Dai et al., 2018; Mollaysa et al.) (to address errors that are still possible when using a regular CFG). These models were able to generate a much higher proportion of valid SMILES at the cost of computational and implementation complexity.

Others works such as Simonovsky & Komodakis (2018), Jin et al. (2018), You et al. (2018) and Liu et al. (2018) explored models that operate on graph representations, largely improving the ability to generate valid molecules. A notable example is JT-VAE (Jin et al., 2018) witch is able to guarantee validity of generated molecules by constructing molecules from a library of fragments.

A.3. Auto-Regressive Models with Latent Variables

There have been several attempts to introduce latent variables into RNNs such VRNN (Chung et al., 2015), and others (Bayer & Osendorfer, 2014; Boulanger-Lewandowski et al., 2012; Fabius & van Amersfoort, 2014). However, the goal of such latent variables is to introduce more stochasticity into the model as a standard RNN has limited power to model the variability observed in highly-structured data. Integrating latent variables into the hidden state of the RNN aims to be able to model more variability that is observed in the structured data. When used to model the conditional distribution $p(\mathbf{x}|\mathbf{y})$, while such models help to learn a distribution with more diverse samples compared to standard RNN models, they do not provide sufficient control over the style of the generated samples. Recently, an approach similar to VRNN, Bird & Williams (2019) also introduces a latent variable $\mathbf{z}$ into RNNs, in addition to modelling more variability of sequences, this paper aims to capture sequence-specific features through latent the variable $\mathbf{z}$ and use it to perform style transfer. The latent variable $\mathbf{z}$ is inferred from input output sequence pairs and the parameters of the RNN are made dependent on $\mathbf{z}$. This enables each sequence to be modelled as a single dynamical system and capture instance-specific variation through $\mathbf{z}$ which later helps us to perform style transfer over sequences. Due to auto-regressive models' strong conditional generation performance, it

Table 7. Reconstruction rates with different conditions.

Condition	Reconstruction
Standard (using $\mathbf{z}$ throughout)	0.4351
Using $\mathbf{z}$ up to 15th step	0.4312
Using $\mathbf{z}$ up to 10th step	0.3922
Using $\mathbf{z}$ up to 1st step	0.0025
Not using $\mathbf{z}$ (prior from step zero)	0.0000

would be interesting to perform style transfer with auto-regressive models in an unsupervised fashion without the need of constructing a paired dataset.

A.4. Strong Decoder

It has been assumed or stated by (Gómez-Bombarelli et al., 2018) that when trained with teacher forcing, the auto-regressive decoder tends to ignore the encoded $\mathbf{z}$ and recover the next token purely from the current token. However, this assumption was overlooked by (Lim et al., 2018b), who proceeded with a similar type of model featuring a strong decoder (an RNN trained with teacher forcing). Mathematically, it is reasonable to assume that if one can learn $p(\mathbf{x}|\mathbf{y})$ using an auto-regressive model trained to predict the next token (Segler et al., 2017), then when training a generative model $p(\mathbf{x}|\mathbf{z},\mathbf{y})$ through a VAE using the same model as a decoder, the generative model could easily converge to the auto-regressive model $p(\mathbf{x}|\mathbf{y})$ by ignoring $\mathbf{z}$. This is because it is easy to recover the next token $p(x_t|\mathbf{y}, x_1, \ldots, x_{t-1}, \mathbf{z})$ without relying on $\mathbf{z}$ at all. This assumption has been generally accepted in subsequent research papers, which have trained $p(\mathbf{x}|\mathbf{z},\mathbf{y})$ without the auto-regressive component when using latent variable models to avoid degenerated latent representations.

To investigate the impact of the latent variable $\mathbf{z}$, we took the model from (Lim et al., 2018b) and tested if their model leads to degraded latent representation. The first reliable measure would be reconstruction, but they did not report the reconstruction performance of their model. Since the model training is very slow, we retrained their model on the QM9 dataset as a proof of concept.

The model performance is reported in Tables 5 and 6. It shows that we have a reconstruction rate of 0.43. Note that $\mathbf{y}$ also carries some information about $\mathbf{x}$. Therefore, to test further, we tried the reconstruction with latent $\mathbf{z}$ that is sampled from the prior instead of sampled from the posterior. This resulted in a 0% reconstruction rate, which further validates that the latent variable $\mathbf{z}$ does carry useful information about the encoded $\mathbf{x}$.

