De Novo Drug Design with JOINT TRANSFORMERs

Adam Izdebski
University of Warsaw
adam.izdebski@mimuw.edu.pl

Ewelina Weglarz-Tomczak
NatInLab
e.weglarz.tomczak@natinlab.com

Ewa Szczurek
University of Warsaw
szczurek@mimuw.edu.pl

Jakub M. Tomczak
Eindhoven University of Technology
j.m.tomczak@tue.nl

Abstract

De novo drug design requires simultaneously generating novel molecules outside of training data and predicting their target properties, making it a hard task for generative models. To address this, we propose JOINT TRANSFORMER that combines a Transformer decoder, Transformer encoder, and a predictor in a joint generative model with shared weights. We formulate a probabilistic black-box optimization algorithm that employs JOINT TRANSFORMER to generate novel molecules with improved target properties and outperforms other SMILES-based optimization methods in *de novo* drug design.

1 Introduction

De novo drug design is an approach to generate novel structures with desired properties from scratch. It opens the door to new classes of drugs, promising to overcome limitations of existing treatments (Schneider & Clark, 2019). While numerous breakthroughs in generative modeling and natural language processing (Vaswani et al., 2017; Radford et al., 2018) advanced the field of drug discovery, *de novo* design remains a notoriously challenging task (Wu et al., 2021; Grisoni, 2023).

De novo design requires to simultaneously (i) generate novel compounds, (ii) accurately predict their target properties and (iii) optimize the generation of compounds towards the desired properties (Brown et al., 2019). However, as the desired properties are rarely observed in the training data, there is an inherent trade-off between generation, prediction, and optimization. The more optimized towards properties from outside the training distribution the generation is, the less reliable the generation of compounds and prediction of their properties become.

Previous work on generative models for *de novo* drug design focused on each of the required components separately. Decoder-only Transformers (Radford et al., 2018) successfully generate novel and chemically plausible molecules (Bagal et al., 2022), but they have no information about the target properties. These can be fine-tuned or coupled with RL approaches, however, without yielding satisfactory results in practical regimes (Neil et al., 2018). Encoder-only Transformers excel at molecular property prediction tasks (Ross et al., 2022; Zhou et al., 2023), but they lack molecule generation capabilities. Optimization of molecules is often treated as a black-box optimization (BBO) problem (Terayama et al., 2021) and solved over a continuous latent space of a Latent Variable Model like a Variational Autoencoder (Kingma & Welling, 2013; Rezende et al., 2014), using an external optimization routine, e.g., Bayesian Optimization (Gómez-Bombarelli et al., 2018; Tripp et al., 2020). However, posing the problem as BBO and employing an external optimization routine tend to guide a generative model far from the true data distribution of chemically plausible molecules, in regions where the generation and prediction becomes unreliable. The lack of a coherent framework that would address all the above challenges at the same time motivates the need for a joint approach.

In this paper, we propose JOINT TRANSFORMER, a joint generative model that simultaneously generates novel examples and accurately predicts their target properties. We achieve this by combining a Transformer decoder (driving the generative performance) with a Transformer encoder and a predictor (both encouraging predictive performance). We propose to train the joint generative model with a penalized log-likelihood objective, which allows simultaneous training of the decoder, encoder, and predictor, enabling joint training and sharing all the weights. Equipped with JOINT TRANSFORMER, we pose *de novo* drug design as a probabilistic version of the BBO problem, where we aim to optimize a given objective function, like in BBO, but only in the regions of the input space, where the generative model assigns high likelihood to samples and the generative and predictive capabilities of the model remain reliable. Finally, we propose a sampling algorithm that utilizes the strong generative and predictive performance of the model to generate optimized molecules. In the experiments on GuacaMol benchmark (Brown et al., 2019), we show that JOINT TRANSFORMER outperforms standard SMILES-based approaches for *de novo* drug design.

2   Methodology

Problem Statement   We define the problem of *de novo* drug design as an extension of the *black-box* optimization (BBO) (Alarie et al., 2021; Audet & Hare, 2017; Terayama et al., 2021), in which we aim to find examples $\mathbf{x}^* \in \mathcal{X}$ that maximize a given *objective function* $f : \mathcal{X} \rightarrow \mathbb{R}$, $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$. Additionally, we define define an example $\mathbf{x} \in \mathcal{X}$ as *semantically meaningful*, if $\mathbf{x}$ could have been generated by the true data generating distribution $p(\mathbf{x})$. This leads to the *probabilistic* BBO (PBBO) defined as the problem of sampling examples $\mathbf{x}^* \in \mathcal{X}$ maximizing the objective function f that could have been generated by the true underlying data distribution, i.e., they are semantically meaningful:

$$\mathbf{x}^* \sim p(\mathbf{x} \mid y_{\max}), \text{ where } y_{\max} = \max_{\mathbf{x} \sim p(\mathbf{x})} f(\mathbf{x}). \tag{1}$$

Our approach   To address the problem of learning and sampling from the conditional distribution $p(\mathbf{x} \mid y)$ in Eq. 1, we propose a joint generative model of examples and corresponding targets. The advantage of such an approach is twofold. First, joint modeling encourages sharing the weights used for generation and prediction, making robust prediction of target values on newly generated examples feasible. Second, the robust predictions give a good indication of whether the newly generated examples have high values of the target. Indeed, the joint generative model allows sampling examples $\mathbf{x}$ that fulfill Eq. 1 and satisfy the desired condition $y \geq y_c$, for every $y_c \in \mathbb{R}$.

The proposed joint generative model, JOINT TRANSFORMER, $p_{\theta,\phi}(\mathbf{x}, y)$ combines three models: a Transformer decoder $p_\theta(\mathbf{x})$, a Transformer encoder $\prod_{d=1}^{D} p_\theta(x_d \mid \mathbf{m} \odot \mathbf{x}_{-d})$, and a predictor $p_{\theta,\phi}(y \mid \mathbf{x})$. The weights θ are shared between the encoder, decoder, and predictor parts. Additionally, the predictor (used either for regression or classification) is stacked on the top of the encoder and is parametrized with weights ϕ. The difference between the decoder and the encoder lies only in the choice of masking used within attention layers, namely, the decoder uses causal masking while the encoder applies bidirectional masking.

The rationale behind our model is the following. First, we share weights θ to entangle the generation and prediction tasks and make robust predictions of target values on newly generated examples feasible. At the same time, sharing weights has the practical advantage of a more computationally efficient model. Second, alongside the Transformer decoder, we incorporate the Transformer encoder to let the predictor learn better representations and process the input in a non-sequential manne, as a lack of bidirectional context may be harmful to predictive performance (Devlin et al., 2018).

In order to learn a single model that combines a Transformer encoder, a Transformer decoder, and a predictor, we propose to minimize a penalized negative log-likelihood of the joint model given by:

$$\ell(\theta, \phi) = -\mathbb{E}_{(\mathbf{x},y) \sim p(\mathbf{x},y)} \Bigg\{ \ln p_\theta(\mathbf{x}) + \ln p_{\theta,\phi}(y \mid \mathbf{x}) +$$

$$\mathbb{E}_{\mathbf{m} \sim q(\mathbf{m})} \left[\sum_{d=1}^{D} \ln p_\theta(x_d \mid \mathbf{m} \odot \mathbf{x}_{-d}) \right] \Bigg\}, \tag{2}$$

where $q(\mathbf{m})$ is an arbitrary masking distribution, and the penalty (the last component of the sum) is the *pseudolikelihood*. Using the penalized negative log-likelihood objective in Eq. 2 encourages the

model to simultaneously operate in two separate modes: input generation and property prediction. First, updating the decoder and learning to process the input in an autoregressive manner drives the generative performance of the model. Second, updating the encoder and learning to process the input in a bidirectional manner drives learning a meaningful representation and the predictive performance, since the predictor shares weights θ with the encoder. The training is discussed in Appendix B.

Probabilistic Black Box Optimization with JOINT TRANSFORMERS   We define PBBO as the problem of sampling from the conditional distribution $\mathbf{x} \sim p(\mathbf{x} \mid y_c)$, where target $y_c \in \mathbb{R}$ is equal or close to the optimal value of the objective function f. However, in Proposition 1 in Appendix C.2 we show that sampling $\mathbf{x} \sim p(\mathbf{x} \mid y_c)$ is equivalent to the conditional generation from a joint model like JOINT TRANSFORMER. Moreover, in Proposition 2 in Appendix C.2 we show that conditional sampling is practically feasible, as long as target y_c is within the support of JOINT TRANSFORMER. In practice, to fix a feasible threshold y_c, one can set a sampling budget of $B \in \mathbb{N}$ examples and then rank examples according to $p(y \mid \mathbf{x})$ choosing the best examples available. Algorithm 1 shows how to facilitate PBBO using JOINT TRANSFORMER.

Algorithm 1 Probabilistic Black-Box Optimization with JOINT TRANSFORMER

Input: JOINT TRANSFORMER $p_{\theta,\phi}(\mathbf{x}, y)$ with parameters θ, ϕ.
    Threshold $y_c \in \mathcal{Y}$. Evaluation budget $I \in \mathbb{N}$. Sampling budget $B \in \mathbb{N}$.
Output: $\mathcal{D}_{\text{new}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{min(I,B)}$
 1: $b \leftarrow 0, i \leftarrow 1$
 2: **while** $b < B$ **and** $i \leq I$ **do**
 3:    Unconditionally sample $(\mathbf{x}_i, y_i) \sim p_{\theta,\phi}(\mathbf{x}, y)$
 4:    **if** $y_i \geq y_c$ **then**
 5:       $\mathcal{D}_{\text{new}} \leftarrow \mathcal{D}_{\text{new}} \cup \{(\mathbf{x}_i, y_i)\}$
 6:       $i \leftarrow i + 1$
 7:    **end if**
 8:    $b \leftarrow b + 1$
 9: **end while**

The proposed optimization procedure addresses the needs of *de novo* drug design in several ways. First, it is probabilistic and therefore tailored to avoid non-realistic molecules as sampled examples. Second, it gives a guarantee of sampling improved examples. Third, it deals with the issue of a low fraction of labeled training data (low fraction of known target values for the molecules) by incorporating unsupervised training phases.