After investigating whether the learned latent variable $\mathbf{z}$ carries meaningful information, we discovered that it does, but only up to a certain step in the auto-regressive generation process. For instance, with the QM9 dataset, where the maximum sequence length T is under 40, the decoder operates in an auto-regressive setting. During reconstruction, while decoding progressively, we observed the following results presented in Table 7:

This indicates that $\mathbf{z}$ carries information that is utilized only until the 16th step, after which it becomes ineffective. This behavior arises from the auto-regressive training with teacher forcing. Initially, when generating the next token conditioned on $\mathbf{y}$ and the current token, there are many possible characters that could be the next token, each leading to a SMILES string with the property $\mathbf{y}$. Therefore, it is crucial for $\mathbf{z}$ to guide the decoding process to ensure the generation of the specific $\mathbf{x}$ that was used to encode $\mathbf{z}$. This makes $\mathbf{z}$ essential for generating the next token at the beginning of the sequence. However, as the sequence progresses, the number of possible next tokens significantly reduces, making it clear what the next token should be based on the current token. Consequently, $\mathbf{z}$ becomes less useful, leading the model to utilize $\mathbf{z}$ primarily for encoding the initial structure of the original SMILES string, but not for the entire sequence.

A.5. Reward-Based Regularizer Behaviour

The initial version of the reward-based regularizer (Pol1) failed to produce a high proportion of valid and unique molecules, regardless of decoder stricture. This may be due to high variance during training as a result of relying on the score function gradient estimate. We attempted to fix this by introducing the regularizer at later points in the training process (where the decoder has learned to generate better molecules) and by testing various decoder scheduling strategies with little success.

A.6. Analysis

Can we actually disentangle the latent variable $\mathbf{z}$ and property $\mathbf{y}$? First of all, unlike in the space of image where we can actually annotate the variations in the data, molecular space is much more complicated. When we apply vanilla VAE to encode molecules to a latent space, we can assume that $\mathbf{z}$ encodes the molecular structure, and previous works have shown that points that are close in the latent space result in molecules that are structurally similar. However, once we assume a molecule is generated from a property variable y and some other latent factor $\mathbf{z}$ where $\mathbf{y}$ and $\mathbf{z}$ are independent under the prior, $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x}|\mathbf{y}, \mathbf{z})p(\mathbf{z})p(\mathbf{y})$, then we can not really say what $\mathbf{z}$ encodes. Therefore, it is not really clear what $\mathbf{z}$ encodes in this setting. Even though we still assume $\mathbf{z}$ encodes some structural features of the molecules, for instance, some functional groups that are not dependent on the property $\mathbf{y}$, it is hard to clarify what aspect of the molecule that $\mathbf{z}$ learns. However, it is clear is that $\mathbf{z}$ does not fully specify the molecule's structure since the property of the molecule is fully determined by the structure of the molecule, and we assume the learned $\mathbf{z}$ is disentangled from $\mathbf{y}$. We can not modify the property of a molecule without modifying its structure. However, note that $\mathbf{z}$ is a learned factor, assuming that it is independent of the $\mathbf{y}$. Therefore, $\mathbf{z}$ should not learn some variations in the molecule data that is independent of the conditioning properties. However, in reality, for a given molecule, it is not clear that if we can always modify its property to the value we want and stay close to the original molecules in terms of structure (Aminanmu, 2021).

B. Additional Experiments With the ZINC250k Dataset

Since submitting this work for review, we have run additional experiments on the ZINC250k subset of the "ZINC is not commercial" dataset (Irwin & Shoichet, 2005). The ZINC250k dataset contains more SMILES that are individually longer and more complex. Additionally all compounds are commercially available and highly drug-like. This makes ZINC a more realistic setting than the enumerated molecular structures in QM9.

ZINC SMILES were pre-processed with the same numerical encoding scheme used for QM9, with sequences padded to a final length of 120. The baseline upper-bound MAE between random pairs of test properties is 1.6067. Summary statistics of the sequences and properties are presented in Figure 1 and the distribution of SMILES characters in Table 8.

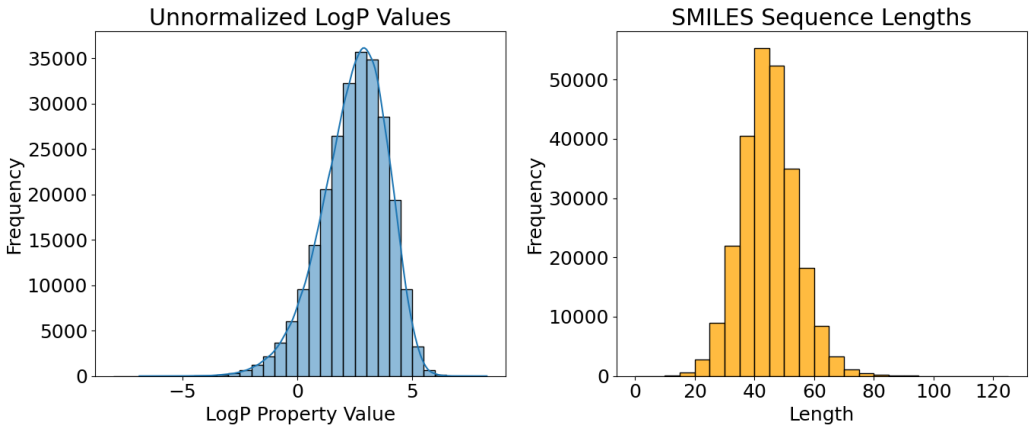

Figure 1. Summary statistics for ZINC250k dataset.