3 Experiments

Setting   To validate the applicability of our approach, we show that JOINT TRANSFORMER combined with the probabilistic Black Box Optimization algorithm (Alg. 1) outperforms alternative methods for generating optimized molecules, as illustrated by application to a *de novo* drug design task. We choose the architecture of GPT (Radford et al., 2018) for JOINT TRANSFORMER in all tasks. The same architecture was previously utilized in MolGPT by Bagal et al. (2022). However, the training of our model is different from Bagal et al. (2022), and follows Alg. 3. Implementation details for JOINT TRANSFORMER are outlined in Appendix E. We use a JOINT TRANSFORMER pre-trained, in an unsupervised manner, using molecules derived from the ChEMBL 24 dataset (Mendez et al., 2019), following Brown et al. (2019) for processing and splitting the data. We additionally fine-tune JOINT TRANSFORMER using penalized log-likelihood on randomly selected subsets of the training data ($N = 1000$) with multi-property objective functions (MPO), derived from the GuacaMol benchmark (Brown et al., 2019), as continuous targets in the range $[0, 1]$, namely, Perindopril MPO, Sitagliptin MPO, and Zaleplon MPO, which are also the three hardest to optimize (Gao et al., 2022).

Methods   We compare the BBO with JOINT TRANSFORMER (Alg. 1) to other SMILES-based molecule optimization methods. In particular, we choose the three best-performing methods across all tasks in a benchmark comparing 25 various optimization methods (Gao et al., 2022): SMILES GA (Yoshikawa et al., 2018), REINVENT (Olivecrona et al., 2017) and an LSTM combined with

a hill-climbing algorithm (LSTM + HC) (Brown et al., 2019). Additionally, we report the Dataset Best value, which is the best value of the objective function present in the dataset, as the upper bound for all screening methods, and MolPal (Graff et al., 2021), which is a deep-learning-based screening method. We include in the comparison the Variational Autoencoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014) combined with Bayesian optimization (VAE + BO) and a Junction Tree VAE (Jin et al., 2018) combined with Bayesian optimization (JT-VAE + BO). Finally, we compare our method to a standard, unconditional decoder-only Transformer model, fine-tuned on examples with corresponding objective values above a fixed threshold $y_c \in \mathbb{R}$ (MolGPT + fine-tune).

Results   JOINT TRANSFORMER is the only method in this experiment that generates molecules better than in the dataset across all tasks, successfully performing *de novo* design (Table 1). The fine-tuned MolGPT is next in line in terms of performance. JOINT TRANSFORMER generates optimized molecules for an evaluation budget as low as 137, 452, and 17 evaluations, for the three optimization tasks respectively - outperforming other non-Transformer-based methods by a large margin. An additional investigation of the distribution of the objective function values for molecules sampled from the Transformer-based models as compared to the data distribution (Figure 1), shows that JOINT TRANSFORMER significantly alters the distribution of the sampled molecules towards optimal objective values, as opposed to MolGPT that only slightly skews the initial data distribution.

Table 1: The highest value of the objective function across all generated molecules (Top1) within 10^3 evaluations. Mean and standard deviation across three independent data splits.

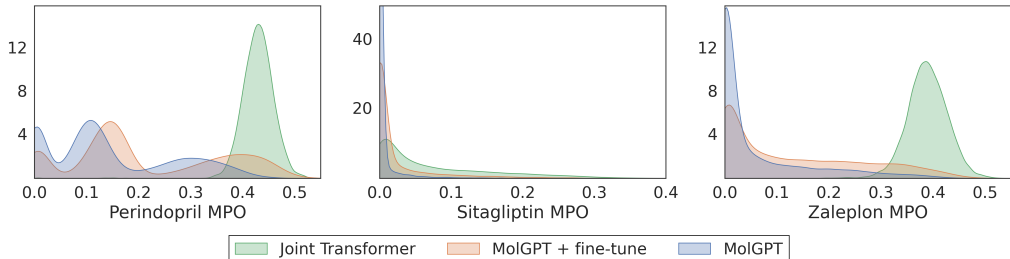| METHOD	PERINDOPRIL MPO	SITAGLIPTIN MPO	ZALEPLON MPO
DATASET BEST	0.53 ± 0.00	0.40 ± 0.02	0.50 ± 0.01
MOLPAL	0.49 ± 0.01	0.05 ± 0.01	0.16 ± 0.09
SMILES GA	0.44 ± 0.01	0.21 ± 0.10	0.28 ± 0.08
REINVENT SMILES	0.45 ± 0.01	0.02 ± 0.01	0.27 ± 0.03
LSTM + HC	0.47 ± 0.01	0.02 ± 0.02	0.14 ± 0.03
VAE + BO	0.45 ± 0.01	0.03 ± 0.01	0.04 ± 0.04
JT-VAE + BO	0.43 ± 0.01	0.05 ± 0.03	0.13 ± 0.11
MOLGPT + FINE-TUNE	0.54 ± 0.03	0.40 ± 0.03	0.51 ± 0.00
JOINT TRANSFORMER (OURS)	$\mathbf{0.55 \pm 0.01}$	$\mathbf{0.43 \pm 0.01}$	$\mathbf{0.55 \pm 0.02}$

Figure 1: Distribution of the objective function values sampled from different models: JOINT TRANSFORMER (green), MolGPT + fine-tune (orange), MolGPT (blue). Best viewed in color.