Table 8. SMILES symbol frequencies in the ZINC250k dataset.

Char.	Freq.	Norm. Freq.	Char.	Freq.	Norm. Freq.	Char.	Freq.	Norm. Freq.
c	2208591	0.20	]	344325	0.03	#	15587	0.00
C	2089152	0.19	@	335552	0.03	r	12722	0.00
(	963303	0.09	H	290126	0.03	B	12722	0.00
)	963303	0.09	F	79430	0.01	\	5839	0.00
1	724766	0.07	+	76813	0.01	5	2592	0.00
O	545244	0.05	S	63791	0.01	I	888	0.00
2	471857	0.04	-	63474	0.01	P	127	0.00
N	437084	0.04	l	42872	0.00	6	88	0.00
=	395693	0.04	s	39088	0.00	7	8	0.00
n	268119	0.02	o	30987	0.00	8	2	0.00
3	166892	0.02	4	30588	0.00			
[	344325	0.03	/	28557	0.00			

B.1. ZINC250k Results and Discussion

We retrained base models with each decoder type and all working configurations of the regularizers. The results for these are presented in Tables 10, 11. During training we used a surrogate model with the architecture described in Appendix Section E whose performance is presented in Table 9. We note that the novelty issue observed with the surrogate trained on QM9 is not present in ZINC250k. As predicted, the introduction of either regularizer (Table 10) does not decrease novelty. Additionally, the one-shot decoder model struggled to generate a non-trivial proportion of valid SMILES, in line with the experience of (Gómez-Bombarelli et al., 2018).

We see that our observations from the QM9 dataset carry over to the ZINC scenario. Conditional generation performance (Table 10) is improved both in terms of validity and property error. These changes are particularly dramatic in the case of

Table 9. Performance of surrogate LSTM model for ZINC250k.

Model	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
LSTM	0.9333	1.0000	0.9964	0.1714

the explicit decoder, where the model performs extremely poorly in both cases. The effect on style transfer performance (Table 11) of models' trained with the regularizer is also similar. Introduction of either form of the regularizer improves the target-generation MAE and decreases the proportion of style transfer failures. Additionally, it slightly decreases the source-generated structural similarity in the explicit case. As with QM9, the Pol2 regularizer appears to have a stronger effect on both tasks, with the exception of the proportion of style transfer failures which is surprisingly higher.

Although the direction of the changes introduced by the regularizers is the same, the absolute performance levels of the models lead to a different trade-off. One difference from the QM9 case, is that the teacher forcing decoder is less effective by itself, only achieving a validity of 0.568 compared to 0.938 when trained with the Pol2 regularizer. Additionally, in the QM9 case, we see that without teacher forcing, the decoder puts significant emphasis on $\mathbf{z}$, and therefore achieves a poor MAE even when the regularizer is used. Overall, these differences make the regularized teacher forcing model appear attractive relative to the explicit one.

Table 10. Conditional generation performance of ZINC250k models.

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
cVAE	0.1852	0.0010	1.0000	1.0000	1.0228
EXP-cVAE	0.4301	0.0020	1.0000	1.0000	1.5033
EXP-cVAE-KLD	0.1716	0.8550	1.0000	1.0000	0.3690
EXP-cVAE-Pol2	0.4321	0.9360	1.0000	1.0000	0.2065
EXP-cVAE-TF	0.5832	0.5680	1.0000	1.0000	0.6600
EXP-cVAE-TF-KLD	0.3439	0.9110	1.0000	0.9989	0.3283
EXP-cVAE-TF-Pol2	0.5369	0.9380	1.0000	0.9989	0.1998

Table 11. Style transfer performance of ZINC250k models.

Model	PCT Valid ↑	PCT Fail ↓	MAE ↓	Tan ↑	Tan-B1	Tan-B2
cVAE	0.3944	0.3765	1.5439	0.5755	0.0921	0.1186
EXP-cVAE	0.6492	0.5699	1.3499	0.6227	0.0741	0.1164
EXP-cVAE-KLD	0.4308	0.2953	1.3105	0.5700	0.1125	0.1149
EXP-cVAE-Pol2	0.6692	0.5642	1.2572	0.6115	0.1138	0.1164
EXP-cVAE-TF	0.8448	0.2656	0.7114	0.4311	0.1013	0.1136
EXP-cVAE-TF-KLD	0.8456	0.0941	0.4119	0.4298	0.1129	0.1115
EXP-cVAE-TF-Pol2	0.8724	0.1444	0.3759	0.4214	0.1145	0.1142

C. Examples of Style Transfer Molecules

Figure 2 shows five examples of successful style transfer for a model trained with the calibration regularizer and explicit decoder (without teacher forcing). We see the model is able to find relatively simple and effective solutions to alter the LogP value.