4   Conclusion

In this paper, we formulated the problem of de novo drug design as an instance of probabilistic BBO (Section 2). We proposed a general-purpose sampling algorithm that performs probabilistic BBO with any joint generative model (Section 2), with theoretical guarantees on the expected runtime as the function of the training data (Section C.2). Finally, we proposed a joint generative model, called JOINT TRANSFORMER, that combines a Transformer decoder, a Transformer encoder, and a predictor in a single model with shared parameters, which is jointly trained with a penalized log-likelihood objective (Eq 2). We empirically showed that JOINT TRANSFORMER successfully outperforms state-of-the-art approaches to *de novo* drug design.

References

Stéphane Alarie, Charles Audet, Aïmen E. Gheribi, Michael Kokkolaras, and Sébastien Le Digabel. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, 9:100011, 2021. ISSN 2192-4406. doi: https://doi.org/10.1016/j.ejco.2021.100011. URL https://www.sciencedirect.com/science/article/pii/S2192440621001386.

Charles Audet and Warren Hare. *Derivative-free and blackbox optimization*. Springer, 2017.

Viraj Bagal, Rishal Aggarwal, P. K. Vinod, and U. Deva Priyakumar. MolGPT: Molecular Generation Using a Transformer-Decoder Model. *Journal of Chemical Information and Modeling*, 62(9): 2064–2076, May 2022. ISSN 1549-9596. doi: 10.1021/acs.jcim.1c00600.

Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. GuacaMol: Benchmarking Models for de Novo Molecular Design. *Journal of Chemical Information and Modeling*, 59(3): 1096–1108, 2019. ISSN 1549-9596. doi: 10.1021/acs.jcim.8b00839.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Peter Ertl, Richard Lewis, Eric Martin, and Valery Polyakov. In silico generation of novel, drug-like chemical matter using the LSTM neural network, January 2018.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization. *Advances in Neural Information Processing Systems*, 35: 21342–21357, December 2022.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276, February 2018. ISSN 2374-7943. doi: 10.1021/acscentsci.7b00572.

David E. Graff, Eugene I. Shakhnovich, and Connor W. Coley. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chemical Science*, 12(22):7866–7881, June 2021. ISSN 2041-6539. doi: 10.1039/D0SC06805E.

Francesca Grisoni. Chemical language models for de novo drug design: Challenges and opportunities. *Current Opinion in Structural Biology*, 79:102527, 2023.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Leon Hetzel, Johanna Sommer, Bastian Rieck, Fabian Theis, and Stephan Günnemann. Magnet: Motif-agnostic generation of molecules from shapes. *arXiv preprint arXiv:2305.19303*, 2023.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

John J. Irwin, Khanh G. Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R. Wong, Munkhzul Khurelbaatar, Yurii S. Moroz, John Mayfield, and Roger A. Sayle. ZINC20—A Free Ultralarge-Scale Chemical Database for Ligand Discovery. *Journal of Chemical Information and Modeling*, 60(12):6065–6073, 2020. ISSN 1549-9596. doi: 10.1021/acs.jcim.0c00675.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2323–2332. PMLR, July 2018.

Artur Kadurin, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, 8(7):10883–10890, December 2016. ISSN 1949-2553. doi: 10.18632/oncotarget.14073.

Andrej Karpathy. minGPT, September 2023.

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

Julia A Lasserre, Christopher M Bishop, and Thomas P Minka. Principled hybrids of generative and discriminative models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pp. 87–94. IEEE, 2006.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Krzysztof Maziarz, Henry Jackson-Flux, Pashmina Cameron, Finton Sirockin, Nadine Schneider, Nikolaus Stiefl, Marwin Segler, and Marc Brockschmidt. Learning to extend molecular scaffolds with structural motifs. *arXiv preprint arXiv:2103.03864*, 2021.

David Mendez, Anna Gaulton, A. Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F. Mosquera, Prudence Mutowo, Michal Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J. Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R. Leach. ChEMBL: Towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1): D930–D940, 2019. ISSN 1362-4962. doi: 10.1093/nar/gky1075.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pp. 4723–4732. PMLR, 2019.

Daniel Neil, Marwin Segler, Laura Guasch, Mohamed Ahmed, Dean Plumbley, Matthew Sellwood, and Nathan Brown. Exploring deep recurrent models with reinforcement learning for molecule design. 2018.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular De Novo Design through Deep Reinforcement Learning, August 2017.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet ChemNet Distance: A metric for generative models for molecules in drug discovery, August 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-scale chemical language representations capture molecular structure and properties, 2022.

Gisbert Schneider and David E Clark. Automated de novo drug design: are we nearly there yet? *Angewandte Chemie International Edition*, 58(32):10792–10803, 2019.

Philippe Schwaller, Daniel Probst, Alain C. Vaucher, Vishnu H Nair, David Kreutter, Teodoro Laino, and Jean-Louis Reymond. Mapping the space of chemical reactions using attention-based neural networks. *ChemRxiv*, 2020. doi: 10.26434/chemrxiv.9897365.v4.

Kei Terayama, Masato Sumita, Ryo Tamura, and Koji Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021. doi: 10.1021/acs.accounts. 0c00713. URL https://doi.org/10.1021/acs.accounts.0c00713. PMID: 33635621.

Igor V Tetko, Pavel Karpov, Eric Bruno, Talia B Kimber, and Guillaume Godin. Augmentation is what you need! In *International Conference on Artificial Neural Networks*, pp. 831–835. Springer, 2019.

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33:11259–11272, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, February 1988. ISSN 0095-2338. doi: 10.1021/ci00057a005.

Zachary Wu, Kadina E. Johnston, Frances H. Arnold, and Kevin K. Yang. Protein sequence design with deep generative models. *Current Opinion in Chemical Biology*, 65:18–27, 2021. ISSN 1367-5931. doi: https://doi.org/10.1016/j.cbpa.2021.04.004. URL https://www.sciencedirect.com/science/article/pii/S136759312100051X. Mechanistic Biology * Machine Learning in Chemical Biology.

Naruki Yoshikawa, Kei Terayama, Masato Sumita, Teruki Homma, Kenta Oono, and Koji Tsuda. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 47(11):1431–1434, 2018.

Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. *ChemRxiv*, 2023. doi: 10.26434/chemrxiv-2022-jjm0j-v4.

A   Author contribution

Anonymized for the submission **AI** - devised the project, designed and performed all experiments, the lead in writing the manuscript

EWT - data preparation (Screening experiment), feedback on writing

ES - supervision, the lead in writing the manuscript

JT - devised the project, supervision, designed experiments and performed "Screening" experiment, the lead in writing the manuscript

B   Training

Training a joint generative model was previously shown to result in a good generator together with a poor predictor (Lasserre et al., 2006; Nalisnick et al., 2019). Moreover, storing the gradients for all the summands of the objective (Eq. 2) is a significant overhead in memory requirements as compared to decoder and encoder-only Transformers. To overcome both issues, we propose a practical training procedure for JOINT TRANSFORMER (Alg. 3) that randomly switches between the input generation and the property prediction (and encoder training) tasks with a hyperparameter $p_{\text{task}} \in [0, 1]$.

The JOINT TRANSFORMER can be trained in an unsupervised, semi-supervised or supervised setting. Depending whether a target $y \in \mathcal{Y}$ is sampled from the dataset $\mathcal{D}$ or is not available (Step 2, Alg. 3), one can include the prediction loss $\ln p_{\theta,\phi}(y \mid \mathbf{x})$ in the penalized log-likelihood objective ℓ (Step 6, Alg. 3), resulting in a supervised setting, or set the prediction loss to zero, resulting in an unsupervised setting. For the training data where only a small proportion of samples have accompanying target values, we split the training procedure of the JOINT TRANSFORMER into first training the model in an unsupervised manner (Alg. 2), then fine-tuning it with supervised data (Alg. 3).

Algorithm 2 Unsupervised training of JOINT TRANSFORMER

Input: A dataset $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}$. JOINT TRANSFORMER $p_{\theta,\phi}(\mathbf{x}, y)$ with parameters θ, ϕ containing
   a decoder $p_\theta(\mathbf{x})$, encoder $\prod_{d=1}^{D} p_\theta(x_d \mid \mathbf{m} \odot \mathbf{x}_{-d})$.
   Task probability $p_{\text{task}} \in [0, 1]$ and a masking distribution $q(\mathbf{m})$.
 1: **while** a stopping criterion is not met **do**
 2:    Uniformly sample $\mathbf{x}$ from the dataset $\mathcal{D}$
 3:    Sample an indicator $u \sim \text{BERNOULLI}(p_{task})$
 4:    **if** $u = 0$ **then**
 5:       Sample mask $\mathbf{m} \sim q(\mathbf{m})$
 6:       Calculate loss $\ell(\theta, \phi) = -\sum_{d=1}^{D} \ln p_\theta(x_d \mid \mathbf{m} \odot \mathbf{x}_{-d})$
 7:    **else**
 8:       Set mask to the causal mask
 9:       Calculate loss $\ell(\theta, \phi) = -\ln p_\theta(\mathbf{x})$
10:    **end if**
11:    Update parameters θ, ϕ using an optimizer w.r.t. loss ℓ
12: **end while**

Algorithm 3 Training of JOINT TRANSFORMER

Input: A dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$. JOINT TRANSFORMER $p_{\theta,\phi}(\mathbf{x}, y)$ with parameters θ, ϕ containing a decoder $p_\theta(\mathbf{x})$, encoder $\prod_{d=1}^D p_\theta(x_d \mid \mathbf{m} \odot \mathbf{x}_{-d})$ and a predictor $p_{\theta,\phi}(y \mid \mathbf{x})$. Task probability $p_{\text{task}} \in [0, 1]$ and a masking distribution $q(\mathbf{m})$.