To better visualise the positive effect of our regularizers, we can examine their ability to edit a molecular structure for a range of target property scores. To show this, we plotted the achieved property scores over the target property scores for the molecules with the lowest, median and highest LogP scores. These results are presented for the explicit decoder models with no regularizer (Figure 3), calibration regularizer (Figure 4), and the second reward regularizer (Figure 5).

An ideal model would perfectly match the target and contain points along the diagonal. We see that the introduction of the regularizers (bottom two plots) are significantly closer to this. In particular they more flexible, and are able to find more distinct solution modes. These are visible in the horizontal striations. The uneven structure of these plots demonstrates the dependence on the ability of a model to perform style transfer on both the source molecule and target property. For example, the molecule on the right hand side appears to be easier to edit for all models.

We repeated this experiment for the teacher forcing decoder (Figures 6, 7 and 8). The base model performs much better due to its increased focus on conditioning information. However we see that both regularizers still improve the range of achieved properties, diversity of solution modes and fidelity.

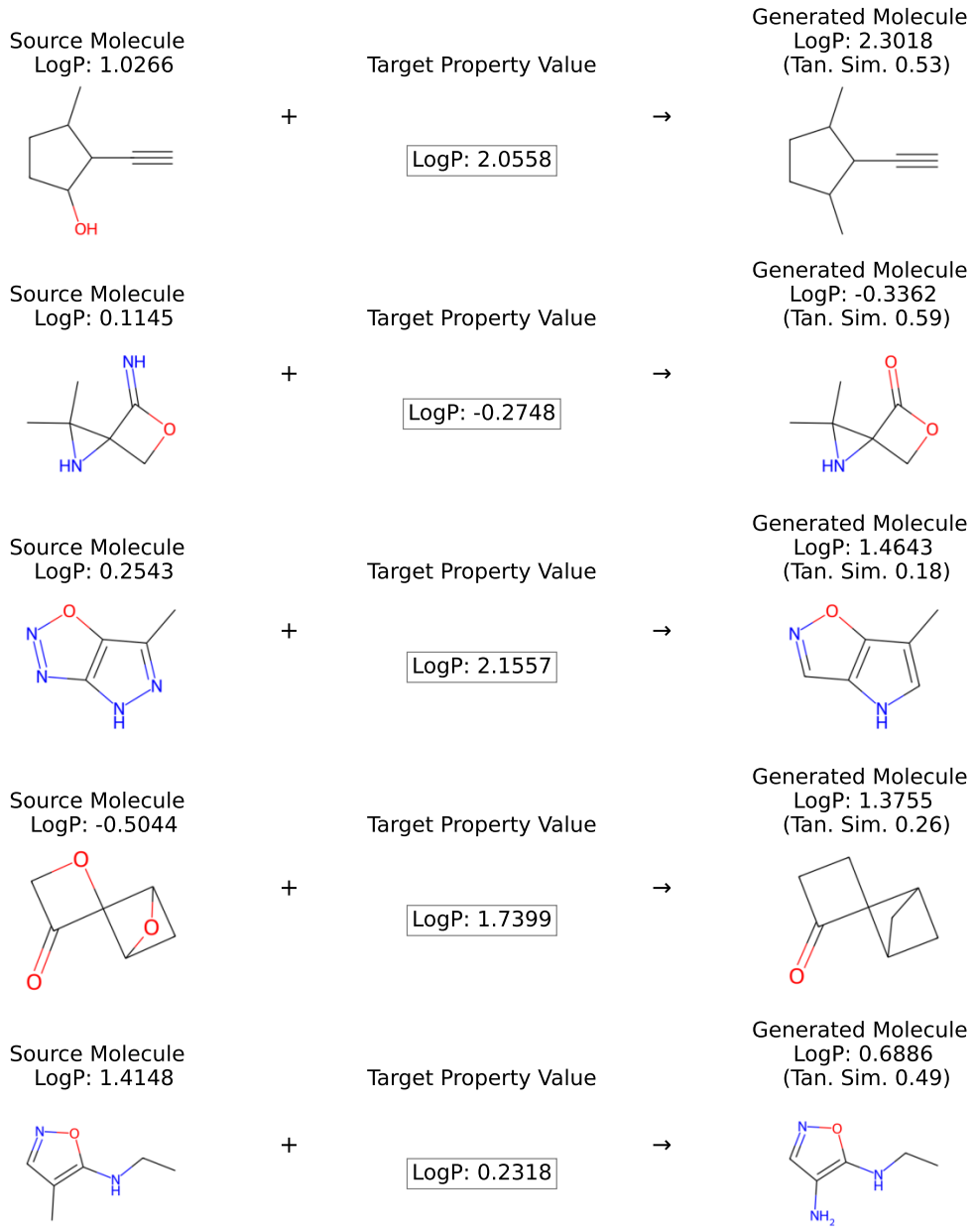

Figure 2. Examples of successful style transfer with EXP-cVAE-TF-Pol2 model.