1: **while** a stopping criterion is not met **do**
2:     Uniformly sample $(\mathbf{x}, y)$ from the dataset $\mathcal{D}$
3:     Sample an indicator $u \sim \text{BERNOULLI}(p_{task})$
4:     **if** $u = 0$ **then**
5:         Sample mask $\mathbf{m} \sim q(\mathbf{m})$
6:         Calculate loss $\ell(\theta, \phi) = -\sum_{d=1}^D \ln p_\theta(x_d \mid \mathbf{m} \odot \mathbf{x}_{-d}) - \ln p_{\theta,\phi}(y \mid \mathbf{x})$
7:     **else**
8:         Set mask to the causal mask
9:         Calculate loss $\ell(\theta, \phi) = -\ln p_\theta(\mathbf{x})$
10:     **end if**
11:     Update parameters θ, ϕ using an optimizer w.r.t. loss ℓ
12: **end while**

C   Sampling

C.1   Unconditional Generation

In the unconditional generation task, we sample from JOINT TRANSFORMER in a two-step manner that results in an unconditional sample $(\mathbf{x}, y) \sim p_{\theta,\phi}(\mathbf{x}, y)$. First, since the decoder part $p_\theta(\mathbf{x})$ does not depend on parameters ϕ and it properly defines an ARM, we sample $\mathbf{x} \sim p_\theta(\mathbf{x})$. Next, we sample a target y from the predictive distribution $y \sim p_{\theta,\phi}(y \mid \mathbf{x})$. The key feature that allows for successful sampling from the joint model is the ability of the JOINT TRANSFORMER to simultaneously operate in two separate modes, namely generate novel examples and predict their target values, which is directly encouraged by training with the penalized log-likelihood objective in Eq. 2.

C.2   Conditional Generation

In the conditional generation task, given a condition $Y \subseteq \mathcal{Y}$, we sample from JOINT TRANSFORMER $p_{\theta,\phi}(\mathbf{x}, y)$ to obtain a conditional sample $(\mathbf{x}, y) \sim p_{\theta,\phi}(\mathbf{x}, y)$, such that $y \in Y$. JOINT TRANS-FORMER generates conditional samples by first sampling $(\mathbf{x}, y) \sim p_{\theta,\phi}(\mathbf{x}, y)$ in the above described unconditional way and then accepting the sample if $y \in Y$. In practical applications, due to a finite runtime, we sample a batch of B tuples $(\mathbf{x}, y) \sim p_{\theta,\phi}(\mathbf{x}, y)$ and choose $(\mathbf{x}, y)$ with y 'closest' to Y. Proposition 1 shows that, despite its conceptual simplicity, the described conditional generation procedure is equivalent to directly sampling from the conditional distribution $p(\mathbf{x} \mid y)$. Moreover, Proposition 2 shows conditions under which conditional generation enjoys a finite expected runtime.

Proposition 1. *Let $p(\mathbf{x}, y)$ be a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$. Let $y_c \in \mathcal{Y}$ be such that $p(y_c) > 0$. Then*

$$p(\mathbf{x} \mid y_c) \propto \mathbb{1}_{\{y=y_c\}}(y)p(y \mid \mathbf{x})p(\mathbf{x}).$$

Proof. Assume that $p(\mathbf{x}, y)$ is a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$. Choose $y_{\max} \in Y$ to be such that $p(y \geq y_c) > 0$. Then a simple application of Bayes rule yields

$$p(\mathbf{x} \mid \{y \geq y_c\}) = \frac{p(\mathbf{x}, \{y \geq y_c\})}{p(\{y \geq y_c\})} = \frac{\mathbb{1}_{\{y \geq y_c\}}(y)p(y \mid \mathbf{x})p(\mathbf{x})}{p(\{y \geq y_c\})}. \tag{3}$$

Since $p(\{y \geq y_c\}) > 0$ and it does not depend on $\mathbf{x}$, we have that

$$p(\mathbf{x} \mid \{y \geq y_c\}) \propto \mathbb{1}_{\{y \geq y_c\}}(y)p(y \mid \mathbf{x})p(\mathbf{x}).$$

$\square$

Proposition 2. *Let $p(y)$ be a probability distribution over $\mathcal{Y}$ with a corresponding cumulative distribution function F. Let target $y_c \in \mathcal{Y}$ be such that $p(y_c) > 0$ and let p be the probability of sampling a target $y \sim p(y)$ such that $y > y_c$. The expected number of trials N until obtaining a sample $y \sim p(y)$ such that $y > y_c$ is equal to $1/p$.*

Proof. Let $p(y)$ be a probability distribution over $\mathcal{Y}$ with a corresponding cumulative distribution function F. Let $y_c \in \mathcal{Y}$ be such that $p(y_c) > 0$. Define r.v. N as the number of trials until obtaining a sample $y > y_c$, where y is distributed as $p(y)$. For each $n \in \mathbb{N}$, the distribution of N is given by

$$P(N = n) = (1 - p)^{n-1} p,$$

where $p = 1 - F(y \leq y_c)$. Hence, the number of trials N follows a geometric distribution with an expected value equal to $\mathbb{E}[N] = 1/p$. $\qquad\square$

Despite its simplicity, the conditional generation of JOINT TRANSFORMER has the advantage of the predictor $p_{\theta,\phi}(y \mid \mathbf{x})$, as it is defined in the input space $\mathcal{X}$, indicating whether the newly generated example enjoys the desired target value. This is in contrast to methods based on LSO and diffusion models, see (Gómez-Bombarelli et al., 2018; Hoogeboom et al., 2022).

D Additional Experiments

D.1 Molecule Generation

Task   In the molecule generation task, the goal is to generate valid and novel molecules that follow the chemical distribution of the training data. Following Brown et al. (2019), we evaluate all molecule generation methods on five metrics: validity, a fraction of the generated molecules that are correspond to a valid SMILES string; uniqueness, a fraction of the generated molecules that are unique; novelty, a fraction of the generated molecules that are not present in the training data; KL Divergence, a measure of similarity of the generated molecules to the training set with respect to selected chemical properties (Brown et al., 2019), as well as Fréchet ChemNet Distance (FCD; (Preuer et al., 2018)), a general measure of similarity of the generated molecules to the training set.

Baselines   As baselines, we select well-established molecule generation models based on SMILES representation (Weininger, 1988): LSTM (Ertl et al., 2018), VAE (Kingma & Welling, 2013; Rezende et al., 2014) and AAE (Kadurin et al., 2016). Additionally, we consider graph-based models: Junction Tree VAE (Jin et al., 2018), MoLeR (Maziarz et al., 2021) and MAGNet (Hetzel et al., 2023). Finally, we include MolGPT (Bagal et al., 2022), which is a Transformer-based model and the backbone for the JOINT TRANSFORMER, sharing the same architecture, but trained differently.

Results   In the molecule generation task, JOINT TRANSFORMER successfully generates valid, unique and novel molecules (Tab. 2). Moreover, JOINT TRANSFORMER generates molecules with properties that closely follow the training set distribution, making the newly generated molecules realistic and physio-chemically plausible, as measured by KL Divergence and FCD. Compared to the backbone MolGPT model, JOINT TRANSFORMER achieves identical performance, showing that the modified training procedure does not hurt the generative functionality of the model. From the generative modeling perspective, this result is counterintuitive, as we can include the reconstruction task to the training procedure of the JOINT TRANSFORMER, without sacrificing its generative performance.

Overall, none of the molecule generation methods achieves best performance across all metrics. Graph-based methods outperform others on validity, as they generate always valid molecules by design. However, the improvement of 3% as compared to Transformer-based models (JOINT TRANS-FORMER and MolGPT) is negligible. Additionally, it comes at the expense of generating molecules with decreased (from 12% to 19%) values of the KL Divergence and FCD metrics. On the other hand, LSTM achieves top performance on KL Divergence and FCD metrics, slightly (1% and 3%, respectively) outperforming Transformer-based methods, but falls behind in the validity of the generated molecules. All methods successfully generate unique and novel molecules. Overall, JOINT TRANSFORMER strikes a good balance between graph-based and SMILES-based LSTM, making it a viable choice for a go-to molecule generation model.

D.2 Unconditional Generation

Moreover, the jointly trained predictor $q_{\theta,\phi}(y \mid \mathbf{x})$ of the JOINT TRANSFORMER generalizes well to data generated with the model $p_\theta(\mathbf{x})$. In particular, the prediction error, as measured by mean

Table 2: Molecule Generation Task. JOINT TRANSFORMER (JT) matches state-of-the-art performance of different molecule generation methods. Training the JOINT TRANSFORMER model on generation and reconstruction tasks simultaneously does not hurt the generation performance of the model.

MODEL	SIZE	VALIDITY (↑)	UNIQUENESS (↑)	NOVELTY (↑)	FCD (↑)	KL DIV (↑)
LSTM	-	0.96	**1.0**	0.91	**0.91**	**0.99**
VAE	-	0.87	**1.0**	0.97	0.86	0.98
AAE	-	0.82	**1.0**	**1.0**	0.53	0.89
JT-VAE	-	**1.0**	N/A	N/A	0.76	0.94
MAGNET	6.9M	N/A	N/A	N/A	0.73	0.92
MOLER	-	**1.0**	0.99	0.97	0.78	0.98
MOLGPT	6M	0.98	**1.0**	**1.0**	**0.91**	**0.99**
MOLGPT (OURS)	6M	0.97	**1.0**	0.97	0.89	0.98
JT (OURS)	6M	0.97	**1.0**	0.98	0.89	**0.99**
JT (OURS)	50M	0.98	**1.0**	0.95	0.90	**0.99**

absolute error, of the JOINT TRANSFORMER fine-tuned on three properties from the Guacamol task (Brown et al., 2019) do not change between the test set and newly generated data (Table 3). This shows good generalization performance of JOINT TRANSFORMER.

Table 3: Mean absolute prediction error (MAE) for the predictor on three property prediction tasks on test and generated data. Mean and standard deviation across independent runs.