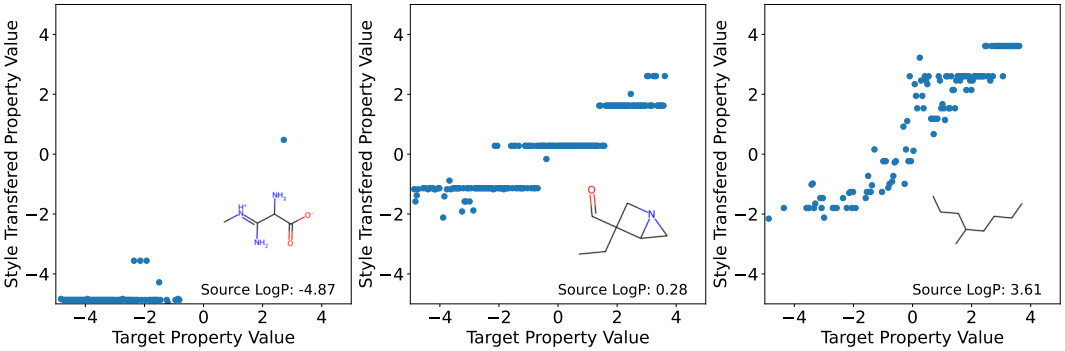

Figure 3. Style transfer generation property scores for a range of target properties sampled from EXP-cVAE model.

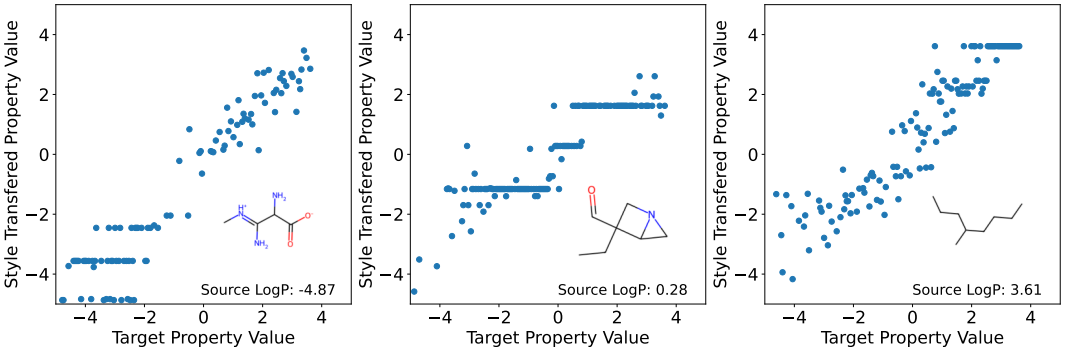

Figure 4. Style transfer generation property scores for a range of target properties sampled from EXP-cVAE-KLD model.

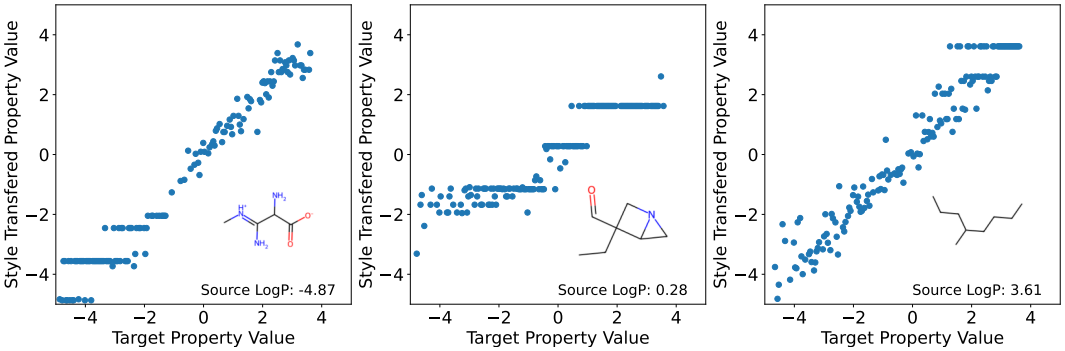

Figure 5. Style transfer generation property scores for a range of target properties sampled from EXP-cVAE-Pol2 model.

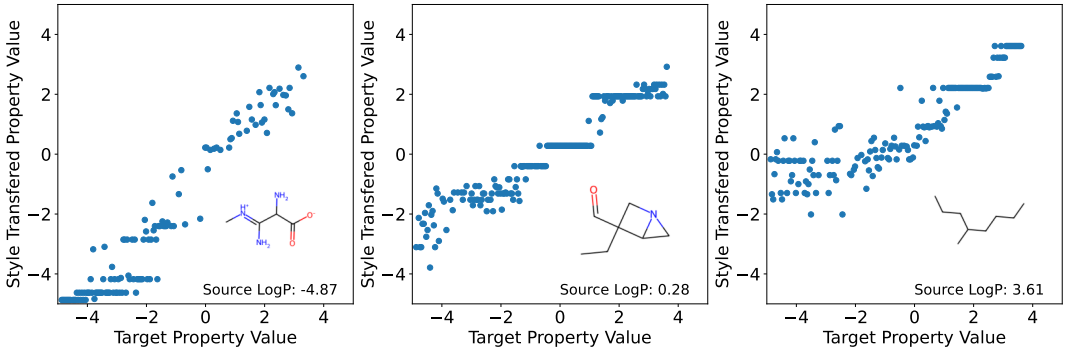

Figure 6. Style transfer generation property scores for a range of target properties sampled from EXP-cVAE-TF model.

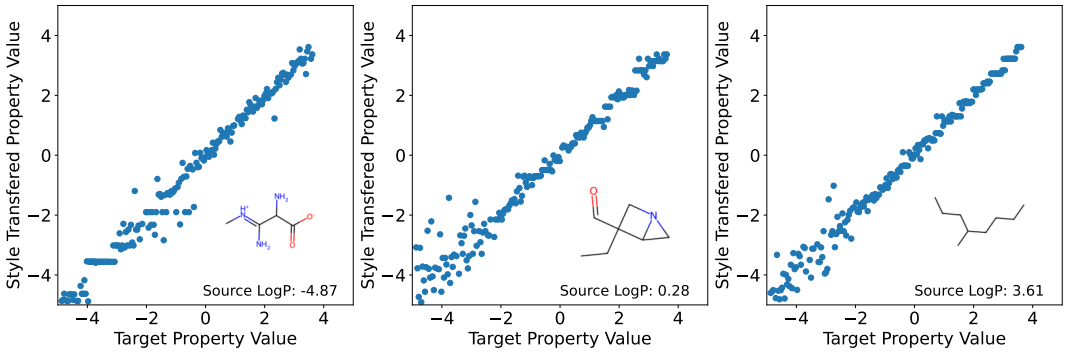

Figure 7. Style transfer generation property scores for a range of target properties sampled from EXP-cVAE-TF-KLD model.

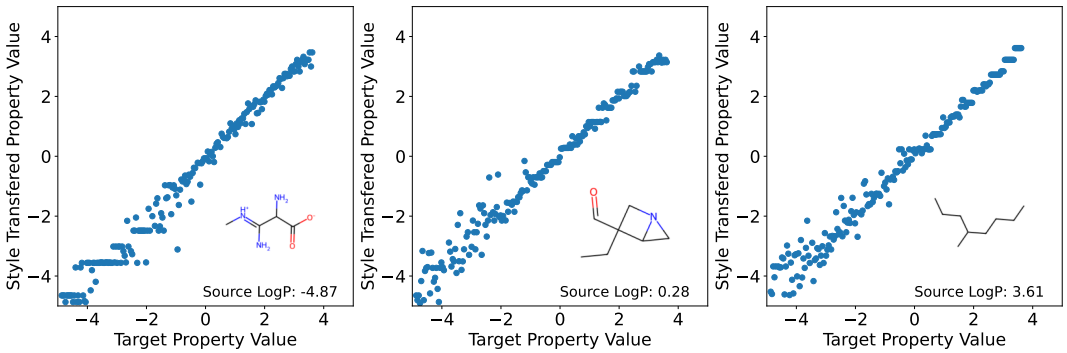

Figure 8. Style transfer generation property scores for a range of target properties sampled from EXP-cVAE-TF-Pol2 model.