METHOD	DATA	PERINDOPRIL MPO	SITAGLIPTIN MPO	ZALEPLON MPO
JOINT TRANSFORMER	TEST	0.014 ± 0.004	0.009 ± 0.001	0.012 ± 0.001
	GENERATED	0.015 ± 0.004	0.009 ± 0.002	0.012 ± 0.001

E   Implementation Details

E.1   Data and Tokenization

We use SMILES (Weininger, 1988) based representations of molecules across all experiments. In all experiments we pre-train the JOINT TRANSFORMER in an unsupervised manner using the ChEMBL database, a manually curated database of molecules with drug-like properties (Mendez et al., 2019). As opposed to other datasets like ZINC (Irwin et al., 2020), ChEMBL contains only molecules which have been synthesized. To ensure reproducibility and comparability with molecule generation baselines we use version 24 of the database that contains 1.8M compounds altogether and apply standard data processing used in the Guacamol benchmark (Brown et al., 2019). As for tokenization of the data, we use a tokenizer based on (Schwaller et al., 2020). We additionally use an augmentation method of SMILES representations based on (Tetko et al., 2019) and similar to (Bagal et al., 2022) across all experiments and methods. This ensures the transferability of results obtained by Bagal et al. (2022) to our experiments.

E.2   Architecture

Our implementation of the JOINT TRANSFORMER follows the implementation provided by (Karpathy, 2023), which is a re-implementation of a GPT-2 (Radford et al., 2019) used by MolGPT (Bagal et al., 2022). The only difference is that during each forward pass, we switch between a causal and a bidirectional masking, depending on the task we are optimizing for. We additionally stack an MLP network on the top of the first output token for prediction. The complete list of hyperparameters is presented in Table 4. Our implementation results in a model with 6.5M parameters.
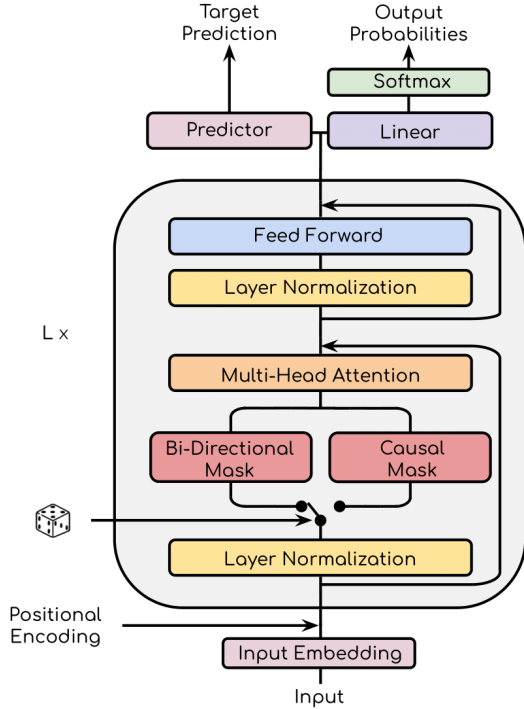
Figure 2: JOINT TRANSFORMER architecture.

Table 4: Model hyperparameters for the JOINT TRANSFORMER used across all experiments.

HYPERPARAMETER	VALUE
ACTIVATION FN	GELU
EMBED DIM	256
NUM LAYERS	6
NUM HEADS	8
FEEDFORWARD DIMENSION	1024
FEEDFORWARD BIAS	FALSE
LAYER NORM EPSILON	1e−5
PREDICTOR HEAD	MLP
PREDICTOR NUM LAYERS	1
PREDICTOR HIDDEN DIM	100

E.3 Training

We provide the complete list of hyperparameters used for training JOINT TRANSFORMER in Table 5. JOINT TRANSFORMER was trained on a single NVIDIA GeForce RTX 2080 TI GPU for 4.2M iterations that took approximately seven days.

E.4 Fine-tuning

As JOINT TRANSFORMER is a joint model, fine-tuning is achieved by standard training (Alg. 3) on the supervised part of the dataset. Unless stated otherwise, we use the same set of hyperparameters for fine-tuning across all tasks, summarized in Table 6. Fine-tuning on a single NVIDIA GeForce RTX 2080 TI GPU for 50K iterations takes approximately an hour. Hyperparameters not listed in Table 6 are shared with the pre-training task.

Table 5: Training hyperparameters of the JOINT TRANSFORMER used across all experiments.

HYPERPARAMETER	VALUE
BATCH SIZE	64
TOTAL NUMBER OF TRAINING ITERATIONS	4.2 M
OPTIMIZER	ADAMW
WEIGHT DECAY	1e−1
BETA 1	0.9
BETA 2	0.95
MAXIMUM LEARNING RATE	6e−4
MINIMUM LEARNING RATE	6e−5
DECAY LEARNING RATE	TRUE
WARMUP ITERATIONS	2000
NUMBER OF LEARNING RATE DECAY ITERATIONS	4.2 M
VALUE TO CLIP GRADIENTS AT	1.0
DROPOUT	0.1
TASK PROBABILITY p_{task}	0.95

Table 6: Fine-tuning hyperparameters for the JOINT TRANSFORMER used across all experiments.

HYPERPARAMETER	VALUE
DECAY LR	FALSE
LEARNING RATE	3e−5
NUM OF ITERATION	50K
TASK PROBABILITY p_{task}	0.1