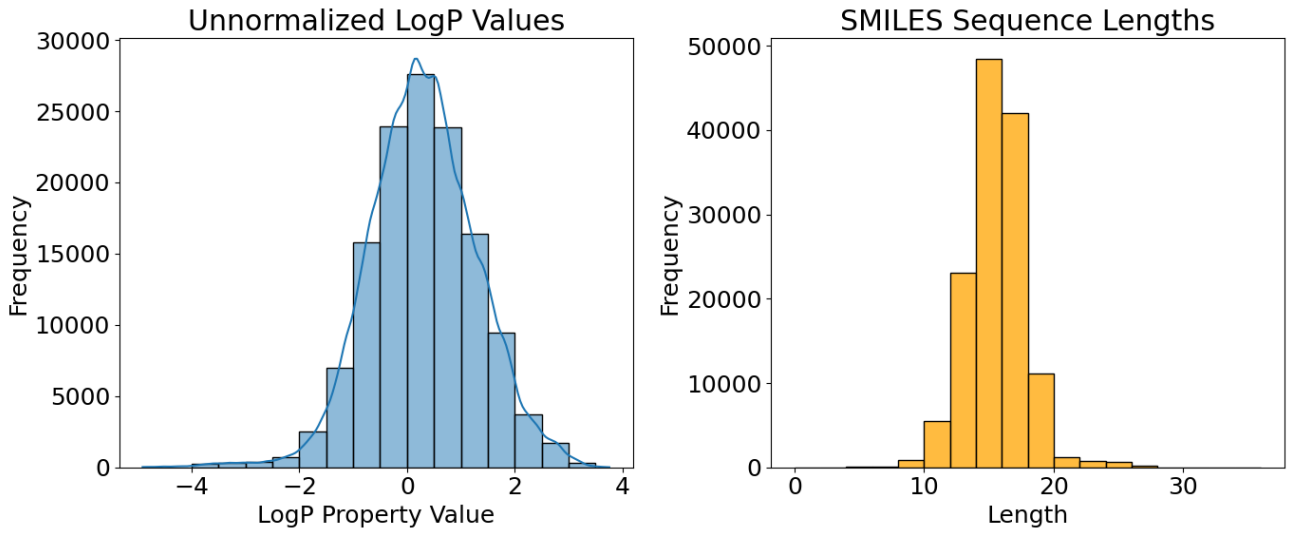

Figure 9. Summary statistics for QM9 dataset.

D. Dataset and Property Statistics

D.1. Dataset and Property Processing

Training used properties and canonical SMILES extracted from the QM9 dataset file (provided as atom coordinates and types) using the RDKit chemo-informatics package (version 2023.09.6), ignoring molecules that resulted in invalid SMILES. The resulting SMILES were encoded as integers and augmented with start and end tokens. The remainder of each sequence was padded with 0 tokens.

D.2. Data Splits

We withheld 10000 random molecules from the dataset for the validation set (tracking progress and determining early stopping) and 10000 for the test set (performance metric computation). The remaining 113885 molecules were used as the training set. The same test-train-validation split was used to train all models, including the baseline model from (Lim et al., 2018a) and the LSTM model for the regularizers.

D.3. Data Exploration

All LogP values were normalized according to the distribution in the training-set (mean=0.2982, std=1.0009). The distributions of unnormalized LogP values and sequence lengths for the entire dataset are presented in Figure 9. The frequencies of various characters in the entire dataset can be found in Table 12.

Table 12. Symbol frequencies in entire QM9 dataset.

Symbol	Freq.	Norm. Freq.	Symbol	Freq.	Norm. Freq.	Symbol	Freq.	Norm. Freq.
C	767830	0.38	=	94597	0.05	H	10568	0.01
1	259368	0.13	c	78726	0.04	o	10174	0.01
O	177823	0.09	n	41409	0.02	4	4952	0.00
2	131751	0.07	#	37027	0.02	F	3314	0.00
(	118216	0.06	3	35634	0.02	-	1974	0.00
)	118216	0.06	[	12523	0.01	+	1847	0.00
N	98355	0.05	]	12523	0.01	5	174	0.00

E. Model Architecture and Training Parameters

E.1. Decoder Structure

Our experiments employ three different decoder structures: the one-shot decoder (cVAE), explicitly auto-regressive decoder without teacher forcing (EXP-cVAE) and explicit auto-regressive decoder with teacher-forcing:

- **One-Shot**: The decoder does not accept the tokens generated from previous steps as input and propagates sequential information solely through the hidden state. Sequences are sampled from the model's predictions by independently sampling tokens with probabilities proportionate to each decoder output for each step in the sequence.
- **Explicit/Auto-Regressive**: At each step in the sequence t, a token is sampled proportionately to the decoders outputs and provided as an input at $t + 1$.
- **Explicit/AR with Teacher Forcing**: At each step in the sequence t, a token is sampled proportionately to the decoder outputs. The token provided as input to the decoder at $t + 1$ is taken from the ground truth SMILES string. During inference the model is sampled in the same manner as the non-teacher forcing version (EXP-cVAE).

E.2. LSTM Parameters

For the surrogate distribution $\tilde{p}(\mathbf{x}|\mathbf{y})$, we trained a conditional LSTM model with a maximum likelihood objective until convergence. The performance of this pre-trained model is shown in Table 3. The LSTM model consisted of an initial embedding layer (47x512) three stacked LSTM layers with input feature size 513 and output feature size 512 and a final linear layer with input size 512 and output size 47. Conditional information was provided by prepending it to the input at each time step. LSTM layers were trained with a dropout probability of 0.2. Initial and final layer weights were initialized using a xavier uniform initialization strategy.

E.3. VAE Architecture and Training

All VAE models shared a common architecture and parameters (except for regularizer specific parameters and the latent variable dimensionality). Training was conducted for up to 500 epochs, with early stopping after 50 epochs without validation loss improvement. Regularizer weights were determined so that the initial regularizer loss component was approximately an order of magnitude smaller than the initial KL loss component. The ELBO KL loss component was annealed according to a sigmoid schedule (from 0.01 to 1.0) centered at epoch 29 and ending at epoch 120.

The VAE model encoder consisted of an initial embedding layer followed by three 1 dimensional convolutional layers of dimensions 47x9x9, 9x9x9 and 9x10x11, each followed by a ReLu activation function. These were fed into two linear layers, one for predicting the embedding mean (435x56) and one for the log variance (435x56). The state decoder consisted an initial embedding layer (47x56) followed by three stacked GRU layers (113x501, 501x501, 501x501) each followed and a final output layer (501x47). The models were trained with gradient clipping above set to magnitude 0.2 and a learning rate scheduler set to reduce learning rate on plateaus to a minimum of 1e-6. An overview of the training parameters is presented in Table 13.

E.4. Implementation and Hardware Details

The model was implemented using the PyTorch library (version 2.2.1) and trained on a NVIDIA GeForce RTX 4090 graphics card using CUDA version 12.3.

Table 13. Training parameters for VAE models.

Parameter	Value
QM9 Latent Dim	156
ZINC Latent Dim	192
Optimizer	Adam
Batch Size	128
Learning Rate	1e-4
ELBO KL	0.001 to 1.0
Calib. Regularizer Weight	0.1
Pol1 Regularizer Weight	0.01
Pol1 MC Sample Size for Latent Points	4
Pol1 MC Sample Size for Decoder	20
Pol2 Regularizer Weight	0.01
Pol2/ MC Sample Size	60

F. Metric Computation Details

Reconstruction was computed from by variationally encoding a 1000 test set SMILES 5 times each. These were then decoder a further 5 times. The reported reconstruction score is the proportion of resulting SMILES that are valid and who's canonical form is identical to the input SMILES."

F.1. Conditional Generation Metrics

Conditional generation metrics were computed by sampling 1000 latent points $\mathbf{z}$ and equipping each with a random $\mathbf{y}$ from the test set. Each pair was decoded 5 separate times to produce the final sample. We evaluated conditional generation with the following metrics:

- **Validity**: The proportion of valid molecules in the sample. We considered SMILES successfully parsed by RdKit to be valid.
- **Uniqueness**: The number of unique valid molecules among all valid molecules. Uniqueness was computed from valid SMILES canonicalized by RdKit.
- **Novelty**: The proportion of canonical forms of valid SMILES not present in the training dataset.
- **Property MAE**: The mean absolute error between conditioning properties and properties computed from the parsed valid SMILES.

F.2. Style Transfer Metrics

"Style transfer (molecular optimization) metrics were computed by variationally encoding 2500 test set SMILES (the source SMILES). Each of these was then equipped with an independently sampled test set property (the target property), and decoded to produce the final generated molecule. The following metrics were used to evaluate style transfer performance:

- **PCT Valid:** The proportion of generated molecules that are successfully parsed by RdKit.
- **PCT Fail:** The proportion of generated molecules that are identical to the source molecules. In these cases the model has failed to consider the conditional information and change the molecule.
- **MAE:** The MAE between the target property and property score computed from valid generated SMILES.
- **Tan:** The structural similarity between the source and generated molecule. Computed as the Tanimoto similarity between the Morgan fingerprints (with radius 3) of each molecule and ranging from 0 to 1.

F.3. Baselines for Tanimoto Similarity

The first Tanimoto baseline was computed as the average similarity between the set SMILES generated through style transfer and a paired set of generations using random latent points instead of the encoded source molecules. The second Tanimoto baseline compared the similarity of two sets of style transferred SMILES with a shared set of target properties.

F.4. Baseline for Property MAE

We determined a baseline upper-bound for the MAE between two molecules from the QM9 setting at 1.1213. It was computed as the MAE between each possible pair of test set molecules and represents a case where property information has been completely ignored.

F.5. Comparative Evaluation of Models

Both conditional generation and molecular optimization are multifaceted problems. For example, if conditional generation model fails to generate either valid, unique or novel molecules, it would not be practical for use in drug design. Similarly, molecular optimization involves a trade off between structural similarity to the source and respecting the conditioning information. This trade off necessarily exists because a molecules property and structure cannot be completely disentangled and consequently it is impossible to change a molecules property without affecting its structure. The degree to which this is possible also depends on both the specific molecule under consideration and the nature of the property being considered. As a result it is difficult to give a general ranking between models where one is not Pareto dominant, and we must instead consider the trade off between the two.